Die approbierte Originalversion dieser Dissertation ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

DISSERTATION

An agent-based model of reality in a cadastre

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

Univ.-Prof. Dr. Andreas Frank
E127
Institut für Geoinformation und Landesvermessung

eingereicht an der Technischen Universität Wien
Fakultät für Technische Naturwissenschaften und Informatik

von

Steffen Bittner
Matrikelnummer 9726778
Institut für Geoinformation und Landesvermessung E127
Gusshausstaße 27-29
1040 Wien

Wien, August 2001 ....

An agent-based model of reality in a cadastre

by

Steffen Bittner

A thesis submitted in partial fulfillment of the requirements
of the degree of
Doctor of Technical Sciences

submitted at the
Technical University Vienna
Faculty of Science and Technology

Advisory Committee:

Univ.-Prof. Dr. Andrew Frank
Institute for Geoinformation
Technical University Vienna

Univ.-Prof. Dr. Thomas Eiter
Knowledge-based Systems Group
Technical University Vienna

Vienna, August 2001 ....

Acknowledgements

This page is dedicated to all people who helped me with my thesis.

First of all I would like thank Prof. Frank for his guidance over the last two and a half years. He was a good advisor for me and discussing my topic with him was always very helpful and encouraging. I thank him for his patience and support. He found the right words at the right moment and perhaps he is not completely aware about how important it was for me.

I would like to thank my second advisor, Prof. Eiter, for his support in the end of my thesis project.

I appreciate my colleagues at the Institute for Geoinformation for their input and criticism and for the fruitful working environment at the department. Here I would like to mention Annette von Wolff for many conversations, not only about science, Edith Unterweger, without her nothing goes at the institute, and my roommates over the last years: Peter Wenzl, Rudi Müller and Andreas Grünbacher. In particular I would like to thank Mrs. Markwart who helped me to improve the quality of my English to a (hopefully) reasonable quality.

This research was supported by the CHOROCHRONOS TMR network of the European Union, by the REVIGIS project and by the FWF project on ontology of a cadastre which gave the financial background for my thesis work.

My special thanks go to Katrin. Without her I am sure that I would not sit here and write these lines. She kept me on the way when I was losing the direction, when I was doubting, when I wanted to give up. She gave me the power to get my thesis project managed. I thank her for her comfort in the bad hours and for the many happy hours we had during the last years in Vienna.

Nicht vergessen moechte ich an dieser Stelle meine Eltern, die meinen Weg über so viele Jahre unterstützt haben, sei es als ich laufen lernte, als ich in die Schule ging, und auch später, als sie mich loslassen und meine eigenen Wege gehen lassen mußten. An allem, was ich erreiche, habt Ihr einen Anteil, denn Ihr habt einen bedeutenen Teil von mir geformt. Deshalb möchte ich Euch diese Arbeit widmen.

Abstract

This thesis investigates the institutional structure of social reality. In order to make the analysis possible we focus on a specific domain, the domain of land management, in particular the Austrian cadastre. Since the field of cadastre is highly determined by laws we are able to investigate the structure of reality in a cadastre by the analysis of the cadastral law. We investigate social processes and focus on appropriate case studies, which are the transfer of ownership of a parcel between two persons and the conflict regarding land use between two persons and its resolution by the organizations of the state.

The thorough analysis of the institutional structures must rely on a sophisticated theoretical foundation and this thesis applies Searle's theory of institutional reality as its major ontological background. Institutional reality can only be comprehensively described by investigating its relationship and interaction to physical reality and Searle's theory gives the appropriate framework for the analysis. Differing from Searle we assume that the institutional structure of society can be described in terms of the individual knowledge and interaction of the actors in the society.

The basic approach of this thesis is the construction of a computational model which can be evaluated by simulating social processes in the model. The correctness of the model and thus of the analysis can be shown by assessing the correspondence between the simulation and the real world process.

The model of institutional reality in a cadastre must include representations of the individual knowledge, intentions and behaviour of the actors involved since they are the central element determining its structure. We represent individuals in the model by agents and use an agent-based approach as the general conceptual framework for the model construction process.

We assume that aspects of the real world can be represented on an abstract level in terms of algebra. This thesis applies an algebraic modeling approach for the representation. We algebraically specify the essential properties of reality and construct a computational model, which forms an algebra and includes the desired properties specified. We use the functional programming language Haskell, which supports this algebraic specification style of the model construction process.

The result of the model construction process is an agent-based simulation model, which correctly represents the social processes under consideration. The simulation model can be used to develop applications. We apply the model to show how to assess transaction costs within the simulation and provide one foundation for the objective comparison of cadastral systems.

This thesis has a theoretical and a practical dimension. One the theoretical side we contribute to the improvement of our understanding about the structure of social reality. On the practical side we show how to construct tools, which can support the development of more efficient cadastral systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 Institutions

"Institutions are the rules of the game in a society..." (North 1997, p.3)

How much of the real world around us can be explained without investigating institutional rules? Any human social interaction relies on institutions. We can talk about physics and chemistry without thinking of institutions; but even in the case of biology we must distinguish biological and social phenomena in order to clarify what biology is about. Social science, law, politics, economy, sociology, everything where social interaction plays a role, needs the investigation of institutions. We cannot *buy* a cup of coffee, cannot *marry* each other, cannot *elect* new *presidents*, cannot be *employed* without the institutional structure defining the concepts and rules for buying, marrying, electing and employing and explaining what a president is.

Much research has been done to understand physical and social reality around us. There is only one reality and the question arises how both parts are connected. How much can we say about the institutions without investigating the physical nature of the objects involved and the activities they define and regulate? Not much. We cannot talk about *buying* a cup of coffee without investigating the value of a cup of coffee, which is determined by the physical properties of the coffee and my physical desire to drink it. We cannot investigate this process without thinking of the physical activity of boiling the coffee and of the physical activity of handing over a physical piece of paper or metal, which has the corresponding value to the value of the cup of coffee. Similar considerations hold for *marrying*, *electing* and *employing* and other social concepts.

This thesis is devoted to the task of analyzing the institutional structure of social reality and its relationship to and embedding into the physical environment.

Society is a really complex matter and in order to enable the analysis of its general structure we have to focus on some particular domain, which allows reducing the complexity of the phenomena involved. It is the *art of research* to find the appropriate case studies that allow the investigation of phenomena with a minimal amount of complexity involved (Frank 2001). The improvement of the understanding of the small part of reality under consideration contributes to the understanding of reality in general. In order to find these minimal case studies, which allow the investigation of the institutional rules, we look at land management systems and in particular at the cadastre.

1.1.2 Cadastre

"Land, after all, is the ultimate resource from which all wealth comes." (Dale & McLaughlin 1989, p.21)

Looking at land management we talk about one of the foundations of the human existence: land. Without it human settlement and life would be impossible. Its value and importance is therefore extremely high. Land cannot be increased, it is a limited resource and an important factor of production. Much effort is invested into the management of land. Cadastral systems are developed all over the world for this purpose. Consequently a major demand for effective and efficient organization of these systems exists (Dale & McLaughlin 1989).

The foundation for efficient cadastral systems is the understanding of the reality which the system should correctly represent. It is not sufficient to investigate only the cadastral registry with its content and input and output operations. The registration process in the cadastral registry captures only a part of reality. The complexity of phenomena involved makes it necessary to widen the scope to the more general view of *reality in a cadastre* that comprises the cadastral registry as well as people acting in the real world. By reality in a cadastre we understand the part of the real world which is influenced by the activities and the content of the cadastral registry. This allows representing a more comprehensive view of the cadastral domain and the analysis of its institutional structure.

The example of ownership transfer under Austrian law shows the characteristic institutional and physical structure of reality in a cadastre. The ownership transfer of a parcel is determined by the following steps: The owner of a parcel makes an offer to another legal person. This legal person accepts the offer. Both will sign a sales contract. For the seller this contract is connected to the duty to transfer the ownership of the parcel. According to his duty the seller applies for the ownership transfer at the cadastral registry. The ownership transfer occurs by the registration in the cadastre. Ownership of a parcel means for the new owner the right to use the piece of land described as parcel in the cadastral registry. It includes the right to exclude any other person from using the land. It comprises the right to mobilize the physical power of the state to prevail against any other person not respecting the right.

The example shows how reality in a cadastre is determined by a variety of physical phenomena and institutional constructs. On one side there are physical phenomena, for instance, land pieces and human beings acting on land. On the other side there are social constructs, for instance, ownership, legal persons and rights. These social constructs influence the behaviour of people in physical reality and thus play a crucial role for the analysis.

Why is the cadastre the appropriate field to find the minimal case studies for the analysis of institutions? First of all the example given has shown that reality in a cadastre is determined by the institutional structure and its embedding into the physical environment. From this viewpoint the cadastre is appropriate for the analysis of institutions. The second point is: Why is the cadastre appropriate to find *minimal* case studies? First, it is a relatively separated part of reality with limited influences from and to other parts of the real world. So most aspects of the real world outside the field can be neglected and the complexity of the analysis reduced. Second, the rules structuring social reality in a cadastre are known to a high degree and are represented in laws. Reality in a cadastre is mainly determined by these written rules and unwritten rules, e.g., common laws or rules of conduct, do not play an essential role. This reduces the research effort and makes the field open to the analysis. Third, the institutional rules in the field of cadastre are relatively simple. The reason is that economic efficiency is a major criterion for the construction of cadastral systems, which enforces the construction of systems with rules as simple as possible.

Concluding we selected the field of cadastre for the analysis, because it has the institutional structure we are interested in, its complexity is limited and it allows the investigation with reasonable effort by the analysis of the cadastral law. From the practical point of view, the analysis of reality is of high relevance because of the importance of the resource it manages: the land.

1.1.3 Computational models

The construction of models is an approach to improve our understanding of reality, which is widely used in science. We follow this approach by constructing a model that represents the cadastral registry embedded into its environment and its interaction with the environment, in particular the interaction with people acting in reality.

There are tools necessary that support and improve the model construction process. Computational models are appropriate for this purpose, because they allow the checking of the model by executing the program and the assessment of its behaviour. For instance, a model representing reality in a cadastre, enables the execution of processes according to the case studies selected for the analysis.

It is advantageous to use a formal tool for the model construction, i.e., a formal specification language with clear semantics. This allows representing the domain in a clear and unambiguous way and helps to avoid misunderstandings. The formal model should be executable in order to enable the construction of computational models. Computational models based on a formal executable specification have advantages compared with models not based on a formal foundation. A model based on a formal specification formally correct represents its specification, i.e., it does the right things with respect to the specification. This simplifies the testing of the model with the case studies and the assessment of its correspondence to the reality it represents.

1.2 Research Question

Assuming that our understanding of the reality around us can be improved by the construction of computational models and that the field of cadastre is appropriate to analyze essential aspects of this reality, the research question of this thesis is the following: Is it possible to construct a computational model of reality in a cadastre?

Since we are interested in the institutional structure of reality and selected the cadastre, because its essential property is its institutional structure, the central issue for the model and this thesis is the representation of the relevant aspects of social reality. The key issue for the analysis of reality in a cadastre is the understanding of the social concepts involved.

Formal computational models have advantages compared with non formal approaches enabling the maintenance of the internal correctness of the model. The main research question of this thesis is therefore:

> *Is it possible to construct a computational, formal model of social reality in a cadastre?*

In order to find sophisticated answers to the research question, the whole issue must be divided into several sub-questions. We distinguish questions about reality from questions about the structure of the model. The major question about reality is the question about how it is structured. The major questions about the structure of the model are the issues of the conceptual framework of the model, its transformation into the computer program (the computational model) and the proof of the correctness of the model. The sub-questions to be answered by the thesis are:

1. How is the structure of social reality in a cadastre characterized?
2. What is the conceptual framework for the model?
3. How can the conceptual framework be transferred into a computational model?
4. If such a model can be constructed can it be shown that it corresponds to the reality it represents?

1.3 Hypothesis

The hypothesis is our answer to the research question, which we will prove in this thesis. The general assumption about reality, which corresponds to our motivation to select the cadastre for the analysis, is that reality in a cadastre is determined by the institutional concepts (e.g., ownership, parcel,...) and their relationships to the physical environment (e.g., land, human beings).

The philosophical foundation for the analysis of the institutional concepts involved, is Searle's theory of institutional reality (Searle 1995). It introduces an approach dealing with institutional concepts and gives the theoretical background of this work. Social reality in general and institutional reality in particular are determined by the behaviour of human beings. Thus for the model a representation of human intentions and behaviour is necessary. Multi-agent theory (O'Hare & Jennnings 1996, Ferber 1999, Wooldridge & Jennings 1995) gives the conceptual framework for this task. It allows formally representing human intentions and behaviour by agents, as far as necessary for the model of reality in a cadastre.
The hypothesis of this thesis reads as follows:

> *A computational, formal model of institutional reality developed in an agent-based framework represents the relevant aspects of reality in a cadastre.*

This hypothesis positively answers the research question: Yes, we assume that we are able to construct a computational model of reality in a cadastre.

1.4 Approach

According to the general idea of explaining things in terms of computational models, the basic approach of this thesis is the construction of computational model of reality in a cadastre. The correctness of the model will be verified by agent-based simulation of appropriate case studies from the real world in the constructed model.

1.4.1 Simulation models

Simulation is a specific way of modelling. The simulation method comprises the following decision steps (Gilbert 1993):

1. No social phenomenon can be examined entirely. The first step is to select the aspects which are of interest. This selection must be influenced by theoretical preconceptions about which features are significant.
2. The modelling approach to be adopted must be chosen. A widely used simulation method is, for instance, the simulation with differential equations. A newer approach is agent-based simulation.
3. The decision about the appropriate level of abstraction: the appropriate level of aggregation must be selected for the units of the simulation (people or organizations).
4. The representation form must be selected. It is typically a computational language.
5. After making these decisions, the model can be constructed, the simulation run and the output examined.

These design decisions determine the approach of this thesis: We apply an agent-based simulation approach on the theoretical foundation of Searle's theory to model reality both on the level of organizations (such as the cadastral registry) and on the level of people (such as owners) for the construction of a model represented in the functional programming language Haskell.

1.4.2 The three stages structure of the thesis

From the basic approach follows a three stages structure of this thesis:

1. The analysis of reality in a cadastre.
2. The construction of the computational model.
3. Agent-based simulation.

The analysis of the domain in question (reality in a cadastre) is the foundation of the model construction. The main task of this step is to identify the relevant features of reality necessary for the representation and the investigation of the structure of the domain. The first step can be characterized as ontological analysis. The result is an ontology, i.e., a conceptualization of reality in a cadastre. We distinguish the construction of the ontology from the representation of the ontology in the model in the second step. The second step comprises the model construction itself, with the claim to capture the relevant aspects of reality identified in step one. The task of the third step is to check of the correctness of the model with respect to the domain in question.

1.4.3 Characteristics of the approach

The following tasks characterize the approach of this thesis:

Start on a *sophisticated philosophical foundation* with an *analysis of the domain* in question (reality in a cadastre). The idea is that the analysis must be based on a solid theoretical foundation, in particular explaining the structure of the institutional part of reality. The foundation on which this thesis is based is mainly Searle's theory of institutional reality.

Use of an *agent-based model* as *conceptual framework* to represent the domain. We regard agent-based models as purely conceptual background for the representation of the model. This distinguishes the approach of this thesis from others which investigate agents as new technology. Our approach allows developing the conceptual ideas independently from a particular formalism or representation mechanism.

Use of an *algebraic specification* as *formal framework* to represent the domain. Focusing on the conceptual ideas, nevertheless this work needs a representation framework. Algebraic specifications (Loeckx, Ehrlich & Wolf 1996, Ehrlich, Gogolla & Lipeck 1989, Horebeek & Levi 1989) allow representing the domain in question in terms of algebra. We regard algebras as appropriate way to model parts of the real world. We assume that the mathematical structure of an algebra has similarities to the structure of reality we want to represent. Algebras model reality on an abstract level.

***Implementation* of the algebraic specification in the *functional programming language Haskell* to achieve a computational model.** A language is necessary which is capable of expressing algebraic specifications and is additionally executable. The functional programming language Haskell (Thompson 1996, Hudak, Peterson & Fasel 1997, Bird & Wadler 1988) allows the construction of executable algebraic specifications.

***Validation* of the model by agent-based simulation.** We assume that it is impossible to analytically prove the correctness of a nontrivial model of a part of the real world. The validity of the model can be assessed by agent based simulation comparing the output of the simulation with the situation in reality.

Restriction of the model to appropriate *case studies*. We restrict the model to the simulation of expressive case studies, since, due to lack of space, it is impossible to construct a comprehensive model of reality in a cadastre. We use case studies which we regard as typical for reality in a cadastre.

Outline possible *applications* of the model. We assume that the model is applicable, i.e., that it is possible to provide solutions to practical problems. We will outline the ideas of possible applications of the model.

1.5 Foundations

This thesis is interdisciplinary work. Hypothesis and approach determine its foundations which we outline in this section. The thesis connects work from very different fields:

- The legal domain: Cadastral systems and cadastral law
- Philosophy: Ontology of social reality
- Artificial Intelligence (Russell & Norvig 1995): Multi-agent theory and multi-agent systems
- Social Science: Simulation of social processes
- Computer science: Algebraic specifications and functional programming.

1.5.1 Cadastre

The task of the cadastre is the administration of land. A cadastre is a parcel-based land registration system, i.e., all data are grouped around the cadastral parcel. The cadastre can serve different purposes, for instance, for the public authority to assess taxes, or for private persons the management of rights private persons hold on land.

Ownership is the most comprehensive right a person can have on a parcel. It is the right to use a piece of land and to exclude all other persons from using this land. Ownership is the central element determining the work of the cadastral registry.

Cadastre is embedded into the private law. The private law regulates relationships between private persons and their relationships to objects. The cadastre deals with a specific class of objects, which are immovable things, i.e., parcels. Legal change in a cadastre is caused by the transfer of rights on land.

An important task of the legal system is to maintain the peaceful living together of people (the peace keeping function). For this purpose the state provides legal instruments for persons to enforce their rights. These instruments are complaints, and after the decision of a court the execution process which physically exerts force against a person not recognizing the rights of others.

1.5.2 The ontology of social reality

Searle's theory of institutional reality focuses on the part of reality determined by institutions. It deals not with social reality in general. Searle distinguishes a physical and an institutional level of reality. Physical reality exists independent of human observers, institutional reality depends on human cognition. Institutional facts existing within institutional reality are characterized by the fact that they assign some status to a physical phenomenon, which cannot be performed only by the physical properties of the object. For example, the function 'money' of a piece of paper, its value, is not derivable from the physical properties of paper. Money exists because we collectively accept particular pieces of paper as medium of exchange and value.

1.5.3 Multi-agent theory

An agent is a entity that is capable of acting in its environment and capable of perceiving its environment. A set of agents which are able to interact with each other and with the objects in the environment form a multi-agent system. The key properties of agents are autonomy and the embedding of the agents into the environment. Autonomy means that the agent is capable of acting based on its percepts and individual experiences. The agent must be embedded into the environment, i.e., it must be an integral part of the environment.

Agents can be constructed according to different architectures. An agent with internal state has an explicit representation of the environment whereas a reactive agent simply reacts to the influences from the environment.

1.5.4 The simulation of social processes

The idea of a simulation is to construct a computational model of a part of the real world, let the model execute and compare its output data with the observations from reality. The model correctly simulates processes from reality if the output data from the simulation and the observations from reality correspond.

Agent-based simulation provides a new solution to the simulation of social processes, because it allows to include representations of individuals with individual capabilities and preferences into the model.

1.5.5 Algebraic specifications and functional programming

An algebraic specification consists of a set of sorts and a set of operations between these sorts. The properties of the operations are described by a set of axioms. An algebraic specification describes a class of algebras, where each algebra is a mathematical structure consisting of a number of sets and functions mapping elements of the sets to each other. Algebras can be regarded as abstractions from the real world preserving the structure of real world phenomena.

Algebraic specifications can be expressed in a functional programming language. That means, functional languages support an algebraic specification style of programming. In a functional language everything, i.e., every object, is a function. New functions can be defined based on other functions. An important feature of functional programs, which limits sources of errors, is that they do not produce side effects.

1.6 Goal and scope of the thesis

This section defines the goals we try to achieve in this thesis and defines its scope. The general goal of this thesis is to improve the knowledge about the structure of reality in a cadastre and about social reality in general. Its task is to develop a *content theory* in opposition to a mechanism theory (Chandrasekaran, Josephson & Benjamins 1999). The thesis is mostly interested in reality in the cadastral domain and not so much in the representation mechanism. The focus is directed to the analysis of reality and a conceptualization of the phenomena influencing reality. It does not investigate the formal properties of a representation mechanism for agent-based simulation. It uses a representation mechanism to apply the conceptual framework developed.

This thesis focuses on the Austrian cadastral system (Marent & Preisl 1994, Twaroch 2000) to achieve a more realistic model. It concentrates on the legal part of the cadastre, omitting spatial issues because the issues raised by the research question are mainly situated in this field. The goal is not to construct a comprehensive simulation model of the whole complexity of reality in a cadastre. The goal is to construct the model according to appropriate case studies.

The case studies we select for this thesis are intended to be expressive and minimal. They are expressive in the sense that they show the institutional structure of the domain we are interested in and minimal in the sense that they are as simple as the issues under investigation allow. These case studies are the transfer of ownership of a parcel and conflicts regarding the use of a piece of land between the owner, the authorized user of the land, and the unauthorized user of the land. The conflict leads to a legal action by the owner and a judgement execution to end the unauthorized land use. The result of these case studies are simulations of social processes in a cadastre.

1.7 Expected achievements

The expected result of this thesis is that it is possible to construct a computational model of reality in a cadastre, which can be successfully validated with the case studies. The development of a computational model in an executable specification language allows the validation by testing the specification. A realistic formalization will be achieved by the orientation to the concrete Austrian system. It will be expected that the analysis of reality in a cadastre leads to a general ontology of the cadastral domain which is applicable to other legal systems. The agent-based framework to be developed in this thesis will be extensible to larger parts of the Austrian cadastral law as well as to other legal domains, possibly from other legal systems. From the choice of the functional programming language we expect a clear and understandable representation that helps to avoid mistakes and support the expression of the ideas of this thesis.

The scientific contribution of a computational model of reality in a cadastre has a practical and a theoretical dimension. It improves the understanding of reality in a cadastre and supports the construction of efficient cadastral systems on the practical side as well as a more realistic view of the world around us on the theoretical side.

We expect that it is possible to apply the model to solve practical issues from real-world cadastral systems. If the computational model of the thesis support this assumption, we will outline the ideas of the application of the model.

1.8 The structure of the thesis

The structure of the thesis follows the three stages approach (see subsection 1.4.2).

1.8.1 The first stage - the ontology

The first stage, the ontological analysis of reality in a cadastre, comprises the chapters 2, 3 and 4.

In chapter 2 we introduce the cadastre. We present the general ideas and focus then on the Austrian legal system. We discuss the embedding of the cadastral law into the whole legal system. The chapter introduces characteristic processes from a cadastre, focusing on the transfer of rights and the conflict resolution between persons.

Chapter 3 introduces the philosophical background of this thesis. It discusses the structure of institutional reality mainly based on Searle's theory of institutional reality. It develops an ontology of institutional reality applying Searle's theory. The chapter relates the concepts to the corresponding terms from philosophy and the positive law, introduced in chapter 2.

Chapter 4 investigates the structure of reality in a cadastre based on Searle's theory of institutional reality. The result is a comprehensive analysis of reality in a cadastre, which is the essential foundation for the model construction.

1.8.2 The second stage - the model

The second stage, the construction of the computational model of reality in a cadastre, comprises the chapters 5, 6 and 7.

Chapter 5 introduces multi-agent theory as the conceptional framework for the model construction. It gives an overview of the concepts of multi-agent theory as a very heterogeneous field of research. It has the goal to clarify the approach this thesis follows, and to differentiate it from other viewpoints of the agent domain. The main task of this chapter is the development of a general, abstract and domain independent architecture of a multi-agent system applicable for the model of reality in a cadastre.

Chapter 6 discusses the formal framework for the model construction. It introduces the algebraic specification style implemented in the functional language Haskell as representation mechanism used for the construction of the agent-based model in this thesis.

Chapter 7 describes the agent-based model of reality in a cadastre. It derives the components of the model from the analysis of chapter 4 and discusses the construction of the model based on the abstract agent architecture of chapter 5. It introduces the basic ideas of the Haskell implementation of the model.

1.8.3 The third stage - the simulation

The third stage, the simulation of social processes in a cadastre, comprises the chapters 8 and 9.

In chapter 8 we discuss the agent-based simulation of ownership transfer and the process to enforce rights within the legal system consisting of a legal action and the judgement execution. We introduce the input and the output of the simulation in detail. Based on the output we investigate the flow of operation during the simulation and compare the results with the description of the processes given in chapter 2.

In chapter 9 we outline the idea of an application based on agent-based simulation. We introduce a method for cost assessment for the parties involved in the process.

Chapter 10 gives a conclusion and describes the contributions of this work. It discusses directions for future work.

Chapter 2

Cadastre

2.1 Introduction

This chapter introduces the legal background of the domain investigated in this thesis. It discusses cadastral systems and the laws determining the work of the cadastral registry. It gives an overview of the main concepts of a cadastre in general as well as of the Austrian legal system, which is the foundation for the model constructed in this thesis.

The chapter describes the concepts involved to the extent necessary for the purpose of this thesis. The domain is influenced by many laws. Twaroch (Twaroch 2000) lists more than 40 (Austrian) laws which directly influence real estate. This chapter does not intend to be a comprehensive introduction into cadastral systems and cadastral law. The model to be constructed in this thesis is mainly based on the legal norms codified in two laws: the ABGB (*Allgemeines bürgerliches Gesetzbuch* (ABGB 1811)) and the GBG (*Grundbuchsgesetz* (GBG 1955)). Section 2.12 introduces and justifies in detail the constraining assumptions we make about the law.

For the investigation of reality in a cadastre it is not sufficient only to discuss the cadastral law itself. It is necessary to describe the embedding of the cadastral law into the general legal system. For example, many rules relevant for a cadastre do not only apply to parcels. They apply to legal objects in general and are therefore not directly formulated in the cadastral law.

The legal status of land is determined by the question who holds rights on pieces of land. This chapter emphasizes the discussion of rights and of the objects and subjects of rights. It discusses change, i.e., legal transactions causing change in these rights.

The question of documentation of legal transactions and of the current legal status of land is an important issue for a cadastre. Documentation is necessary to prove rights as well as to constitute rights in a cadastre. This chapter introduces the main rules of documentation in the Austrian cadastre.

The legal system allows people enforcing their rights with the help of the coercive force of the state. How to use the force of the state for the enforcement of rights is an important factor influencing reality in a cadastre. This chapter describes the processes necessary to enforce rights. It discusses legal actions and the execution process in the context of a cadastre.

The chapter starts with the discussion of the general principles of land registration and of the main categories of systems existing in the world. It then focuses on the Austrian law in particular and describes its main principles. Based on these main principles it discusses rights, objects and subjects of rights and legal transactions. Then follows the description of the system of documentation and the introduction of the rules to enforce rights with the help of the state. The chapter concludes with the discussion of the assumptions we make about the law in this thesis.

2.2 Systems of land registration

Land is the foundation of human settlement. It cannot be increased, it is a limited resource. The purpose of land tenure systems is to manage this important resource. A land registration system is a system of documentation that supports this social task.

Land is permanent by nature, but the rights associated to it and their extent change. All over the world there exist many different systems to record legal change on land pieces. Whereas the systems strongly differ in detail, there are two principal approaches to land registration:

- the *registry of deeds*,
- the *registration of title to land.*

Deed recording systems are widely used, for instance, in the United States, title registration systems exist, for instance, in the countries of central Europe. In order to prove who holds a right on a particular piece of land it is necessary to investigate the *title*. The title is "the evidence of a person's right to property" (Dale & McLaughlin 1989, p.19).

A deed is a written and signed instrument that conveys some interest in property (Garner 1996, p.172). In a deed registration system, copies of each deed, proving the transfer of a right on a parcel, are stored in a public register. The basic unit of registration is the legal document. These documents are the primary claim to ownership of rights. To prove the ownership of a right a person has to establish the 'chain of title', i.e., the complete list of legal transactions represented by the deeds stored in the registry that justify his title. The title of a person is based on the title of his predecessor.

In a title registration system the basic unit of registration is the land parcel. The rights that people hold on a parcel are recorded in the registry, i.e., the registry record is the primary claim to ownership. The content of the registry is guaranteed to be correct by the state. Damages caused by incorrect entries in the registry will be compensated by the state.

A cadastre is a parcel-based land information system. That means that all data is organized around the cadastral parcel. A parcel, in a general sense, is a proprietary land unit, which has an own identity. It is a continuous area of land for which unique and homogeneous interests are recognized (There are objects, which are treated similarly, such as ships, mines and airplanes). The principal function of a cadastre is to provide data concerning land ownership, land value and land use (Dale & McLaughlin 1989). The goals of a land registration system can be categorized into three areas (Twaroch 2000):

1. Private and economical purposes: security of ownership and credits.
2. General economical and political purposes: real estate management, stimulation of the real estate market.

3. Public purposes: tax assessment, inspection of the land market, landscape development.

According to its primary purpose a cadastral system can be classified as (Dale & McLaughlin 1989, p.13):

- Juridical cadastre,
- Fiscal cadastre.

A juridical cadastre is a legally recognized record of land tenure, i.e., it is primary concerned with the legal aspect of land management. It deals with rights persons have on pieces of land.

The main purpose of a fiscal cadastre is property valuation for taxation. It contains information necessary for this purpose. It provides the necessary information to determine the value of each parcel and the tax due to it.

The Austrian cadastral system is an instance of a title registration system. Its primary purpose is to determine the legal situation of land parcels, therefore it is a juridical cadastre.

2.3 Parts of a cadastre

A (juridical) cadastre usually consist of two parts (Dale & McLaughlin 1989, p.25):

1. The description of the parcels in form of either maps or survey measurements.
2. The record of the attributes associated with each parcel, in particular, the name of the owner and the rights associated with the land.

The first part is called the descriptive part and the second part is called the attribute part of the cadastre. The function of the cadastral map or description is to identify the land piece in reality to which the attributes refer. The two parts of the cadastre are cross referenced to maintain the correspondence of both parts.

The Austrian cadastre follows the partition into two parts. It consists of

1. the real estate register (*Kataster*[1])
2. the land register (*Grundbuch*)

The real estate register contains the descriptive part of the cadastre in form of cadastral maps. The land register is the attribute part of the cadastre. It mainly contains information about the legal status of the parcels, in particular information about the owners of parcels and the rights associated with the parcel.

2.4 The monopoly of force

The monopoly of force is the foundation of most legal systems. The legal system is the sum of all norms governing the living together of people which are enforceable by the power of the state (Krejci 1995, p.1).

[1]Translations from (Dale & McLaughlin 1989)

The state claims the monopoly of force, i.e., the state claims to be the only one who uses violence. The goal of the monopoly of force is to avoid that people enforce their rights themselves. The state alone exerts force with the goal to achieve justice. It includes a third neutral party into the exchange between people, which is able to enforce arrangements.

The monopoly of force provides an economic benefit to the society. It regulates the relationships between people. It reduces the uncertainty in the exchange between human beings and supports in this way economic growth and increases the wealth of the society (North 1997).

The monopoly of force has peace keeping function avoiding that everybody enforces his interests by physical violence. This is a major foundation of the peaceful living together of people.

The monopoly of force has a protection function. Weak persons should be able to enforce their rights with the help of the state. The monopoly of force gives all humans respecting the rules of the system equal chances to protect their interests.

The legal system gives people that act corresponding to its rules and norms the possibility to use the force of the state to act according to their interests also against other people. It gives everybody the same possibilities to use the power of the state to maintain his rights.

2.5 The structure of the Austrian legal system

This section describes the structure of the Austrian legal system. Reality in a cadastre is not only influenced by the cadastral law itself. Many legal concepts essential for the description of reality in a cadastre are not directly defined in the cadastral law, but inherited from other branches of law. This section describes the embedding of the cadastral law into the whole legal system of Austria.

With the Austrian legal system we describe a part of the positive law, which is a system of law implemented within a particular political community by political superiors (Garner 1996, p.488). In general the legal system consists of two fields of law (Krejci 1995, p.7):

- the material law (*materielles Recht*),
- the formal law (*formelles Recht*).

The material law defines the rules for the living together of humans. It regulates the relations between people and the relations between people and things. Material law norms regulate, for instance, the relationship between people and parcels, e.g., who owns a parcel.

The formal law defines procedures to enforce legal norms. Rules of the formal law define, for instance, how to sue another person.

The distinction of material law and formal law corresponds to the terms *substantive law* and *procedural law* from the Anglo-American legal system. The procedural law describes the rules that prescribe the steps for having a right or duty judicially enforced. The substantive law defines the rights and duties themselves (Garner 1996, p.504).

The cadastral law belongs to the field of private law or civil law (*Privatrecht*). The private law is the body of law dealing with private persons and their property and relationships (Garner 1996, p.500). The private law gives the possibility for people to regulate their relationships in a legally enforceable (i.e., with the help of the force of the state) way (Krejci 1995). The private law gives the general legal conditions for the autonomous organization of the human life. Autonomy (autonomy of the private (*Privatautonomie*)) means that the people can freely arrange their life according to the legal norms.

The cadastral law belongs to the object law or thing law (*Sachenrecht*), which is a branch of the private law. The object law regulates the relationships between people and things (or objects). The term 'object law' corresponds to the term *'property law'*, used in the Anglo-american law tradition, which is "the complex of jural relationships between and among persons with respect to things" (Britannica.com 2001)[2]. The object law regulates the immediate control over objects. An object in the legal sense is anything different from a person to which rights can relate (Krejci 1995). Objects are, for instance, cars and parcels.

Objects can be movable (personal things) or immovable (real things). Movable are all objects which can be moved without loss in substance (e.g., cars). Immovable objects are real estate. The cadastral law is a branch of the private law that regulates the relationship between people and immovable things.

The private law defines the rights themselves. Additionally rules for the enforcement of the rights are necessary in the case of conflicts. The procedural law (*Verfahrensrecht*) describes the rules for the enforcement of the material private law. For instance the procedural law defines the steps to mobilize the power of the state in the case a person does not respect the ownership of another person of a parcel. It describes how to sue the person and how to apply for the execution of the judgement.

2.6 Principles of the Austrian land register

The land register (*Grundbuch*) describes the legal status of parcels. The land register is the public register which records the parcels and rights associated with the parcels (Krejci 1995, p.174). In the context of the Austrian legal system we refine the general definition of the term 'parcel' given in section 2.2 by the following definition: A parcel is a part of the earth which is registered in the real estate register and is identified there by a unique identifier (Marent & Preisl 1994, Twaroch 2000). Parcels are the smallest entities on which a person can hold rights, i.e., rights are always related to parcels as a whole. In the land register rights are recorded referencing the number of the parcel, i.e., only with reference to the identifier of the parcel, not with reference to the parcel's concrete spatial shape and boundaries. Thus change in the legal status of parcels can be described without discussing spatial and boundary issues.

Exceptions from this general rules exist: Whereas the rights are always assigned to the parcel as a whole they can be restricted, i.e., have only consequences to a part of its whole area. A typical example is a right of way.

The Austrian land register (see section 2.3) consists of two parts: the first part records the rights on parcels and the parcel owners (the *Hauptbuch*). Its entries serve as the title to land. The second part records the deeds which were used to prove the title (the *Urkundensammlung*).

We can distinguish legal and technical principles of the Austrian land register (Twaroch 2000). We will introduce in the following six fundamental legal principles

[2]http://www.britannica.com/eb/article?eu=117332&tocid=28470

(Krejci 1995, p.176):

1. The publicity principle (*das Publizitätsprinzip*).
    - The formal publicity principle. The land register is public. Everybody can examine the content of the register in order to inform himself about the legal status of parcels.
    - The material publicity principle. A person is protected, who trusts the content of the land register, in the case that the content of the register deviates from the legal situation in reality. The damage arising from the incorrectness of the register will be compensated by the state. A person who comes in good faith can rely on the completeness and correctness of the register. The material publicity principle is also known as principle of trust or principle of the protection of good faith (*Vertrauensprinzip*) (Twaroch 2000).
2. The registration principle (*das Eintragungsprinzip*). Rights can only be constituted, changed and destroyed by the registration in the land register (there are some exceptions from these general rules, e.g., adverse possession).
3. The application principle (*das Antragsprinzip*). Registration only takes place on application. The land register does only change the content of the registry on application.
4. The principle of the registered predecessor (*das Prinzip des bücherlichen Vormannes*). Registration is only possible against a person registered as the owner of a particular right.
5. The priority principle (*das Prioritätsprinzip*). The application which is first received at the land register gets the priority.
6. The legality principle (*das Legalitätsprinzip*). Before registering the validity of the registration must be checked by the land register. Another aspect of the legality principle is that only a closed number of rights can be registered (closed system of rights) (Twaroch 2000).

Technical principles of the land register comprise the following (Twaroch 2000):

1. The determination principle (*das Bestimmtheitsprinzip*). The object a right refers to must be uniquely determined.
2. Completeness: All information about objects and rights must be completely registered and maintained.
3. Security of data: The data must be protected from unauthorized change and loss of data due to hardware or software failures.
4. Quality and precision: The data must have sufficient quality and precision for the purpose of land registration.
5. Documentation of changes: The decision making process of the registry and the change in the data itself must be documented.

2.7 The subjects and objects of rights

2.7.1 Subjects of rights

A subject of rights (*Rechtssubjekt*) is every person who is able to hold rights (Krejci 1995, p.27). A subject of rights can be every legal person. A legal person is a natural or artificial person possessing legal rights and duties (Garner 1996, p.370). The legal person is the central element of the legal system. The legal system gives legal persons the opportunity to act according to their own interests. The purpose of a land register is to manage rights legal persons hold on parcels.

Natural legal persons are humans beings. The existence of a natural person begins with the birth (nevertheless a natural person is holder of some (human) rights before his birth). The existence of a natural legal person ends with the death of the human being. A natural person does not have all rights from the beginning of his life. A natural legal person attains complete legal rights with the age of majority or legal age (Garner 1996, p.24). Complete legal rights especially are civil rights, such as the capability of agreeing to a contract, and political rights, for example, the right to vote. In the Austrian legal system a person attains complete rights (*volle Geschäftsfähigkeit*) on completion of his 19th year of life.

The law permits that groups of legal persons join and act as a single (artificial) legal person. An artificial legal person is an entity created by law and given the legal rights and duties of a human being (Garner 1996, p.479). Artificial legal persons can be organizations, such as corporations. They begin to exist by foundation and attain full rights with the moment of their foundation. Artificial legal persons act by their representatives, i.e., by natural legal persons representing the organization. In most cases it is not necessary to distinguish artificial and natural legal persons because both are able to hold the same rights and to perform the same activities.

2.7.2 Objects of rights

Objects of rights (*Rechtsobjekte*) are all objects to which rights can relate. The term 'object' describes anything different from human beings usable by human beings as defined by the object law (see section 2.5). The field of cadastral law deals with a special class of objects, with parcels, which are immovable objects.

2.8 Rights

Rights are the consequences or legal effects of legal norms. The norms of the legal system (*Rechtsnormen*) describe typical situations (*Tatbestände*) in which particular legal effects occur (*Rechtsfolgen*). These legal effects can be rights (or claims) and duties. A situation can be "A sells B a parcel C". The legal effect of this situation is (under Austrian law) the right of the buyer B to demand the transfer of ownership of parcel C from A.

2.8.1 Subjective rights

A subjective right (*ein subjektives Recht*) is the authority given by the legal system to enforce the behaviour defined by the legal system with the help of state power (Krejci

1995, p.26). It is an expectation or interest guaranteed by law (Garner 1996, p.550). The subjective right gives the holder of the right the freedom to enforce his right, or not to enforce his right, with the help of the state. According to the example of a parcel sale, B has the right to enforce his right but not the duty.

For every right a person has, somebody else has a corresponding duty (Garner 1996, p.550). A right gives somebody the authority to perform an activity and at the same time somebody else has a corresponding duty either to perform a particular activity or to refrain from performing a particular activity. If A sells B the parcel C corresponding to B's right to demand the transfer of ownership, A has the duty to perform this ownership transfer as well as the duty to refrain from selling the parcel to anybody else.

2.8.2 Absolute and relative rights

The Austrian legal system distinguishes absolute rights (*absolute Rechte*) from relative rights (*relative Rechte*). Absolute rights hold against everybody else. Parcel ownership is an instance of an absolute right. It holds against everybody else and establishes for everybody else the duty to respect the rights of the parcel owner. All property rights belong to the class of absolute rights.

Relative rights are claims. A relative right only holds against particular other persons. A contract, for instance, only establishes rights and duties for the contractors, it defines mutual claims of the participating parties.

2.8.3 Ownership

A person can own particular rights on an object or the complete object. The metaphor usually used is a bundle of sticks (Dale & McLaughlin 1989, p.19). A person can own particular sticks of the bundle or the whole bundle. Strictly speaking a person cannot own an object itself, the person owns the exclusive right to use the object according to his own interests (Dale & McLaughlin 1989, p.19).

The Austrian legal system distinguishes ownership, as the complete and most comprehensive right a person can have on an object (*Vollrecht*), from restricted rights (*beschränkte dingliche Rechte*). Restricted rights a person can have on a parcel are, for instance, mortgages and usufruct. Ownership under Austrian law is defined as the right to dispose of an object and to exclude every other person from using this object (ABGB 1811, §354). Ownership and other restricted rights are absolute rights (see subsection 2.8.2) and are as such protected against anybody.

Ownership must be distinguished from possession. Ownership is the right to possess a thing. Possession is the actual dominion over a thing. Possession of property is not recognized by the legal system as a right. Possession normally does not create legally recognized claims over a thing. An exceptional case is, for instance, adverse possession where the possession over a long time creates the ownership right.

2.9 Legal transactions

Legal transactions or transfers (*Rechtsgeschäfte*) play an important role in the description of reality in a cadastre. Nearly all change in the legal status of parcels occurs by legal transactions. The major effort in the maintenance of cadastral systems is invested into the correct representation of legal transactions in the registry. The maintenance of the correctness of the cadastral registry under occurring legal transaction is a crucial issue for land registration systems.

In general, legal transactions (under the Austrian law) are declarations of will (*Willenserklärungen*), given by legal persons, which the legal system recognizes as cause for legal effects (Krejci 1995, p.38). Legal transactions are legally enforceable declarations of will. Legally recognized declarations of will can be unilateral, e.g., a testament, or multilateral, e.g., a contract. In particular a contract is an agreement between two or more parties creating obligations that are enforceable by law (Garner 1996, p.134).

The conclusion of a (mutual) sales contract is usually performed by the following two steps: If a person is interested in an object another person owns, he makes an offer to the owner of the object. If the transferee accepts, his declaration of agreement with the offer establishes the contract.

A legal transfer of an object under the Austrian law consist of two parts or two sub-transactions. The first transaction establishes obligations to be fulfilled in the future (*das Verpflichtungsgeschäft*), i.e it creates relative rights. The second part directly influences the existing legal status of objects. It transfers, destroys and restricts rights (*das Verfügungsgeschäft*), i.e., it changes absolute rights. The first transaction establishes the obligation to perform the second transaction. The first part of the transaction is typically a sales contract. Rights are transferred during the second step, in the case of movable objects, usually by handing over the object. If a person A sells his computer to person B, A and B sign a sales contract declaring their mutual will and agreement to perform the transfer. In the second step ownership is transferred by handing over the computer to the seller.

2.9.1 The transfer of ownership

In the Austrian legal system in order to transfer ownership two preconditions have to be fulfilled. Ownership transfer needs a valid title (*Titel, Rechtsgrund*) and a valid mode (*Modus*). The title is usually given by a contract. The mode stands for a valid method of acquisition. Valid modes are the handing over of the object in the case of movable objects and the registration in the land register in the case of immovable objects. In some cases the legal system allows exceptions. Ownership transfer can occur without valid mode, for instance, for parcels in the case of adverse possession. In the case of adverse possession a person acquires ownership of a parcel without a legally recognized transaction that creates his right.

According to the two parts of the transaction (*Verpflichtungsgeschäft* and *Verfügungsgeschäft*), ownership of a parcel is transferred by performing the following activities: In the first sub-transaction both parties conclude a contract, i.e., a sales contract about the parcel concerned. The contract is a valid title for ownership transfer. It establishes the obligation for the seller to transfer the ownership in the second step to the buyer. For the buyer it establishes the obligation to pay the price for the parcel. According to his obligation in the second step the owner (and seller) transfers the ownership of the parcel to the buyer. In the case of immovable objects, i.e., parcels, ownership is transferred by registration in the land register. The registration of the ownership transfer is the valid mode for immovable objects. The buyer becomes the new owner of the parcel.

The way ownership transfer on parcels occurs in the Austrian legal system is very different from the Anglo-American system. In the American system ownership transfer occurs by performing a land sales contract (Garner 1996, p.135). Ownership is directly transferred by concluding the contract.

2.10 Documentation

2.10.1 The purpose of documentation

In the field of private law the Austrian legal system allows contracting without documentation, i.e., it is possible to conclude oral contracts without any written documents, or to conclude contracts without oral communication at all (e.g.,: take an apple and give some money to the seller and go away with the apple). If the state demands documentation he pursues the following purposes (Krejci 1995, p.60):

- secure evidences,
- publicity,
- protection.

The first point has the goal to provide the possibility to prove later on, what has been agreed. For instance, a contract must be written down in order to avoid conflicts about its content. The second point has the purpose to make the legal situation visible for every person. The third point has the purpose to protect innocent persons against fraud by making apparent what has been agreed.

For the Austrian cadastral system all three points, the securing of evidences, publicity and protection, are central elements. Publicity is one of the major principles of the Austrian cadastral system (see section 2.6). Documentation is essential for the land register to make the legal situation of parcels apparent. To avoid conflicts between people regarding rights on land, the complete and comprehensive documentation is necessary to prevent mistakes and fraud and to represent the valid legal situation in reality.

2.10.2 Documentation of deeds and documentation of title

We distinguish the documentation of contracts (deeds) and the documentation of title (the registry). By the documentation of deeds the complete list of legal transactions can be maintained and investigated, since each deed represents a single transaction. By documenting the title the legal consequences of the single transactions are represented, i.e., the current legal situation. The combination of the documentation of the transactions with the documentation of the legal consequences of these transactions form the basis of the cadastral registry.

All interaction between the environment and the registry occurs in a documented manner. The Austrian cadastral system follows the application principle (see section 2.6). Written applications are necessary to cause change in the rights. Every deed that proves a title of a person to land is recorded in the cadastral registry with the goal that no transaction can occur without documentation in the land register.

To achieve the correspondence between the legal situation in reality and its representation in the registry the change of rights and the documentation are closely linked.

The act of documenting the change in the rights itself causes the legal change. The registration of a right on a parcel establishes the transfer of the right. Neither the application principle nor the closest possible connection of documentation and ownership transfer enable the complete correspondence between content of the registry and reality (Bittner 1998), for instance, in the case of inheritance and adverse possession. Thus the documentation of the valid legal situation must remain incomplete and faulty.

2.10.3 The realization of documentation

The system of documentation can be realized in electronic form or in traditional (paper based) way. The main way of documenting the legal status is the documentation on paper. Documentation is not restricted to paper based systems. Electronic documentation is also possible in the case that the technical and organizational conditions exist to guarantee the completeness and correctness of the data. For instance, the Austrian cadastral system stores the information about the legal situation of parcels in an electronic database (*Grundstücksdatenbank*).

2.11 Processes to enforce rights

This section discusses how a person enforces his rights, i.e., it investigates cases where conflicts occur between people. It introduces procedural law, which provides the legal instruments for a person to mobilize the power of the state in order to enforce his rights.

2.11.1 Complaints

Conflicts between people are resolved according to fixed rules defined by the procedural law. The rules are equally applied to all cases and all people. Conflicts are resolved by the responsible court. The person, who thinks that his rights are violated, has to start a legal action against the person who restricts his rights. Then the person suing is called the plaintiff. His opponent is called the defendant. The complaint will be served to the defendant. The defendant has now the opportunity to answer the complaint and to explain his point of view. After that a hearing takes place in the court. After assessing all arguments and evidences a judge decides the question. The result of this assessment is then stated in a judgement.

The judgement becomes valid, if the defendant accepts the judgement, if it is confirmed by the final court or if it is not appealed after a particular period of time. If the defendant does not accept the decision of the court he has the possibility to appeal against the judgement. The appeal is the proceeding undertaken to reverse a decision by a lower court by bringing it to a higher court. It is the submission of a lower court's decision to a higher court for review and possible reversal (Garner 1996, p.36). If the court of last instance confirms the initial decision (the final judgement), i.e., the defendant's efforts to achieve a reversal of the judgment have failed, the judgement becomes valid. The obligations stated in the judgement now hold for the parties. The final judgement is the precondition for the mobilization of state force. A valid judgement establishes the execution title which is required for the application of the judgement execution.

2.11.2 The judgement execution

If the defendant after the final judgement does not voluntarily act according to the obligations defined by the judgement, the plaintiff as possessor of the execution title has the possibility to apply for judgement execution. The judgement execution is the final process to enforce the rights of the plaintiff by using the physical force of the state. The execution procedure defines the rules according to which the force of the state will be applied. After checking the validity of the execution title the execution will be carried out. A sheriff (under Austrian law a *Vollstreckungsbeamter* or *Gerichtsvollzieher*) takes away the owed object or the owed amount of money. The valid legal situation is now re-established. The plaintiff successfully enforced his right.

2.11.3 Conflicts regarding land use

In the case of conflicts regarding land parcels the Austrian legal system defines several legal instruments to enforce rights, the most important are the complaint of the not possessing owner for surrender of the parcel (*Eigentumsklage*) and the complaint of the owner of a parcel against a person who does not recognize his rights (*Eigentumsfreiheitklage*). For example, a person unauthorized uses a part of a parcel that another person owns (e.g., uses it as field). Under Austrian law the land registry (*Grundbuchamt*) is the responsible court. After the judgement becomes valid, and the unauthorided user does not abandon the land, the judgement execution would end the land use by the defendant (the defendant would be evicted) so that the owner can again use his parcel according to his own interests which is the situation guaranteed by his ownership right.

2.12 Assumptions about the law

This section describes the assumptions about the cadastral law we make in the remainder this thesis with the goal to focus on the essential structure of reality in a cadastre.

Focus on the land register. This thesis deals with the legal status of parcels. The legal status of parcels as well as change in the legal status is recorded in the land register. The records of the land register establish the title to land. In the following it is sufficient to focus on the land registry and its embedding into the whole legal system.

The detailed discussion of spatial issues will be omitted. Cadastral systems deal with the description and representation of parcels, their shape and boundaries, as well as with the legal status of parcels. This thesis is mainly interested in the legal status of land, i.e., who holds rights on pieces of land. It omits the discussion of spatial issues related to land and its representation in the cadastre, as the legal topics of this thesis can be described without detailed reference to spatial issues.

Focus on ownership and ownership transfer. This thesis investigates the principal structure of reality in a cadastre. For this purpose we focus on the the ownership of a parcel which is the central element in the cadastral law. The rules for other rights (restricted rights, such as mortgages) are quite similar. A further investigation of other rights would not contribute to the results of this thesis as it is not intended to give a complete description of the cadastral law.

The transfer of ownership of a parcel is the example of a legal transaction we will focus on in the following. The transfer of ownership is the major cause of legal change in a cadastre and thus an essential part of reality in a cadastre.

Focus on natural legal persons of legal age. For the discussion of change in the legal status of parcels we focus on natural persons of legal age because only they are capable of signing contracts, which is typically necessary to cause change in the legal situation of parcels. Exceptions from this rule exist, for instance, in the case of inheritance. In the remainder of this thesis we assume every legal person to be of legal age to discuss typical cases of right changes.

No discussion of defects of will and defects of contracts. We omit the discussion of specific legal issues, such as defects of will and defects of contracts. For a further discussion see (Krejci 1995, p.51) or (Bydlinski 1996)). We assume all contracts concluded are valid and represent the intentions of the participants.

2.13 Summary

The purpose of this chapter was to introduce the rules of the legal system relevant for the cadastre. We focused on a concrete system, namely the Austrian cadastral system.

The Austrian legal system is an instance of a title registration system. It cannot be completely described without the discussion of the embedding of the cadastral system into the legal system in general.

The objects of rights in a cadastre are parcels. The subjects of rights are legal persons. The main important right is the ownership right as the most comprehensive right one can have on a parcel. We focused on the discussion of the ownership right.

The main purpose of the legal system is to avoid the direct application of physical power between persons to enforce their rights. The alternative possibilities for legal persons to enforce their rights with the help of the power of the state regulate the living together of people. These possibilities are a central element determining reality in a cadastre. We introduced complaints and the execution process as the way to mobilize the physical power of the state.

Chapter 3

The ontology of institutional reality

3.1 Introduction

This thesis describes reality in a cadastre, i.e., the part of the real world which is determined by the activities of the cadastral registry, the cadastral law, and the people interacting with the cadastral registry according to the rules given by the law. We characterize this domain as part of social reality. This chapter introduces the philosophical foundation of this thesis, namely a theory about how the institutional part of social reality is structured and constructed.

Ontology is the description of what is, i.e., the study of reality and its structure. This chapter gives the background for the discussion of 'what is' reality in a cadastre, of the basic entities, rules and relations determining this domain. We discuss several interpretations of the term 'ontology' with the goal to introduce a definition applicable for this thesis. The chapter introduces an ontology of institutional reality mainly based on the work of the philosopher John Searle in his book *"The construction of social reality"* (Searle 1995).

The basic elements of Searle's theory are physical and institutional facts, status functions, collective intentionality, constitutive rules and conventional power assigned to institutional facts. Physical facts exist in physical reality, institutional facts exist in institutional reality. Institutional facts are characterized by the assignment of some status to physical phenomena by collective intentionality. Constitutive rules define the conditions for the assignment of these status functions. Collective intentionality means that the status is collectively recognized by the people in the domain in question. We simplify Searle's theory discussing the structure of institutional reality without collective intentionality involved in the construction of institutional facts. Conventional power is assigned to institutional facts, i.e., rights and duties. We will emphasize the discussion of conventional power and its relationship to the corresponding concepts from the legal domain and its relationships to physical activities.

Institutional reality does not exist independently of language. The chapter concludes with a discussion of the role of language in institutional reality.

3.2 Ontology and ontologies

Traditionally *ontology* is a branch of philosophy. Recently *ontologies* are used in computer science for the construction of information systems (Guarino 1998). This section discusses the different notions of ontology relevant for this thesis. We distinguish ontology in the philosophical sense and ontologies in the computer science sense.

Ontology is the metaphysical study of being and existence (Fellbaum 1998). In the philosophical sense ontology is the science of what is, the kinds and structures of the objects, the properties and relations in every area of reality. In simpler terms it provides a classification of entities (Smith to appear). An ontology in the philosophical sense is language independent, i.e., independent from the language it is expressed in.

Ontologies in the computer science sense are engineering artifacts, constituted by a specific vocabulary used to describe a certain domain and a set of explicit assumptions about the intended meaning of the vocabulary (Guarino 1998). Thus they are language dependent. Ontologies in the information science sense are constructed with a certain use and a specific computing environment in mind. The main purpose of ontologies in this sense is to provide information reusable for all parties in the given domain.

Ontology in the traditional, philosophical sense is not a science about how we conceptualize the world, it describes the world itself (Peuquet, Smith & Brogaard 1999). It assumes that there exists one world to be described. Ontology is distinguished from epistemology, which is the philosophical theory of knowledge (Fellbaum 1998). Ontology does not presuppose epistemology, it is completely independent of our knowledge of the world. Strictly speaking, if there is one world there must be one ontology. Human conceptualizations of this world are not ontologies in this sense (Frank to appear). Human conceptualizations can be described as e-ontologies (epistemological ontologies) whereas an r-ontology (reality based ontology) is an ontology in the traditional sense (Peuquet et al. 1999). Both e-ontologies and r-ontology are language independent, the first describes parts of the world based on human knowledge, the second describes what is independent of human knowledge. Figure 3.1 shows the different concepts of ontology.

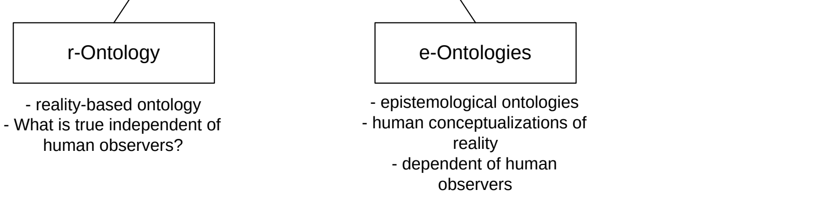
Ontology
Ontology in the philosophical sense
- language independent
Ontology in the computer science sense
-language dependent domain descriptions
r-Ontology
- reality-based ontology
- What is true independent of human observers?
e-Ontologies
- epistemological ontologies
- human conceptualizations of reality
- dependent of human observers

Figure 3.1: Categorization of ontology

According to the terminology used above we will first introduce, based on Searle's theory, an r-ontology of the structure of institutional reality, i.e., an ontology that is intended to show the objective structure of institutional reality. Based on this r-ontology about how institutional reality is constructed, we will develop an e-ontology of the cadastral domain. The r-ontology of institutional reality explains how institutional facts are constructed whereas the e-ontology of the cadastre discusses, which institutional facts actually exist.

Since ontologies in the computer science sense are specifications of conceptualizations (Guarino 1998) the computational model of this thesis can be regarded as the specification of the e-ontology of reality in a cadastre. The computational model itself can be seen as ontology in the computer science sense.

3.3 Five tiers of ontology

Reality can be conceptualized by distinguishing the following five tiers (Frank to appear) of ontology. Table 3.1 shows the five tiers of ontology.

On the physical level (tier 0) there exists one single physical reality. It is determined by properties at every point in space and time. Space and time are the fundamental dimensions of this reality. Observable reality (tier 1) assumes that the properties of tier 0 are observable at each point in space. These observations are incomplete, imprecise and approximate. The object world (tier 2) assumes that objects are constructed by human cognition based on observations of uniform properties of regions in space and time. Objects continue in time. Social reality (tier 3) is created by social rules. Social rules create facts which are valid within a social context only. Cognitive agents (tier 4) use their knowledge to make decisions. These decisions are acquired gradually and lag behind reality.

Frank's ontological tier 0 can be characterized as r-ontology describing what exists independent of human observers. All other levels correspond to e-ontologies focusing on different conceptualizations of reality.

Ontological Tier 0: Physical Reality:

- the existence of a single physical reality,
- determined properties for every point in time and space,
- space and time as fundamental dimensions of this reality.

Ontological Tier 1: Observable Reality:

- properties are observable now at a point in space,
- real observations are incomplete, imprecise and approximate.

Ontological Tier 2: Object World:

- objects are defined by uniform properties for regions in space and time,
- objects continue in time.

Ontological Tier 3: Social Reality:

- social processes construct external names,
- social rules create facts and relationships between them,
- social facts are valid within the social context only.

Ontological Tier 4: Cognitive Agents:

- agents use their individual knowledge to derive other facts and make decisions,
- knowledge is acquired gradually and lags behind reality.

Table 3.1: The five tiers of ontology

We present here an adapted version of the five tier ontology that shows the tiers 2 and 3 as subsets of the individual beliefs of the agents (tier 4). We assume that there exists a physical reality (tier 0). Human beings make observations from this reality (tier 1) and derive individual beliefs from these observations (tier 4). These beliefs form conceptualizations of physical reality in terms of objects (tier 2). Object world in this sense is the human conceptualization of what exists independent of human observers in physical reality. Applying social rules human beings derive individual knowledge about social reality (tier 3). This knowledge depends on our concepts of the physical world (tier 2) and represents human knowledge about phenomena which exist only in relation to human observers.

This adapted version of the five tiered ontology allows conceptualizing social reality in terms of individual beliefs and interaction of human beings, i.e., social facts exist only in the human mind. Figure 3.2 shows the adapted version of the five tiered ontology.

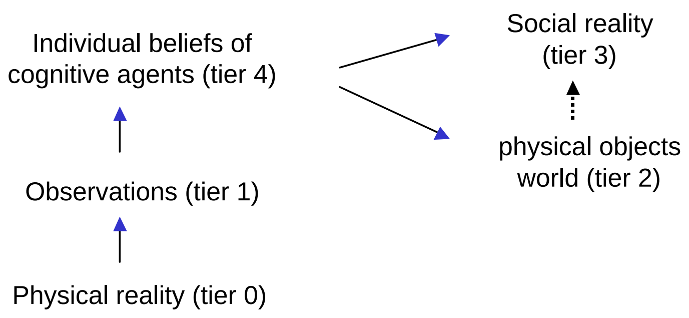
Individual beliefs of cognitive agents (tier 4)
Social reality (tier 3)
physical objects world (tier 2)
Observations (tier 1)
Physical reality (tier 0)

Figure 3.2: Five tiers of ontology (adapted version)

3.4 Searle's theory of institutional reality

This section gives a detailed description of Searle's theory of institutional reality, which is the major foundation for the view of reality we present in this chapter.

3.4.1 Facts

In his theory Searle talks about facts (physical and institutional), not about objects, events or relations between objects. In the following we will discuss the concept 'fact'.

Searle assumes that there exist facts in reality and statements in a language. Statements are attempts to describe things in the world existing independent from the statements. Statements can be true or false. A statement is true if it corresponds to the fact it states (Searle 1995, p.201) (this is called correspondence theory of truth). In this sense the fact defines the conditions that make a statement true. For instance, the statement "There is snow on Mount Everest" is true only if there is snow on Mount Everest. A fact is now anything in the world that makes a statement in a language true. Sometimes the term 'fact' is replaced by 'situation' or 'state of affairs'. Facts are sets of objects and their relationships. Figure 3.3 shows the correspondence between facts in reality and statements in a language.

Fact is a very broad notion that can stand for any entity in external reality or any systematic relation between entities. Facts can concern objects, for instance, the

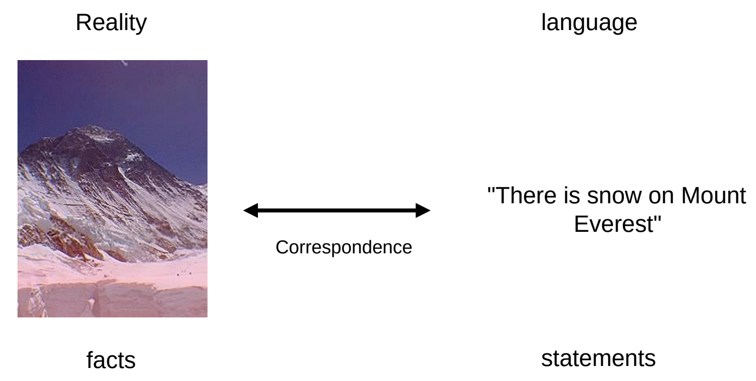

Figure 3.3: Facts and reality vs. statements and language

fact stated by the statement "This is a car." identifies an object. The term 'fact' is a more general notion then, for instance, the terms 'object' or 'event'. If the statement identifies a concept, such as an object, event or relations between objects and events, we can identify the fact with the concept in question, i.e., we can talk about objects, events and relations.

3.4.2 Physical and institutional reality

Searle distinguishes a physical and an institutional level of reality. Within physical reality physical facts exist and within institutional reality institutional facts exist. Physical facts exist independent of human observers in external reality, for instance, there exist cars, snow on Mount Everest, pieces of land or pieces of paper. Institutional facts are observer relative and exist only in the human mind, such as money, the owner of a parcel or a driving license.

Institutional reality is based on physical reality. Institutional facts have always a physical foundation, for instance, in the case of money, there are pieces of paper or data stored in a database (realized in the physical structure, for instance, of a hard disk). In a similar way any form of language needs some physical realization, such as air waves, paper or data structures in a computer system. Physical reality is prior to institutional reality. Its structure and properties determine institutional reality. Institutional facts need physical facts for their creation and existence.

Institutional reality is context dependent (Frank to appear). Institutional facts exist only in particular contexts, for instance, "ownership" depends on a particular legal context and dollar bills are only money in the context of the American monetary system. Institutional facts exist only in the context of specific institutions.

3.4.3 Status functions

Human beings do not experience the world as material objects or as collection of molecules. Human beings experience chairs, tables or cars, i.e., we experience the world by assigning functions (Searle 1995, p.15). If we experience a chair, then we experience something where we can sit on, in the case of a car, something we can drive with.

Functions, humans assign to phenomena in external reality, are not intrinsic to the physical properties of the phenomena. They are assigned from the outside by conscious observers, i.e., they are observer relative. The function chair, for instance, does not exist without a human being that can sit on it. The physical properties of a chair say nothing about 'sitting on it'. This is a function that only exists relative to human observers that can sit on the physical object.

A similar idea was developed in the field of ecological psychology by J.Gibson (Gibson 1979). He introduces the concept of "Affordances", which are what an object, an assembly of objects, or an environment offers people to do. While originally strongly related to human perception, i.e., to physical affordances, the concept was recently extended to include social-institutional affordances (Raubal 2001).

A specific class of functions are status functions (Searle 1995, p.41). They cannot be performed only by the physical features of the phenomenon. A chair, for instance, can perform its function by its physical properties. It has the form, the material and the stability that allow people to sit on it. In opposition the function 'money' cannot be performed only based on the physical properties of a piece of paper. The piece of paper does not have itself a value that allows functioning for the exchange of value. Status functions are functions that are beyond the physical properties of the phenomena. A piece of paper functions as medium of value exchange, because the status 'money' of the paper is recognized by the society.

3.4.4 Collective intentionality

Status functions are assigned to phenomena by collective intentionality. Searle defines collective intentionality as the human capability to collectively share intentional states, such as beliefs and intentions (Searle 1995, p.23). In particular this comprises the capability to assign status functions.

Collective intentionality (such as 'We believe') must be distinguished from singular intentionality (such as 'I believe'). Collective intentionality cannot be reduced to singular intentionality (for instance, to the form: 'We believe = I believe that you believe that I believe that...'). Searle assumes that collective intentionality is a biologically primitive phenomenon that cannot be reduced to anything else (Searle 1995, p.24). Singular intentionality is derived from collective intentionality a human being shares with others. Collective intentionality is a capability Searle presupposes for his theory (Smith & Searle 2001).

Searle defines all facts involving collective intentionality as social facts (Searle 1995, p.26). Collective intentionality is necessary for the creation of institutional facts but is not limited to. According to the definition, social facts form a much more comprehensive class, which includes institutional facts as a subclass.

3.4.5 Constitutive rules

Constitutive rules define the conditions under which a status function is assigned to a phenomenon, i.e., the conditions for the creation of the institutional fact. For example: Which conditions create the institutional fact 'money'? What are the conditions that cause that we collectively accept pieces of paper as money? Constitutive rules play this role.

Two kinds of rules exists: regulative rules and constitutive rules (Searle 1995, p.27). Regulative rules define the rules for existing activities, for instance, the rule 'drive on the right hand side of the street' regulates an already existing activity: car driving. In opposition constitutive rules create the possibility of activities. For instance, the rules

of chess playing create an new activity: chess playing, which did not exist before the rules were defined. Constitutive rules define under which conditions status functions are assigned to phenomena. Institutional facts only exist within systems of constitutive rules. The system of rules creates the possibility of institutional facts.
Constitutive rules have the following characteristic form:

X counts as Y in C

X is a physical fact to which a status function should be assigned. Y is an institutional fact after assigning the new status function. C is a context in which the assignment of function occurs. According to the example of money, the X fact is a piece of paper, the Y fact is money, i.e., a piece of paper with the status function 'medium of value exchange' assigned. The context in this case would be, for instance, the legal system of the USA if the piece of paper is a dollar bill. The constitutive rule for money is then 'This particular piece of paper counts as money in the USA'.

Constitutive rules can form hierarchies. Then the X term can be an institutional fact on a lower level of institutional reality that is involved in the creation of the institutional fact Y on a higher level of institutional reality. For example, in the rule 'The citizen A of the US counts as president of the US in some particular context C' the X term, 'The citizen A of the US' is itself an institutional fact.

Since social reality is context dependent (Frank to appear), the definition of the context in the rules plays an important role. Contexts can be specific, such as, 'chess playing' in which moving of a piece of wood in a specific situation counts as beating checkmate, or general, such as the legal system, in which particular activities count as ownership transfer. Contexts can be hierarchically nested. For instance, a particular activity (handing over of a piece of paper) performed at the cadastral registry (the specific context) counts as application (which is only meaningful in the context of the cadastral law embedded into the legal system in general).

3.4.6 The creation of institutional facts

For the creation of institutional facts three elements are necessary:

- status functions
- collective intentionality
- constitutive rules.

The three elements together allow the construction of institutional facts: Collective intentionality assigns a new status to a phenomenon. The status has a function which cannot be performed only by the physical features of the phenomenon. The assignment creates a new fact, an institutional fact, a fact created by human agreement (Searle 1995, p.46). The assignment of the status occurs according to the constitutive rules for the fact. The constitutive rules define the conditions for the creation and destruction of institutional facts. Institutional facts are characterized by the assignment of conventional power, i.e., rights to perform activities connected to the institutional fact. Figure 3.4 illustrates the basic ideas of Searle's theory. Figure 3.5 shows the elements involved in the construction of the institutional fact 'money'.

In the simplest case institutional facts are created by assigning a status function to a physical phenomenon. In this case, according to the rule *X counts as Y in C*,

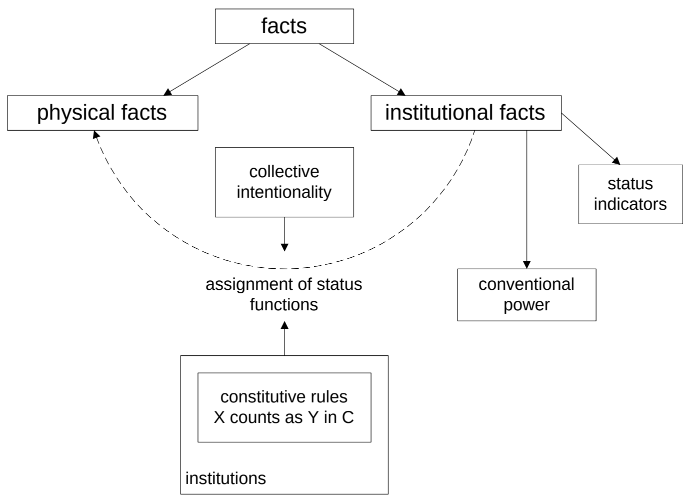
facts
physical facts
institutional facts
collective intentionality
status indicators
assignment of status functions
conventional power
constitutive rules X counts as Y in C
institutions

Figure 3.4: Searle's theory of institutional reality

the X term is a physical fact and the Y term is an institutional fact. If the rules are hierarchically nested, the X term can be itself an institutional fact. On the bottom of a hierarchy of constitutive rules must be always a fact which is not a matter of human agreement, i.e., a physical fact. According to the example of presidency (see subsection 3.4.5), on the bottom of the hierarchy defining the rules for the creation of the fact 'president of the US' must be a physical fact, i.e., a human being to which the status is assigned.

3.4.7 Status indicators

Institutional facts are not derivable from the physical phenomenon to which the status function is assigned. Their existence presupposes that people have knowledge about these facts. Institutional facts require representations, i.e., physical objects which make visible the status assigned. Such representations are called status indicators. Status indicators are, for instance, weeding rings, written contracts or uniforms. Wedding rings (physical objects made of metal) represent the institutional fact that the person who carries the ring is a married adult. A written contract (a piece of paper) represents the event of concluding a contract. A uniform has signs that show the position and status the person holds, for instance, in an army.

Important is that status indicators do not create institutional facts. Their function is only epistemic, i.e they transport knowledge about an assigned status.

3.4.8 Institutions

Institutional facts exists within institutions (Searle 1995, p.113). Institutions permit the creation and destruction of institutional facts. Institutions consist of constitutive rules of the form X counts as Y in C and specifications of the rights which are connected to institutional facts. Institutions are just sets of constitutive rules and sets of rules defining rights. These rules define the institutional facts possible within the institution. Institutions are a possibility to group constitutive rules existing in a community. Strictly speaking institutions are just names for sets of rules.

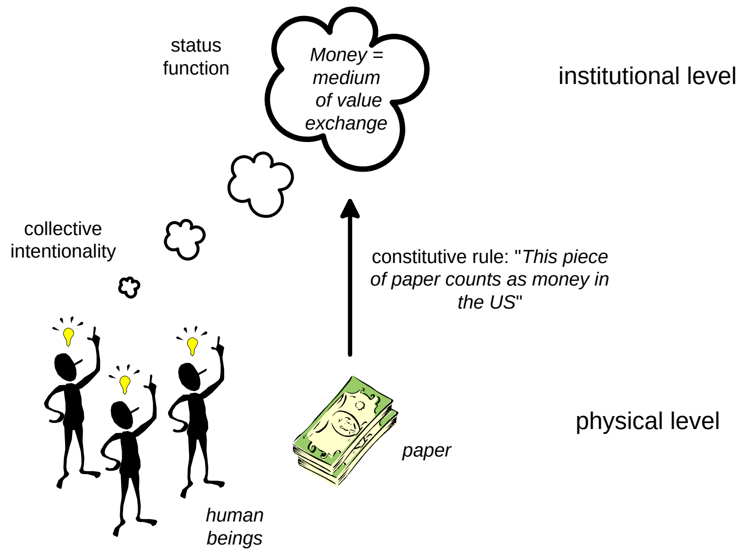
status function
Money = medium of value exchange
institutional level
collective intentionality
constitutive rule: "This piece of paper counts as money in the US"
physical level
paper
human beings

Figure 3.5: The construction of institutional facts according to Searle: the example of money

3.4.9 Reality seen through Searle's theory

After the description of the main content of Searle's theory we now can discuss the properties of institutional reality as seen through his theory. Reality consists of two parts, the physical part, existing independent of human beings and the institutional part, existing by human agreement. Physical reality consists of physical phenomena, such as objects and events, which stand in systematic relationships (Searle refers to all situations constructed out of physical phenomena as physical facts). Searle presupposes that there exists a specific class of these objects capable of collective intentionality and with the capability to symbolize (Smith & Searle 2001), i.e., to create language. These objects are human beings. Human beings assign status to phenomena. The conditions for this assignment can be described by constitutive rules. Human beings are capable of collectively sharing beliefs about assigned status functions. Status functions assigned to physical phenomena, which are collectively recognized by the majority of a community, Searle refers to as institutional facts. Conventional power is connected to collective accepted status functions, i.e., the possibility to perform some activities to which again status functions are assigned.

To summarize: There are human beings and other physical phenomena in external physical reality. Institutional reality exists only in the mental state of these human beings, i.e., institutional reality exists only as collectively shared beliefs in the mind of human beings.

3.5 The role of collective intentionality

In Searle's theory collective intentionality plays an important role. This section shows how institutional facts can be constructed without collective intentionality involved.

3.5.1 Preconditions for the imposition of collective intentionality

Collective intentionality means that human beings are able to share particular intentional states, such as beliefs and desires (see subsection 3.4.4). In the context of institutional reality this means that people collectively accept and recognize the status function assigned. In order to establish collective intentionality, four elements must be involved:

1. Everybody must have knowledge about the institution.
2. Everybody must accept the institution.
3. Everybody must have knowledge about the status function assigned (within the institution).
4. Everybody must accept the status function assigned.

In modern society (and in particular in a cadastre) it is impossible to establish institutional facts by collective intentionality, i.e., by everybody knowing and accepting the fact. Modern society is too complex. Nobody can gain enough knowledge necessary to construct the complex structure of institutional facts, which nevertheless exists. Institutional facts must be constructed in a different way without collective intentionality involved.

3.5.2 The monopoly of force

In modern, complex social reality collective intentionality is replaced by the monopoly of force, i.e., by the fact that particular institutions have the power to create and maintain the institutional structure of the society. The monopoly of force has three aspects: the monopoly to assign status, the monopoly to define rules and the monopoly of violence.

The monopoly to assign status

In modern society the authority to create, maintain and destroy institutional facts is given to particular organizations. For instance, conventional power is assigned to the cadastral registry to transfer ownership. In a similar way other organizations of the society have the power to create money. We further assume that some organizations have complete knowledge about the status functions assigned. For instance, in the field of cadastre institutional facts are created, first, by our knowledge about the institution (e.g., ownership), second, by our knowledge about the responsible organization (for instance, the cadastral registry) which we can use to acquire the necessary knowledge about the institutional status of particular objects (e.g., human beings with the status owner, land pieces with the status parcel). Third, institutional facts are created, because organizations have the conventional power to assign and destroy status functions. The monopoly to assign status can be characterized as the monopoly to apply the institutional rules.

The monopoly to define institutional rules

In modern society institutions are not created, because we collectively accept the rules, the institution consists of. This is replaced by the authority of some organizations of the state to define rules and to alter the institutions. The constitutional law, for instance, defines the general rules of the living together of people in a country. The constitution can be altered by particular decisions of the government of the concerning country. The definition of the institutional rules defines the context in which institutional reality exists. In the legal domain the most general context is the legal system.

The monopoly of violence

Searle claims that institutional facts acquire conventional power, because they are collectively accepted. In modern society institutional facts acquire conventional power because the rules of the institutions are enforced by the state. The humans in the society recognize the authority of the state to use brute force according to the legal rules. The monopoly exists as long as most people of the society accept the system of institutions, regardless of single individuals violating it. Similarly the authority of courts to decide in the case of conflicts between people is constructed. The monopoly of violence is the monopoly to physically enforce institutional rules.

3.5.3 Collective intentionality and institutions

Collective intentionality is not involved in the assignment of status functions. The several aspects of the monopoly of force replace collective intentionality in the construction of institutional facts. Collective intentionality is related to institutions. The institutional structure, i.e., the system of institutions comprising, for instance, the institution of the monopoly of force, the institution of ownership or the institution of money exist by collective intentionality, i.e., by collective acceptance. The legal system as the most general institution, comprising the other institutions, exists, because the majority of the citizens accept the institution. This corresponds to the view of the state as a social contract between its citizens, between the ruler and the ruled, as introduced by Hobbes, Locke and Rousseau (Britannica.com 2001)[1].

3.5.4 Collective intentionality as social phenomenon

Searle claims that collective intentionality is a biologically primitive phenomenon, which cannot be explained in terms of anything else (see subsection 3.4.4). We explain collective intentionality as a social phenomenon based on the economic interests of the individuals. Institutions exist because they improve the economic exchange between human beings (North 1997). The individuals accept and recognize the institutions as long as they are economically advantageous for them. As long as the institutions provide benefits to the majority of the individuals in a society they will be accepted by the individuals. Collective intentionality is the result of the individual acceptance of the institutions by the individuals.

[1]http://www.britannica.com/eb/article?eu=70216&tocid=0

3.6 Conventional power: Rights

This section discusses the effects of the assignment of status functions in reality. The assignment of status functions creates conventional power, it assigns rights and obligations related to activities.

These activities are not necessarily of physical nature. Institutional facts can create the possibility of institutional activities. Institutional facts create the possibility of activities that cause the imposition or destruction of status functions. The institutional fact 'money' creates the possibility of paying. The institutional fact 'ownership of an object' creates the possibility to transfer ownership. Institutional facts do not create the possibility of physical activities: It is impossible by defining an institutional fact to create for human beings the physical capability of flying.

Conventional power can be assigned to institutional activities (such as: paying) or physical activities (such as: evicting). Nevertheless conventional power to perform activities is always based on physical activities. The activity 'paying', for instance, is physically based on the handing over of pieces of paper. Ownership transfer can be based on the handing over of the object.

As this thesis is interested in the cadastral domain which is determined by legal concepts, we will relate Searle's concept of conventional power to the concept of a right from the philosophy of law (Kanger & Kanger 1966, Kanger 1981) and to the usage of the term right from in the positive law (see chapter 2).

3.6.1 Searle's concept of conventional power

Conventional power is connected to institutional facts. The creation of institutional facts creates conventional power and the destruction of institutional facts destroys conventional power. Conventional power can be directly assigned to a subject, as in the case of the owner of an object, or indirect, where the status is assigned to an object, e.g., dollar bills, as it is in the case of money.

Conventional power is always the power to do something or to constrain someone else from doing something (Searle 1995, p.104). Here we introduce for clarification *active conventional power*, the first case, and *passive conventional power*, the second case.

Searle distinguishes *positive and negative conventional power*, i.e., *enablements and requirements* (Searle 1995, p.104). In the case of enablements conventional power grants power to a subject. For instance, the conventional power to buy objects up to a value of one dollar is granted to the possessor of a dollar bill (strictly speaking, it is the power to pay debts of one dollar). In the case of requirements the conventional power of subjects is restricted. For instance, a park ticket creates the requirement for a subject to pay an amount of money.

Combining the concepts of active and passive conventional powers with the concept of positive and negative powers we can discuss four modes of conventional power.

1. *Active, positive conventional power.* This means the possibility of a subject to perform an activity. For instance, the owner of a parcel has the power to sell his parcel.

2. *Passive, positive conventional power,* i.e., the possibility of a person not to perform a particular activity. For instance the citizen of a country has the passive

conventional power to have privacy, i.e., he has the power not to give all private information about himself.

3. *Active, negative conventional power,* i.e., the requirement for a subject to perform a particular activity. For instance, the citizen of a country is required to pay taxes.

4. *Passive, negative conventional power,* i.e., the requirement not to perform a particular activity. For instance, the seller of a parcel is required not to sell the parcel twice.

3.6.2 Conventional power and physical capabilities

Searle does not explicitly discuss physical powers[2]. He does not distinguish between physical powers and rights. When he talks about powers, he means rights (Smith & Zaibert to appear). We assume that there must be a physical power or a physical capability of a human being as foundation for the creation of rights. We distinguish between physical capabilities and (institutional) rights.

Physical capabilities and rights cannot be defined as subclasses of each other. There are situations where a person can have a right without physical capability or the physical capability without a right (Zaibert 1999). For instance, one can have the right to use a piece of land without having the capability (because of sickness or age). Human beings often have capabilities without rights, e.g., one has the capability trespass a parcel another person owns, without the right to act in this way. People have the capability to injure other humans without the right to do so.

We do not draw the conclusion that physical capabilities and rights are independent of each other. We assume that rights are only meaningful (and consequently exist) if, in principle, there exists the possibility to act in a way that the right prescribes, e.g., that there are human beings with the physical capability to act in the described way. There exist no rules defining or restricting the right to swim over the Atlantic ocean, because no human being has the capability to show such kind of behaviour. We assume that rights are always based on physical capabilities. Powers (physical capabilities) are the most primitive relationship between human beings and land (Smith & Zaibert to appear). All rights people can have on land parcels are based on these human capabilities regarding land.

3.6.3 The concept of a right

In the philosophy of law the concept of conventional power corresponds to the concept of a right. In general a right is a relation between two parties (subjects or organizations) X and Y which concerns a given state of affairs S between X and Y (Kanger & Kanger 1966). It is a particular relation and often it is used in the way that one party's claim corresponds to another party's duty (Kanger 1981). In general there are four basic types of rights (Kanger & Kanger 1966): *claim, immunity, power*[3] *and freedom.* We will introduce these four basic types and give their interpretations (Kanger 1981):

[2]Here we use the term 'power' synonymous with physical capabilities and the term 'right' for conventional power (in Searle's sense). Both terms must be distinguished from 'power in a legal sense' which designates a particular type of a right (see section 3.6.3).

[3]To be distinguished from Searle's usage of the term 'conventional power'. If there is the danger of confusion we refer to 'power in Searle's sense' and 'powers in the legal sense'

Rights (of X against Y concerning state of affairs S)	positive law
claim	duty of some Y to establish S
power	duty of all Y not to establish S
immunity	duty of all Y not to destroy S
freedom	no claims of other Y concerning S

Table 3.2: Rights and the positive law

X has against Y a *claim* with respect to the state of affairs S(X,Y) if Y has against X a duty with respect to S. This means that it shall be the case that Y sees to it that S is the case. Y has against X the duty to perform an activity that establishes the state of affairs S.

X has against Y an *immunity* with respect to S means that it shall be the case that Y does not see to it that not S. Y should not perform an activity that destroys the state of affairs S.

X has against Y the *power* with respect to S means that it may be the case that X sees to it that S. X has the possibility to perform an activity that establishes S.

X has against Y the *freedom* with respect to S means that it may be the case that X does not see to it that not S. This means that X has not the duty against Y to establish the state of affairs not S. X does not have the duty to abandon S.

3.6.4 Relating the types of rights to the positive law

The positive law is the system of law implemented and laid down within a particular political community by political superiors (Garner 1996, p.488). The positive law is the concrete realization in a particular legal system. We relate the concept of a right from the positive law (see chapter 2) to the types of rights introduced above.

The concept of a right in the positive law is characterized by the correspondence of rights and duties. For every right another person has a corresponding duty (see section 2.8). Personal (relative) and real (absolute) rights can be explained in terms of other persons duties.

Personal rights find their correspondence in the type claim. A claim defines a duty for another person. Claims and duties are most important for the positive law because they regulate concrete obligations between parties. The law is mainly concerned with the question how to avoid and resolve conflicts between persons regarding these obligations.

Real rights correspond to the types of rights immunity, freedom and power. They can be defined by claims and duties a subject has against all other subjects of the community. The type immunity of X with respect to a state of affairs S can be defined in terms of the duty of all other subjects Y not to destroy S. The freedom of X against Y concerning S is the absence of duties for X, i.e., can be described as the absence of any claims of other subjects Y against X with respect to S. The power of X according to S is the duty of all other persons Y not to establish the state of affairs S. Power defines the possibility to establish S, which is equal to the duty of all others not to block this possibility. Table 3.6.4 shows the relationships between the types of rights and the concept of a right used in the positive law.

Conventional power	Rights
active, positive	power
active, negative	duty
passive, positive	freedom
passive, negative	immunity of all others

Table 3.3: Conventional power and rights

3.6.5 Relating conventional power to rights

The phrase 'X sees to it that S' is a synonym for the possibility of an activity. 'X sees to it that S' means that X performs an activity to establish the state of affairs S. This allows relating conventional power (to perform activities) to rights. Table 3.6.5 shows the relationships between conventional power and rights.

Active positive conventional powers are related to powers in the legal sense. They define the possibility for a subject to perform a particular activity that establishes a particular state of affairs.

Passive positive conventional powers are related to freedom. They define the possibility not to perform a particular activity, i.e., the right to maintain a particular state of affairs.

Active negative conventional powers are related to duties (and the corresponding claim another person has). They define the obligation for a subject to perform an activity that establishes a particular state of affairs.

Passive negative conventional power relates to immunity. Passive negative conventional power of a subject X concerning an activity A is the immunity of all other subjects Y against X with respect to a state of affairs be established by performing the activity A. Passive conventional power is equivalent to the absence of any power of X with respect to the state of affairs S.

3.7 The building blocks of institutional reality

From the analysis so far we are able to derive the building blocks of institutional reality. According to subsection 3.4.9 the ontology of institutional reality consists of physical facts corresponding to physical phenomena and institutional facts corresponding to beliefs about status functions that are assigned to phenomena. We distinguish therefore: (physical) *phenomena* and *status* assigned, which are first two building blocks of institutional reality.

The physical level of phenomena and the institutional level of status are connected by *constitutive rules.* Constitutive rules define the conditions for the assignment of status. Constitutive rules are the key element unifying the physical and the institutional level of reality. They are the third building block of institutional reality.

For the description of reality, change plays a crucial role. Change in reality occurs by events. Events are caused by human activities. *Physical capabilities* define the possible activities for human beings on the physical level. *Rights* define possible human activities on the institutional level according to the status assigned to phenomena (see subsection 3.6.2). Rights are based on physical capabilities of human beings, i.e., there are no rights independent of physical capabilities. To explain change, physical capabilities and rights of the human beings are essential aspects of institutional reality.

The building blocks necessary to explain the structure of institutional reality are the following:

- (physical) phenomena
- (institutional) status
- constitutive rules
- (institutional) rights (connected to status) of human beings to perform activities
- (physical) capabilities of human beings to perform activities

This set of building blocks represents an *external view* of reality. The external view characterizes the fact that it regards every component as independently existing in reality. Physical phenomena are represented on the same level as status. In reality there is a qualitative difference between both. Physical phenomena exist in external reality whereas status as well as constitutive rules and rights exist only in the human mind. To achieve a closer correspondence to reality we have to respect this difference. This leads to an *individual-based (internal) view* of reality . With respect to the internal view the building blocks of institutional reality are (see figure 3.6):

- (physical) phenomena
- beliefs in the humans minds (about constitutive rules, status and rights)
- (physical) capabilities of human beings

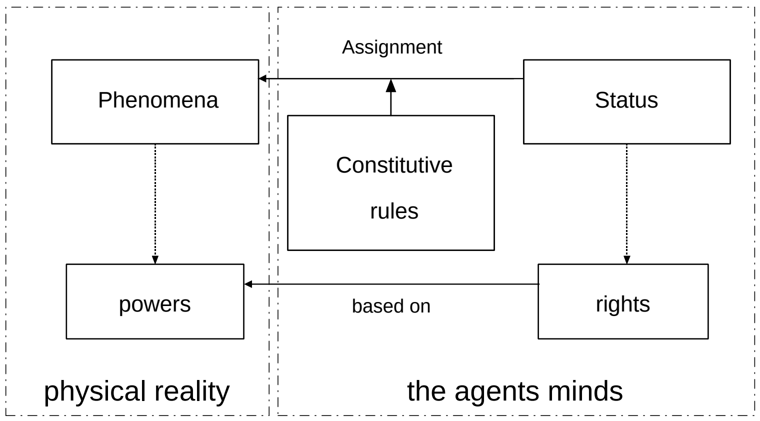

Figure 3.6: The building blocks of institutional reality

Our ontology fits to the adapted version of the five tiered ontology (see figure 3.2). The usage of the term 'phenomenon' corresponds to the usage of the term 'object' within the five tiered ontology. Our physical level is related to tier 2, the object world. Object world presupposes collective human cognition which forms objects from uniform properties of space and time. We conceptualize physical reality based on objects, or, in our terms, based on different categories of phenomena.

According to Frank the object world is not an r-ontology, because human cognition is involved in the object construction. We do not follow this argumentation since the central question is not how we conceptualize some aspects of the real world, the central question is: Do the phenomena we conceptualize in some way exist independent of our conceptualization? For objects, such as, a bottle of water or a piece of land the answer is yes, the phenomena we designate as bottle or piece of land exist in reality independent of us. Therefore we regard our ontology of physical phenomena and physical capabilities of human beings (to act in their environment, to assign some status to phenomena, to define rules, to maintain representations about their environment) as an r-ontology.

The institutional level of reality corresponds to a subset of tier 3. Ownership, for instance, does not exist independent of human conceptualization.

By the transition from the external to the individual-based view of reality we can explain institutional reality (tier 3) in terms of the individual beliefs of the human beings (tier 4) constructed by the interaction of the individuals. In particular there exist no collective beliefs, human beings can have different beliefs about the institutional structure. Institutional reality is a property of the whole system, not of the individual human beings.

3.8 The role of language

Language assigns meaning to words people utter, write or in another way realize physically. Meaning is the capability of an object to represent something else, i.e., it is a status function and its function is to represent something. Maps, for instance, can represent a part of the surface of the earth. Language plays an important role for the structure of social reality, for Searle it is one of the capabilities he presupposes for his theory (Smith & Searle 2001). We distinguish the constitutive and the epistemic function of language.

Language is necessary for the construction of all institutional facts. This is the constitutive function of language. In order to have institutional facts at all, a society must have a form of language. Language is therefore prior to other institutions (Searle 1995, p.60). Language is itself an institutional fact, the function of language is to represent meaning, i.e., to represent or symbolize something beyond itself. Language, i.e., speech acts do not need other institutional facts for their creation. Every other kind of institutional fact needs language for its creation.

Language is not only necessary for the creation of institutional facts. Language has an important epistemic function. It is very often a status indicator symbolically representing institutional facts. In order to recognize institutional facts there must be a symbolic representation, because the status function is not derivable from the physical properties of phenomena. In order to recognize a piece of paper as dollar bill there must be a symbolic representation saying that the value is one dollar.

3.8.1 Speech acts

Language can be described by speech acts. The concept of a speech act was mainly developed by John Austin (Austin 1962) and John Searle (Searle 1969). The basic idea of speech act theory is that utterances made by human beings in everyday life situations essentially are actions analogous to physical actions. Speech acts are usually performed by a speaker with some intention. Utterances do not change the physical properties of objects in the same way as physical actions (e.g., dropping a stone) do. They affect the mental state of other human beings (their beliefs and desires) and thus

change the state of the world in a way only similar to physical actions. This subsection is based on the introduction to speech acts given by (Wooldridge 1992).
According to Austin by uttering a sentence a human being performs three types of activities:

1. *Locutionary acts.* The locutionary act is performed by (physically) uttering words.

2. *Illocutionary acts.* Illocutionary acts transport some illocutionary force, e.g., they inform about some state of affair or they intend to cause the hearer to perform a particular activity. They transport the force to change the mental state of another human being. Illocutionary acts are usually performed with performative verbs, such as 'inform', 'request' or 'demand'.

3. *Perlocutionary acts.* Perlocutionary acts indicate the effect of the speech act, e.g., requesting something (the illocutionary act) makes someone do something.

These three acts do not exist independently of each other, they are performed simultaneously. By uttering a sentence an illocutionary force is transported and a possible effect is achieved. Only together the three acts form a complete speech act. They are based on each other: an effect of a communication action needs the physical utterance of words which transports some illocutionary force.

Searle formulated necessary and sufficient conditions for the successful completion of speech acts, i.e the conditions for a speech act to have the desired effect. If the speaker utters a sentence, which is a request for the hearer to perform a particular action, such conditions are, for instance, that the hearer is able to hear and decode the sentence or that the hearer is able to perform the desired action.

Searle observed that there are performative verbs with similar illocutionary force. He gives the following characterization for speech acts according to their illocutionary force.

1. *Representatives.* Representative acts commit the speaker to something is being the case, i.e to the truth of the expressed proposition (for example, 'I assert...').

2. *Directives.* A directive is an attempt by a speaker to get the hearer to do something (e.g., requests: 'Please give me the...').

3. *Commissives.* Commissive acts commit the speaker to some future course of action (e.g., 'I promise to return.').

4. *Expressives.* An expressive act expresses some psychological state (e.g., 'I thank you for...', 'I am sorry to hear that.').

5. *Declarations.* Declarations effect some change in an institutional state of affairs (e.g., by performing the act of appointing a person as a chairman, then the person is the chairman). Declarations create institutional facts.

Often the term 'speech act' is synonymously used to illocutionary acts.

Speech acts can be analyzed according to the physical and institutional level of reality. The locutionary act is the physical action of, for instance, pronouncing words (causing waves expanding in the air) or writing on a piece of paper (putting ink on paper). Meaning is a class of status functions assigned to perlocutionary acts. If a speech act has declarative illocutionary force (e.g., 'I declare you husband and wife') the perlocutionary act creates new institutional facts (e.g., 'husband' and 'wife').

3.9 Summary

In this chapter we introduced the philosophical foundations of this thesis. The foundation of this thesis can be characterized as *ontology of institutional reality*.

The ontology of institutional reality of this chapter is mainly based on Searle's theory of institutional reality. Searle's theory describes reality as consisting of a physical part and an institutional part. Institutional facts existing in institutional reality are characterized by status functions. These status functions are assigned to physical facts. This assignment works according to constitutive rules of the form X counts as Y in C. Status indicators make institutional facts apparent and visible. Language plays an important role for the construction of institutional reality and can be described by speech acts.

Searle assumes that collective intentionality is involved in the assignment of status functions, i.e., institutional facts exist by human agreement or acceptance. We replaced collective intentionality by the monopoly of force, which gives particular organizations of the society the authority to create and change institutional facts. Thus collective intentionality is not involved in the creation of institutional facts. Collective intentionality is only involved in the creation of institutions, which exist by collective acceptance.

We simplified Searle's theory and assume that collective intentionality can be explained without presupposing a biological capability. Collective intentionality can be explained in terms of the single intentionality combined with a general economic principle. Collective acceptance of institutions is created if there is some economic benefit for the individuals, which causes the acceptance of the rules by the majority of the human beings.

Constitutive rules can be codified and are often codified in particular in laws. Searle's theory gives an approach to the analysis of legal domains, to the law itself and to reality determined by laws.

Legal domains are strongly influenced by rights. We emphasized the discussion of rights based on research done in the philosophy of law. We related the concept of a right from the legal domain to the concept of conventional power form Searle's theory. We found many similarities which allow the investigation of (legal) rights in terms of Searle's theory. Rights are strongly related to physical capabilities of human beings. They only exist if there is the physical possibility for human beings to act in a way, which is regulated by the rights.

Chapter 4

The analysis of reality in a cadastre

4.1 Introduction

This chapter analyzes reality in a cadastre based on the discussion of the legal domain (chapter 2) and of the ontology of institutional reality (chapter 3). The analysis captures the part of the real world necessary to understand the interaction of the cadastral registry with its environment.

The ontology of reality in a cadastre is a conceptualization of the part of reality in question, i.e., according the the characterization of ontologies given in section 3.2, it is an epistemological ontology. It represents a part of reality based on human knowledge. Reality in a cadastre cannot be described by an r-ontology (reality-based ontology) because as part of institutional reality it depends necessarily on human knowledge. Nevertheless it will be developed based on the ontology of institutional reality (chapter 3), which gives an account of the objective nature of how institutional facts are created, but not which institutional facts actually exist.

The ontology is intended to be independent of concrete national systems. However we discuss social processes from the Austrian legal system (see chapter 2). This is not a limitation of the approach. It is used to achieve a more realistic discussion based on a concrete system and it is intended as foundation for the computational model based on the Austrian system we develop in chapter 7.

The starting point is the idea that it must be possible to construct ontologies for certain legal domains, independent of the specific national legislation. The different legislations resolve the same problems, thus the ontologies should be likely similar (Frank 1997*b*). It should be possible to elicit the rational core of an ontology of real estate that is independent of the rule sets and practices of a specific country (Stubkjaer 2000).

The chapter starts with an explanation of what we understand by the phrase 'reality in a cadastre'. Next we justify our assumption that reality in a cadastre is a part of institutional reality.

Smith and Zaibert identify the following building blocks of the ontology of a cadastre (Smith & Zaibert to appear):

- sorts of things which can be objects of rights (immovable things),
- the issue of registration,
- different kinds of rights.

We discuss the objects of rights by investigating the relationship between parcels and land pieces. The issue of registration we discuss in connection to the creation of institutional facts and their representation. Then follows the description of the specific properties of rights in a cadastre.

The main part of the chapter introduces the ontology of reality in a cadastre based on three essential aspects and their interrelationships. These aspects we designate as (Bittner, Wolff & Frank 2000):

- Ontological categories of phenomena.
- Levels of reality in a cadastre.
- The distinction between facts and rules for their creation and existence.

The ontology is based on the connection of these three aspects which cannot be reduced in terms of each other.

The chapter concludes with a detailed discussion of the parts of the ontology in order to provide the necessary foundation for the computational model of reality in a cadastre.

4.2 What is reality in a cadastre?

Real estate is a complex historical product of interaction between human beings, legal and economic institutions, and the environment (Smith & Zaibert to appear). Usually models of the cadastre describe the cadastral registry as database and discuss its internal formal rules (see, for instance, (Frank 1996, Navratil 2001)). By investigating the cadastral registry and its input and output operations, it is not possible to discuss all relevant aspects influencing the work of the cadastral registry.

For example, in the Austrian and German law exists the concept of 'incorrectness of a cadastre', which is the non-correspondence between the content of the cadastral registry and the legal situation in reality. The maintenance of the correctness is not achievable in general because the cadastral registry is not a complete representation of the situation in reality (Bittner 1998). We made the discussion of the issue of correctness possible by adding a model of legal reality to the model of the cadastral registry, which allows the investigation of the differences between both parts of the model (Bittner & Frank to appear).

By reality in a cadastre we understand the part of the real world, which is influenced by the activities and the content of the cadastral registry. It comprises the information system cadastral registry fully embedded into its environment. The environment consists of human beings and organizations, and their interaction with each other and with objects, mainly land parcels, i.e., real estate. Persons act according to or violating the rules of the legal system. Reality in a cadastre is characterized by the activities of human beings and organizations where the cadastral registry is only one active entity influencing the state of the real world. Reality in a cadastre is influenced by the activities of autonomous individuals of the society embedded into the legal system of a country. This comprises the interaction with the cadastral registry, as well as with courts in the case of conflicts. An important factor is the physical force of the state a person can mobilize to enforce the rules according to his own interests.

Change is generally caused by human activities. This excludes change caused by other, natural events, such as floods and earth quakes or changing riverbeds causing change in parcel boundaries. We regard natural events as separate cases and do not include them into the discussion here.

4.3 Cadastre as part of institutional reality

Now we justify our claim that reality in a cadastre is part of institutional reality.

Real property covers a section of the surface of the earth, but the essence of real property is the relation between the owner and the land. Without a society a person would hold land in possession, rather than own it (Stubkjaer 2001), because rights are only created within the legal system of a society. The foundation of real estate lies much more in the different kinds of rights over land as it does in the physical dimension of the land itself (Smith & Zaibert to appear). Reality in a cadastre is determined by rights, which are institutional concepts.

Rights (e.g., ownership) are legal concepts which are regulated and introduced by the legal system and for the case of landed property codified in the cadastral law. Laws represent the codified rules of institutional reality. Rights and the relationship between owners and land are clearly part of institutional reality.

Real estate is a product of the deliberative or intentional activity of human beings (Smith & Zaibert to appear). Property rights on land differ insofar from other forms of property rights that they not only capture the institutional aspects of rights. Additionally the objects of the rights, i.e., the parcels, are themselves institutional. The existence of a parcel is a matter of human institutions, without human institutions it is only raw land (Zaibert 1999).

The objects the cadastral registry deals with, the parcels, and the relations between owners and parcels, the rights, are institutional. Reality in a cadastre does not only comprise parcels and owners of rights. Reality in a cadastre also comprises courts, the cadastral registry itself, and sheriffs enforcing the power of the state. But these elements of reality exist only in the context of the legal system and are therefore also part of institutional reality.

Institutional reality in a cadastre is always based on physical reality, i.e., institutional reality does not exist independent of physical reality. The phenomena, the cadastral registry deals with, are of institutional nature, which is grounded in physical reality.

4.4 Parcels and land pieces

In this section we will investigate the relationship between land pieces, which are physical facts existing in external reality and parcels, which are institutional facts.

4.4.1 Fiat and bona fide boundaries

Smith introduced the distinction between fiat and bona fide boundaries (Smith 1994, Smith 1995). Bona fide boundaries are boundaries in the objects themselves. They exist independent of human cognitive acts. They are discontinuities in the underlying reality (Smith 1995). In our terminology they are physical facts determining the boundaries of physical objects. Coast lines and rivers are typical examples of bona fide boundaries. Fiat boundaries owe their existence to acts of human decision or related human cognitive phenomena (Smith 1995). In our terminology they are institutional facts, i.e., status functions assigned to the boundaries of physical objects in external reality. The boundary between the eastern and western hemisphere or the boundary between the Atlantic and the Indian Ocean are examples of fiat boundaries.

Fiat boundaries are created based on bona fide boundaries or without the existence of bona fide boundaries. Fiat boundaries can exist before bona fide boundaries (e.g by defining the boundary on a map). After the creation of the fiat boundary a bona fide boundary can be constructed, for instance, by boundary markers, such as, fences.

The distinction of fiat and bona fide boundaries is helpful for the discussion of real estate. It is assumed that the spatial dimension of real estate is appropriately described by the fiat – bona fide dichtonomy (Stubkjaer 2000).

4.4.2 Parcel boundaries

Strictly speaking there are no parcel boundaries on raw land. The existence of parcels of real estate is a matter of human institutions, without human institutions there is only raw land. The existence of land parcels is ontologically dependent on a highly complex system of cognitive acts, beliefs and expectations of human beings (Smith & Zaibert to appear). Also bona fide parcel boundaries are only boundaries if someone regards the concerning discontinuity on the earth surface as a boundary (Zaibert 1999).

To characterize parcel boundaries we need to introduce the concept of a *legal boundary*. Legal boundaries are always fiat boundaries. Perhaps they can be related to bona fide boundaries, as it is the case for coast lines, but they need not. Legal boundaries are characterized by the fact that they are secured and enforced via state power (Smith & Zaibert to appear).

What an owner owns is not raw land, what an owner owns is a parcel. A parcel is based on representations of reality in a cadastral map. Parcels are represented by their legal boundaries in the map. Parcels are created by registration in the cadastral map, for example, in the case of the foundation of a new cadastral registry or in the case that a parcel is divided and a new parcel is created by the registration of the divided part of the parcel.

4.4.3 Parcels and rights

Parcels are characterized by their identifiers. The spatial extension of a parcel is determined by its legal boundary, its identifier not. If the parcel boundary changes, the identifier of the parcel remains unchanged. The identifier of a parcel is described by a name or a number. For instance, the name 'Margaretengürtel 14' uniquely identifies the parcel, where the building is erected, which is the place where I live.

Rights on parcels are related to parcels as a whole. That means the rights are related to the identifier of a parcel and not to its spatial extension. Strictly speaking the space of land ownership is not the earth surface, it is a space of parcel names which identify a parcel on a map, which is characterized by its legal boundaries. These legal boundaries themselves are fiat boundaries. The maintenance of the correspondence of legal boundaries and bona fide boundaries is an important task for a cadastre, but it does not affect the legal situation of the parcel itself.

Figure 4.1 shows the connection between land, parcels and rights (ownership). To summarize: Parcels are constructed based on raw land by introducing legal boundaries, which are represented on a cadastral map. Parcels are characterized by their legal boundaries (which are always fiat boundaries) and their identifier. Rights are always related to the parcel as a whole, i.e., to the parcel identifier.

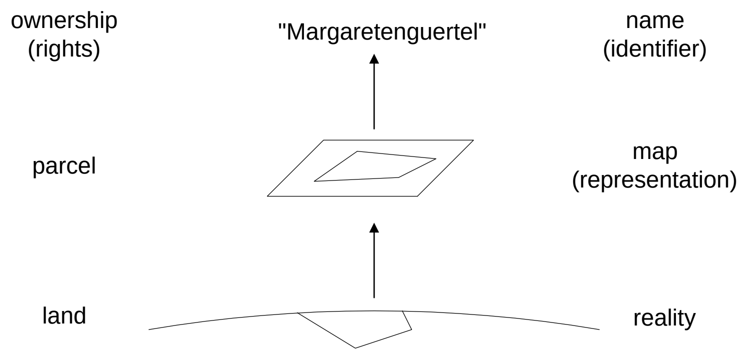

Figure 4.1: The relationship between land, parcels and rights

4.5 Institutional facts and status indicators in a cadastre

For the understanding of the system of documentation with the cadastral registry as its central part, the investigation of the relationship between the creation of institutional facts and the creation of status indicators is the key element. We argue in this section that the cadastral registry is a system of status indicators for events changing the institutional status. The core point is that the act of creation of the status indicator and the act of changing the status are identical.

In reality in a cadastre institutional facts are usually created by communication actions, i.e., by declarative speech acts (see subsection 3.8.1). These institutional facts are not derivable from the physical properties of the phenomenon to which the status is assigned (see section 3.4.3). Neither the physical properties of a human being allow to derive the status 'owner of a parcel', nor the physical properties of a piece of land are sufficient to derive the status 'parcel'.

Humans perceive events by their effects. Communication events creating institutional facts, do not necessary create physical effects after they have finished, in particular in the case of oral communication. In this case not only the status itself is not visible, also the phenomenon, which created the status, is not visible. This is a critical source of mistakes, fraud and errors in the field of cadastre.

Status indicators (see section 3.4.7) represent institutional status. The system of documentation in the context of the cadastral law (see section 2.10) serves as status indicators for the existing institutional facts regarding landed property. Contracts are an example of status indicators. The central element in the system of documentation is the cadastral registry representing and indicating the legal status of land pieces and the status of human beings regarding land.

A major issue for the system of status indicators, i.e., for the cadastral registry, is to maintain the correspondence between the status they indicate and the actually existing institutional facts. The solution to this issue is the combination or at least the association of the act of representing the fact (the creation of the status indicator) with the act of creating the institutional fact. If two persons sign a contract, this event, which represents the creation of the contract, itself creates the contract. In the cadastral registry (in a title registration system) the act of transferring a right is identical with the act of registration. The right will be transferred by registering, i.e., representing this event in the cadastral registry.

Not in all cases the connection is so strict. In a deed recording system the act of transferring a right is not identical with its registration, but closely linked. The transfer of status takes place by signing the deed. In this case the creation of institutional facts depends on the content of the registry, i.e., a particular status can only be proved based on the content of the registry, which stores the deeds.

The system of documentation is imperfect, since there are events possible that change the institutional status, which are not registered in the registry. For instance, the death of a person changes the institutional status of a parcel the person owns, without creation of the status indicator in the registry.

The system of documentation in a cadastre has an epistemic and an ontological effect. The registration of lands affects landed property as an aid to knowledge. For instance, it gives information about who owns a parcel and where a parcel begins and ends. The ontological effect of registration alters the institution of land itself, its status and structure (Smith & Zaibert to appear).

4.6 Rights in a cadastre

In section 2.8.3 we described ownership with the metaphor 'bundle of sticks'. Ownership of a parcel is different from other property rights. If one gives away all sticks from the bundle the ownership right itself remains unaffected. A person remains owner of a parcel even though he perhaps gave away all rights connected to the ownership. Perhaps he gave away the right to use the land he owns or the right to sell the land. He remains owner. Speaking in Searle's terminology the ownership right is the institutional fact and the individual rights are conventional powers assigned to the institutional fact. The institutional fact remains unaffected even though no conventional power is assigned.

Different types of conventional power in Searle's sense can be expressed by different types of rights (see section 3.6.5). Rights relevant for the cadastre can be characterized by the two basic types of rights *duty* and *power* (in the legal sense). Duties describe obligations between persons, whereas powers in the legal sense define possible activities of autonomous individuals.

The legal system defines the general rules for persons to act according to their own interests. It gives persons the power (in the legal sense) to enforce their interests with the help of the state. Persons may act or not according to the powers (in the legal sense). For instance, if the ownership right of a person is not recognized by another person, the owner has the power (in the legal sense) to enforce his right, i.e., he may enforce his right or not. If a person acts according to his power (in the legal sense) it is possible to create claims which can be described by the corresponding duties (a person has to abandon illegal land use, a person has to to transfer ownership, a person has to pay an amount of money).

Two kinds of activities need to be investigated in the analysis of a cadastre: activities of persons according to the powers (in the legal sense) they enjoy and activities of persons according to duties they have. Institutional change in a cadastre can be described by investigating these two basic types of rights.

4.7 Social processes in a cadastre

Change plays a major role for the cadastre (Al-Taha 1992). Reality in a cadastre is determined by social processes causing change. Since it is impossible to investigate the whole variety of social processes characterizing social reality in the field of cadastre in one thesis, we focus on processes we regard as characteristic for the cadastre.
In the field of cadastre we distinguish two kinds of processes:

1. Processes where human beings act according to the legal system.
2. Processes where conflicts occur, i.e., processes where human beings violate legal rules, and where the organizations of the legal system resolve the conflicts.

Social processes according to the legal rules mostly deal with legal change of parcels, which are the objects of rights in a cadastre (see chapter 2). Ownership of a parcel is the central element determining the legal status of a parcel. Here we discuss the transfer of ownership of a parcel between two persons under Austrian law (see section 2.9) as characteristic case of processes of the first category.

Conflicts between persons arise if persons act violating legal rules, i.e., if their physical activities do not correspond to their rights (see section 3.6.2 for a discussion of physical capabilities and rights). A central function of the legal system is to resolve conflicts between persons, which is the foundation for the peaceful living together of people (see section 2.4). Processes of the second category comprise a physical component (i.e., physical activities and the physical force of the state) and an institutional component (i.e., the process to resolve the conflict and to mobilize the force of the state). As characteristic case of a process of the second category we simulate the complaint of an owner against a person unauthorized using a parcel he owns and the following execution process to mobilize the force of the state (see section 2.11).
The case studies we discuss in this thesis consist of the following two processes:

1. The transfer of ownership of a parcel between two persons.
2. The conflict between to persons regarding the use of a piece of land consisting of two sub-processes:
    (a) The complaint of the owner of a parcel against an unauthorized user of his parcel.
    (b) The judgement execution against the unauthorized land user.

4.7.1 The transfer of ownership of a parcel

In the scenario of the ownership transfer the following actors are involved:

- A, the owner of a parcel, who wants to sell a parcel,
- B, a legal person, who wants to buy a parcel,
- the cadastral registry, which maintains the system of documentation and changes the institutional status.

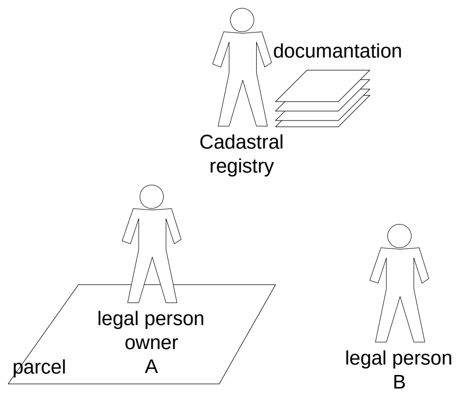

Figure 4.2: The scenario of ownership transfer

Figure 4.2 shows the scenario and the actors involved.

The owner A of a parcel offers his parcel to another legal person B for sale. B checks the ownership of A by asking the cadastral registry. If the registry confirms the ownership of A, B accepts the offer. Accepting the offer counts as contract. According to the contract A applies for ownership transfer at the cadastral registry. The cadastral registry transfers ownership and B becomes the new owner. Figure 4.3 shows the individual activities during the process.

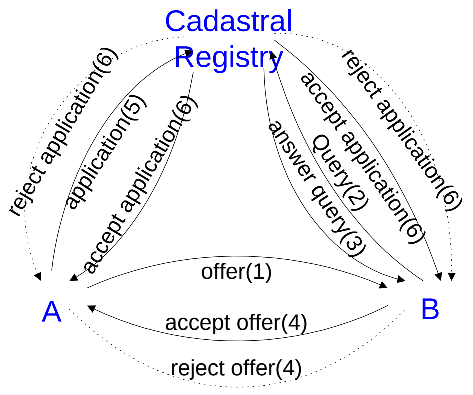

Figure 4.3: The individual activities during ownership transfer

4.7.2 Conflicts regarding land use

The scenario consists of two sub-processes, the complaint and the judgement execution. We assume in the scenario that there is an activity ’land use’ covering the variety of possible physical activities of human beings regarding land. For the simplification of the discussion we regard land use as exclusive, i.e., a piece of land one person uses, cannot be used by another person.
In the scenario the following actors are involved:

- A, the owner of a parcel, who wants to use his parcel,
- B, a legal person unauthorized using the parcel of A,

- the cadastral registry maintaining the system of documentation that proves the ownership of A,
- the court, responsible for the resolution of the conflict between A and B
- the sheriff, who is responsible for the enforcement of the legal rules

Figure 4.4 shows the scenario and the actors involved.

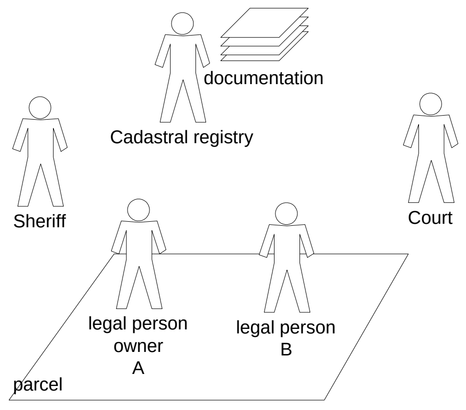

Figure 4.4: The conflict scenario

The owner of a parcel A tries to use his parcel and observes that a person B already unauthorized uses his parcel. A, i.e., the authorized user of the parcel, sues the unauthorized user B. In order to achieve success of his complaint the authorized user has to prove his ownership at the court, by providing a copy from the content of the cadastral registry. The court decides the case and pronounces the judgement. For the unauthorized user B the judgement creates the obligation to abandon the land use. For the plaintiff A it creates the execution title.

If B does not abandon the land use after the judgement, the execution title empowers A to mobilize the power of the state to enforce his right, i.e., the execution title enables the application of the judgement execution. If A can prove the execution title, the sheriff responsible for the execution evicts the land. The legal situation is re-established and A can use his piece of land according to his interests. Figure 4.5 shows the individual activities of the actors during the conflict resolution process.

4.8 Ontological categories of phenomena in a cadastre

According to Searle status functions can be imposed on different ontological categories of phenomena (Searle 1995, p.97):

- People,
- Objects and
- Events.

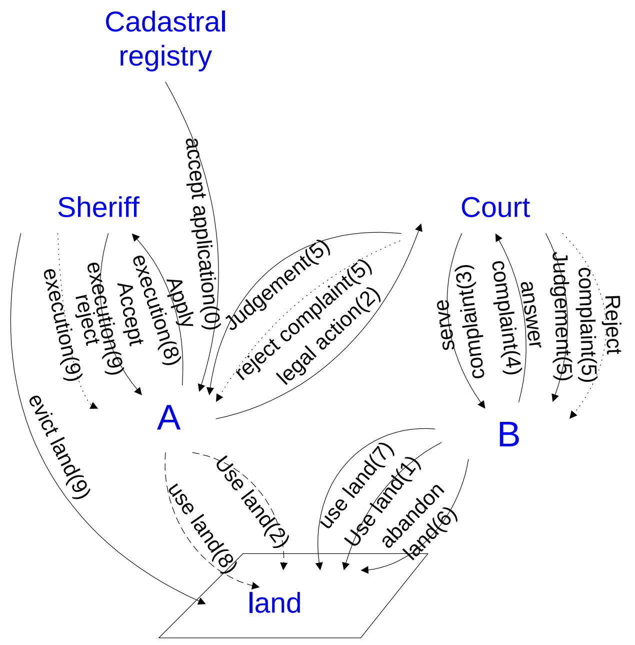

Figure 4.5: The activities during the conflict resolution process

We apply this categorization to the phenomena found in a cadastre. Figure 4.6 shows the categories according to the example.

We regard the term 'object' only in a restricted sense. We exclude events from the category of objects. Events are ontologically different from objects. Their existence depends on the existence of basic objects on which the event occurs. For instance, the event 'falling of a stone' depends on the existence of the object that falls and perhaps on the existence of a human being that caused the event. Events endure a limited time, whereas objects can be regarded as static in time. Events change the properties of objects. We only discuss events caused by human activities, i.e., we exclude external events, such as earthquakes from the discussion. We distinguish two kinds of human activities: communication actions causing communication events and physical actions causing physical events.

We distinguish subjects from objects. We use the term 'subject' instead of people to include organizations into this category. Strictly speaking subjects are a subclass of objects, which are the active entities in the world. Subjects inherit all properties from objects and have additionally the capability to perform actions. By their activities subjects cause events and change in this way other objects and subjects. Subjects are the central category in the ontology of reality in a cadastre because they cause all change in the real world. Subjects can hold rights on objects and are able to assign status functions to phenomena. They are involved in the construction of all institutional facts. The differentiation between the ontological categories 'object' and 'subject' corresponds to the distinction made in the law (see section 2.7) which distinguishes subjects of rights from the objects of rights.

According to the process of ownership transfer, the category of subjects comprises legal persons. Legal persons are the subjects of rights. A specific person is the owner of the parcel which has some rights to perform activities concerning the parcel.

The category of objects comprises parcels and the system of documentation. Parcels are the objects of property rights whereas legal documents represent change in a cadas-

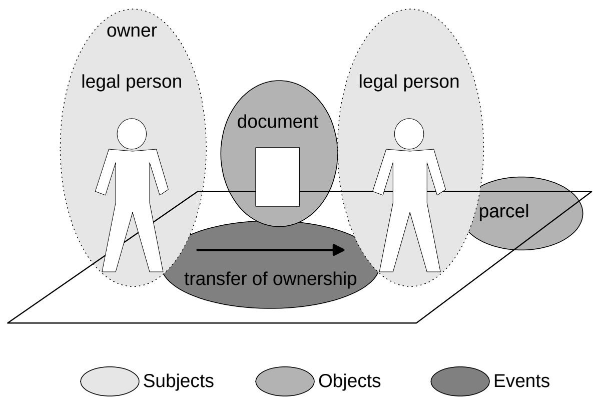

Figure 4.6: Ontological categories of phenomena in a cadastre

	Physical reality (phenomena)	Institutional reality (status)
Objects	land pieces, documentation	parcels
Subjects	human beings	owner, legal person
Events	human action (writing on paper)	transfer of ownership

Table 4.1: Ontological categories on two levels of reality

tre, they are status indicators. Parcels belong to the specific category of immovable things.

The category of events comprises the transfer of ownership. The event is caused by an activity of the subjects involved in the event. In this case the event is caused by a communication action, i.e., a speech act, which is physically realized by a written document. Figure 4.6 shows the different categories of phenomena involved in the transfer of ownership of a parcel.

4.9 Levels of reality in a cadastre

The distinction between the physical and institutional level of reality has to be considered in the analysis of reality in a cadastre. On the physical level as well as on the institutional level there are facts of the three ontological categories.

Objects, subjects and events exist only on the physical level of reality. We conceptualize the physical level as consisting of phenomena of these categories. Nevertheless the categorization applies to the institutional level as well. There are status functions which we can distinguish according to the kind of phenomenon they are assigned to. We differentiate between status assigned to objects, to subjects and to events. In general we designate entities on the physical level as *phenomena* and entities on the institutional level as *status* (see section 3.7) . Table 4.9 shows the facts on the two levels of reality with regard to the process of ownership transfer.

Within the category of subjects on the physical level there are human beings. These human beings count on the institutional level as legal persons and owners. Within the category of objects we can distinguish parcels on the institutional level and pieces of land on the physical level. In the category of objects the system of documentation exists, which is physically realized in some way. The system of documentation has the property that it represents and makes apparent institutional status assigned to events. Events on the physical level are human activities. In the case of ownership transfer it is a speech act (see subsection 3.8.1) where the locutionary act, the physical foundation of the speech act, is, for instance, the action of a human being who writes on a piece of paper. The event 'writing on paper' counts on the institutional level as transfer of ownership which is represented on paper in the registry. Figure 4.7 shows the example of ownership transfer separating the physical and institutional level.

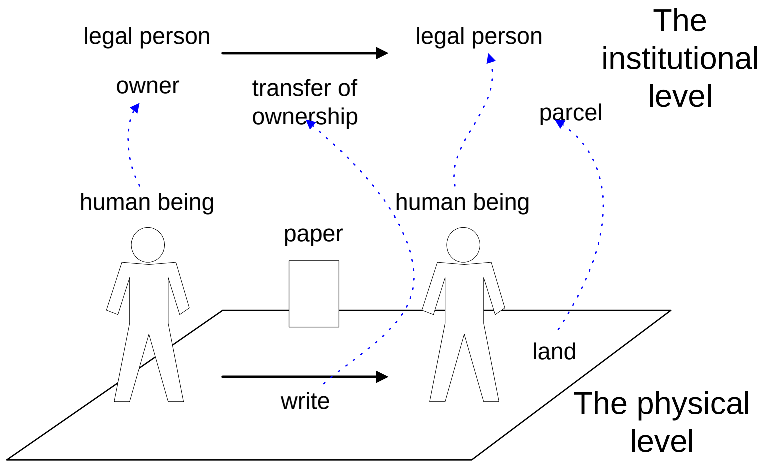

Figure 4.7: Levels of reality in a cadastre

The e-ontology of reality in a cadastre, which we develop based on the distinction of the physical level of objects, subjects and events and of the institutional level of status, can be regarded as subset of the five tiered ontology (section 3.3). It focuses on the object world tier and on the social reality tier.

4.10 Facts vs. rules for their creation and existence

Searle's theory distinguishes between institutional facts and institutions. Institutions consist of sets of constitutive rules and define the conditions for the creation and existence of facts (see section 3.4.8). The distinction between rules for the creation and existence of facts and the actually existing facts is a fundamental property of reality which does not only apply to institutional facts. It is also a property of physical reality. In the following we write rules and facts when we talk about this distinction.

The idea is comparable to the following metaphor: Given a theatre performance. The script defines the rules for the play of the actors. The stage and the actors and the activities they perform are phenomena actually existing in reality, are facts. The script describes what kinds of facts possibly will be created. This must be distinguished from the actual play of the actors, i.e from the actually evolving situation in reality. Nevertheless the actual situation in reality depends on the rules defined in the script.

Facts describe what actually *is* in the world and rules describe how the world can evolve. Rules describe how facts can possibly be created and destroyed. Facts represent a state of the world (current, former oder future state), whereas rules represent how the world can change between these states.

The distinction between facts and rules applies on the institutional level. Constitutive rules define the rules for the creation and existence of institutional facts. For example, constitutive rules describe under which conditions a human being counts as owner. This must be distinguished from actually existing owners in reality. Figure 4.8 illustrates the situation with regard to this example.

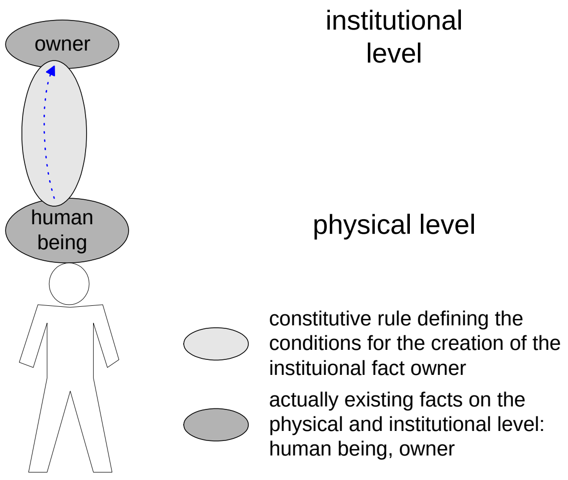
owner
institutional level
human being
physical level
constitutive rule defining the conditions for the creation of the instituional fact owner
actually existing facts on the physical and institutional level: human being, owner

Figure 4.8: Rules vs. facts in a cadastre

The distinction between facts and rules applies on the physical level. Rules (the rules of physics) define the conditions for the creation of objects, subjects and events as well as their properties when they exist. The physical, in particular geophysical laws describe how pieces of land are created as well as their properties when they come into existence. The physical laws must be separated from the actually existing objects. Since we are interested in the change of status of objects and not in the change of the objects themselves, we can regard objects as static, i.e., we can assume that objects do not change. Consequently the rules for the creation of objects do not play a central role. Rules are much more important in the discussion of events describing the conditions causing an event and its possible effects. We discuss this issue now in connection with human actions, physical capabilities and rights.

In section 3.6.2 we introduced the distinction between (physical) capabilities and rights. Capabilities define possibilities for activities of human beings on the physical level. Rights define possibilities for activities that assign a specific status to a phenomenon. Capabilities and rights can be regarded as rules defining the conditions for human activities. The human activities, actions performed by persons, must be distinguished from the rules that describe their possibility. Capabilities define, for instance, the physical ability of a human being to write on a piece of paper. This ability enables the possibility to perform an action of this type. A human being can write a specific moment in time on a piece of paper. Rights, for instance, define the possibility of an owner to transfer ownership. This possibility must be distinguished from the actually occurring event 'ownership transfer' that a person causes by an action.

Rules on the institutional level are always grounded in physical reality. This applies to constitutive rules as well as to rights. On the bottom of a hierarchy of constitutive rules is always a physical fact, rights are always based on physical capabilities of human beings.

4.11 The structure of reality in a cadastre

We introduced three essential aspects in the analysis of reality in a cadastre: ontological categories of phenomena, levels of reality and the distinction between facts and the rules for the creation and existence of facts. The three aspects cannot be expressed in terms of each other. They represent different views on the issues involved in reality in a cadastre, none of them can be omitted. The idea is that every combination of the aspects is meaningful and expresses an element of the structure of reality. For instance, there are facts on the physical level within the category of subjects: human beings. There exist rules on the institutional level within the category of events: constitutive rules for an ownership transfer.

The discussion of all these combinations of the essential aspects forms a comprehensive view of reality in a cadastre. The analysis can be expressed in terms of these aspects and their interrelationships. Figure 4.9 shows the structure of reality in a cadastre.

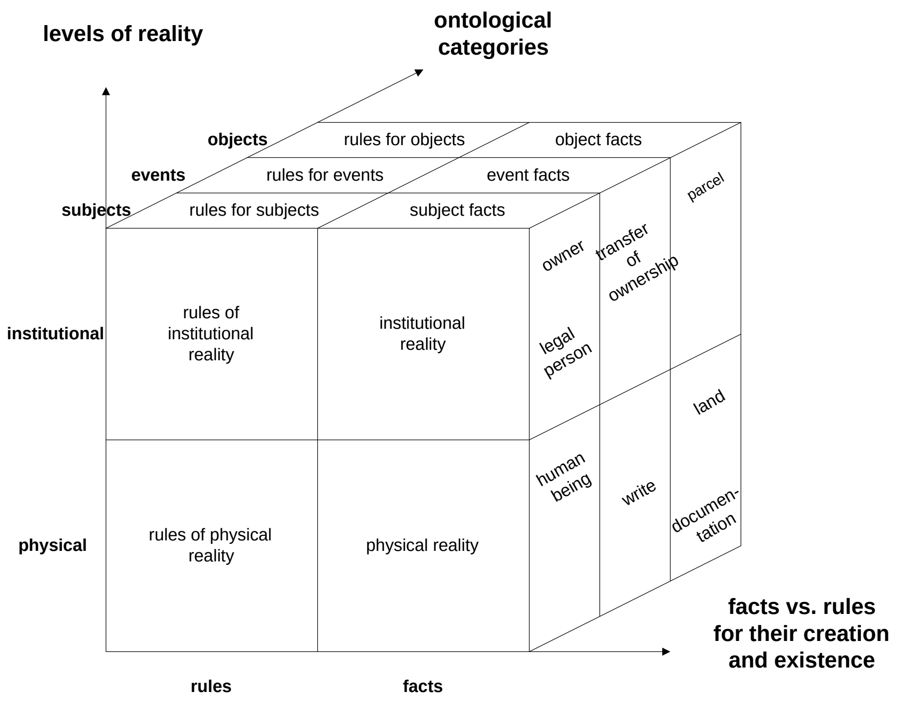

Figure 4.9: The structure of reality in a cadastre

The representation in the cube allows distinguishing 12 sub-classes in the whole structure of reality in a cadastre (corresponding to the sub-cubes). In the following we discuss these subclasses.

4.11.1 The physical level of reality in a cadastre

The discussion starts with the physical level of reality in a cadastre, which corresponds to the lower part of the representation (figure 4.9). We investigate the phenomena involved according to the ontological categories of phenomena.

Physical objects

Facts. The central element in reality in a cadastre is the land. Reality in a cadastre is grouped around the management of this resource (see section 2.2). Rights are not directly assigned to raw land pieces but rights concern land (see section 4.4).

Reality in a cadastre needs a system of documentation (see section 2.10). Physical foundation for the system of documentation is usually paper. Modern systems support an electronic based system of documentation (see section 4.5 for the discussion of the role of documentation).

We identify two classes of physical objects which are important for the cadastre: a system of documentation and land pieces. We leave open the concrete physical manifestation of the system of documentation.

Rules. According to the two classes of objects relevant for reality in a cadastre we now discuss rules for land and the system of documentation.

The rules for land describe what qualifies as land pieces which human beings would accept as parcels. Land is part of the surface of the earth. Its evolution is described by geophysical laws. The main important property of land is its spatial extension, in particular its two-dimensional extension. Land must be usable by human beings, for instance, for agriculture or to erect buildings on it. The term 'land' usually means parts of the solid earth surface and cadastral systems mostly deal with solid land pieces. Land in a broader sense can also comprise water-covered parts of the earth surface, which offers different possibilities of usage for human beings.

The rules for the system of documentation represent the requirements for physical objects to be used for the representation and, in particular, for the registration of ownership rights concerning land. To be used for documentation the physical objects must be durable to maintain the information they contain over a long period. Physical objects must be manageable for human beings in order to use them for documentation. This means that the spatial extension, its weight and form must be in a way that it can be used for maintaining information. Physical objects must be changeable in a way that they can bear the data by some aspects of its physical properties. The physical manifestation of the system of documentation is not the crucial question for the cadastre. Important is the fact that it serves its purpose as system of status indicators (see section 4.5).

Physical subjects

Facts. Physical subject facts are the most important entities determining change in reality in a cadastre. Physical subject facts are human beings. We do not distinguish between groups of individuals forming organizations and single individuals. Both can be legal persons and thus be holder of the same rights and duties (see section 2.7). Organizations of individuals are represented by single individuals acting for the whole group. Human beings are the active entities in reality, because they are capable of acting in their environment, i.e., to perform actions.

Rules. Physical subject rules concern in the first place physical capabilities of human beings to perform activities and their effects (see section 3.6.2). Physical capabilities relevant for reality in a cadastre are all kinds of land use, communication actions and physical violence against other human beings. The effects of human activities are events that change the physical environment.

Physical events

Facts. Events relevant for reality in a cadastre are caused by physical subjects, i.e., human beings. We distinguish physical and communication events for clarification, not because there is a fundamental difference. Communication events are physically determined by air waves in their oral form or physical changes of objects, which we can interpret as writing.

Events can be identified with the effects of human actions since we do only investigate change caused by human communication actions. Nevertheless human activities are not the only source of change in a cadastre. Events not under consideration here are, for instance, floods and earthquakes, and the death of a human being.

Rules. Physical event rules describe the possible effects of events. Events can affect human beings or objects. Events affect human beings by changing their mental state in the case of communication events, or physically by violating. Events can affect physical objects, in the case of reality in a cadastre pieces of land or the physical manifestation of the system of documentation.

4.11.2 The institutional level of reality in a cadastre

This subsection discusses the institutional level of reality in a cadastre, which corresponds to the upper part of the representation (see figure 4.9). We investigate the institutional status involved according to the three ontological categories. We do not use the terms 'institutional object', 'institutional subject' or 'institutional event'. No objects, subjects and events exist on the institutional level of reality, there are only status functions (for short status) assigned to phenomena. Therefore we talk about status and facts (see section 3.4.9).

Status assigned to objects

Facts. Status is assigned to land pieces. The assignment of status to land pieces creates new institutional facts of the type parcel. Rights concerning land pieces are defined in the cadastre with regard to parcels, not the physical land itself (see section 4.4).

The system of documentation is itself not an institutional fact, it is in the first place a status indicator for events that changed the legal status of parcels (see section 4.5).

Rules. Parcels are created by an act of foundation by the state or by dividing existing parcels. The creation of a parcel means the registration of the legal boundaries and of a unique name for the parcel (see section 4.4).

There are no rules for the system of documentation because it is not an institutional fact. We regard the function of the system of documentation as purely epistemic, i.e., it provides knowledge about the institutional status of subjects objects and events.

Status assigned to subjects

Facts. Reality in a cadastre is determined by activities of legal persons and by activities of the state. The state acts through its organizations or representatives. The organizations and representatives relevant for the cadastre are the cadastral registry, responsible for the registration of legal transactions between legal persons concerning land, the court responsible for resolving conflicts between legal persons and the sheriff, who represents the physical power of the state to enforce rights of legal persons. A specific class of legal persons are owners of parcels who hold an exclusive right on parcels.

Rules. Institutional subject facts are created according to the legal rules by assigning status to human beings. The status of the representatives of the state is created by an act of foundation by the state. Reality in a cadastre is concerned with change within the legal system, not with the change of the legal system itself. The institutional status of the representatives of the state can be regarded as static and given. The cadastral registry, courts and sheriffs do not change their status, thus it is not necessary to include constitutive rules for their creation into the discussion.

Rules on the institutional level comprise rights which are connected to status assigned. We assume that the following rights are important for reality in a cadastre: A legal person can have the right to physically use a piece of land a parcel refers to (the content of the ownership right (see subsection 2.8.3)). A person can have the right to transfer the ownership right to another legal person (see subsection 2.9.1). A person can have the right to perform physical power against legal persons. The state reserves this right for some of its representatives, the sheriffs (see section 2.4). Persons have the right to conclude contracts. Persons have the right to sue other persons if they believe that their rights are restricted (see section 2.11). Subjects have the possibility to perform activities according to these kinds of rights.

Status assigned to events

Facts. Institutional change occurs by assigning status to physical events. In reality in a cadastre institutional change occurs by communication actions between persons, i.e., we need only to investigate communication events. Communication events are speech acts of different categories (see subsection 3.8.1 for the discussion of speech acts). Declarative speech acts are performed by representatives of the state, namely by the cadastral registry in registering ownership transfer (in the Austrian system, see section 2.9.1) or by the court pronouncing a judgement (see subsection 2.11.1). A person can exchange knowledge by performing representative speech acts (e.g., inform). In particular the cadastral registry informs other persons about the legal status of parcels. Persons can perform commissive speech acts, i.e they can promise to perform particular activities (such as: an offer).

Rules. Institutional event rules define the possible effects of events on the institutional level. The effects relevant for the cadastre is change in the institutional status of persons or parcels. In particular the effects of declarative speech acts performed by the cadastral registry are ownership transfers concerning parcels (under Austrian law, see section 2.9.1). Speech acts can also increase the knowledge of persons (speech acts transport information) or they can create duties (e.g., in the case of promises).

4.12 Assumptions about reality

The analysis so far allows to make some assumptions about reality in a cadastre, which are sufficient for the cadastral domain and allow the simplification of the general theory.

In reality in a cadastre *the hierarchy of constitutive rules is confined to two levels.* On the lower level there are rules that assign status directly to events. For instance, in the case that an agent accepts an offer to buy a parcel, the status "contracting" is directly assigned to this event. On the higher level status is assigned to subjects and objects that participate in this event. In the case of contracting the status "buyer" and "seller" is assigned to the participating agents. These two levels are sufficient to explain all aspects of reality captured by the analysis.

We can assume the existence of statically existing physical objects and subjects. The analysis works with a predefined set of human beings, respectively particular groups of human beings (the cadastral registry, the court, a buyer of a parcel), and land pieces, which are the main important elements on the physical level of reality (see subsection 4.11.1). Accordingly no rules for the creation of these physical objects need to be given. For example, if we do not investigate the death of human beings, we do not need to analyze the rules for such event occurrences. We will not discuss legal change caused by the death of persons.

We distinguish status that can be changed by events (regarded as *dynamic*) and status that remains *static*. For instance, it is assumed that land pieces do not change their institutional status. Further we assume that the legal persons in the system do not change, i.e, for instance, the death of a person is not represented. Additionally the representatives of the state, i.e., the cadastral registry, courts and sheriffs, do not change their institutional state. Static beliefs do not correctly represent reality, they are a restriction of the analysis that does not investigate change in these beliefs.

The argument justifying the assumption of static objects and status functions is that, first, some aspects of reality are outside the scope of the analysis, which can be held as constant. Second, some aspects change much slower than the phenomena under consideration that they can be regarded as static.

Short term status beliefs and *long term status beliefs* can be distinguished. Short term beliefs have only conventional power during the events that created them. That means that they are only connected to rights during the event itself. For instance, to the status "seller of a parcel", rights are only connected during the process of selling. After the parcel is sold the function of the belief is purely epistemic, i.e., it represents the knowledge about the fact that there occurred the sale of a parcel at some moment in time in the past. Long term status beliefs keep their conventional power if the constituting event is finished. For instance, to the the status owner of a parcel, rights are connected as long as no change in the ownership of the parcel occurs. All the time there exists the right to use the parcel and to exclude every other person from the land use (see subsection 2.8.3).

4.13 Summary

In this chapter we developed an e-ontology of reality in a cadastre. By the term 'reality in a cadastre', we understand the cadastral registry embedded into its environment and its interaction with the human beings acting in this environment. We justified the assumption that reality in a cadastre is a part of institutional reality.

The chapter investigated the relationship between physical pieces of land, parcels and connected rights. We introduced the concept 'legal boundary', which is always a fiat boundary and can be related to a bona fide boundary. Rights are connected to parcels as a whole, i.e., to their identifies, and not to their spatial properties (e.g., legal boundaries). The change in the legal status of parcels is not affected by simple change of the boundary.

The creation of status indicators and the creation of institutional facts is closely linked. If possible, the process of the creation of the fact is connected to the process of the creation of the status indicator in order to achieve the most complete correspondence between the system of status indicators in the cadastral registry and the institutional status in reality.

We developed an ontology of reality in a cadastre based on the distinction of three aspects of the analysis:

- Ontological categories of phenomena,
- Levels of reality,
- The distinction of facts and rules for their creation and existence.

These aspects are related and we discussed each combination of the three aspects.

The analysis of this chapter is independent of national cadastral systems, whereas the social processes we used as case studies are from the Austrian legal system. We presented a comprehensive view of the cadastral registry embedded in its environment based on the ontology of institutional reality presented in chapter 3.

The analysis of reality in a cadastre concludes the first part of this thesis. Its major goal was to develop the foundations for the construction of the computational model later in this thesis. We grounded the analysis on a solid philosophical foundation and identified the relevant aspects of reality necessary for the computational model. We achieved a comprehensive categorization of the issues involved which allows the construction of a minimal model only based on the essential aspects determining reality in a cadastre.

Chapter 5

The architecture of a multi-agent system

5.1 Introduction

For the construction of a model of (institutional) reality in a cadastre, the representation of human beings and their individual capabilities, decisions and actions is the core element, since institutional reality exists only in relation to human beings and their individual beliefs and activities (see chapter 3). We represent human beings and their capabilities in terms of *agents* and model reality in a cadastre in a multi-agent model.

This chapter introduces the conceptual background for the construction of the agent-based model of reality in a cadastre. It discusses multi-agent theory as the scientific discipline researching on agent-based models. It describes the main concepts of the field and focuses on developing an abstract agent architecture, which can be applied to the task of modelling reality in a cadastre.

The chapter starts with an introduction into the theoretical background of multi-agent theory. We introduce multi-agent theory without discussing representation mechanisms. This thesis regards an agent-based approach as purely conceptual framework. Representation issues will be discussed in chapter 6. Chapter 6 introduces algebraic specifications, which are implemented in the functional language Haskell, as representation mechanism for the agent-based model. Following the subdivision of the field of multi-agent theory into research on

- agent theories[1],
- agent architectures and
- agent languages (Wooldridge & Jennings 1995),

this chapter focuses on agent theories and and architectures and chapter 6 emphasizes the discussion of an agent language suitable for this thesis. Agent theories deal with questions, such as, what an agent is and what the general properties of agents are. Agent architectures discuss the types of agents and the structure of concrete agent architectures. Agent languages investigate languages for programming and implementing agents for experimentation and applications.

[1]to be distinguished from the term multi-agent theory describing the whole body of research

5.2 Overview of multi-agent theory

> "A new field is only defined by its problems, not its methods/technics" (Nwana & Ndumu 1999, p.8).
>
> "Intelligent agent technology is a rapidly developing area of research. However, in reality, there is a truly heterogeneous body of work carried out under the 'agent' banner." (Nwana & Ndumu 1996, p.3)
>
> "Es gibt sicher so viele Definitionen des Gebietes, wie es Forscher gibt, die darin arbeiten." (Schröder 1993, p.10)

Multi-agent theory is a young scientific field without common paradigms. Different people from different domains working in the field have different understandings of the concepts. There is much literature about the topic (e.g., (Ferber 1999, Weiss 1999, O'Hare & Jennnings 1996, Russell & Norvig 1995, Schröder 1993, Bond & Gasser 1988*b*)). Because of the heterogeneity of the field we give a comprehensive overview of the domain, which allows relating our approach to the whole body of research. We focus on the different viewpoints of the field, which are the core elements in understanding the different approaches to multi-agent theory[2].

Multi-agent theory is regarded as important new contribution to several areas of research. First, it is an important new way to conceptualize and implement software applications (Wooldridge 1999). Agent technology is expected to be one of the key computing paradigms over the next 10 years (Gilbert, Aparicio, Atkinson, Brady, Ciccarino, Grosof, O'Connor, Osisek, Pritko, Spagna & Wilson 1995). Second, it introduces the concept of collective intelligence and emergence of structures by interaction (Minsky 1985) into the field of artificial intelligence. Third, in the field of social simulation it offers the promise to allow the modelling of autonomous individuals and interaction between them (Gilbert & Troitzsch 1999). Multi-agent theory has the potential to stimulate and contribute to a broad variety of scientific fields.

5.3 Approaches to multi-agent theory

There are several approaches to multi-agent theory, according to different backgrounds, goals, viewpoints of the researchers involved. All capture different aspects of the field. This section discusses these viewpoints and characterizes the approach of this thesis in relationship to them.

5.3.1 Objectives to study multi-agent systems

There are three classes of objectives to study multi-agent theory (Bond & Gasser 1988*a*):

- a natural systems approach,
- an engineering-science perspective,
- a person-machine coordination approach.

[2]Much stimulation for this chapter came from the course by Paolo Petta "Software Agents" at the University Vienna in summer semester 2000 (Petta 2000).

The *natural systems approach* studies strategies and representations people use to coordinate their activities. It understands the field as a way to improve the understanding of human behaviour. Subject of research are people as social beings. An *engineering-science perspective* studies multi-agent theory with the goal to construct automated problem solvers for specific applications. Subjects of research are computer programs and machines and their application in the software developing and engineering process. A *person-machine coordination approach* brings these two perspectives together by studying the working together of people and machines. It analyzes collections of people and machines that work together in a coordinated way. Subjects of research are human beings and machines/programs as well as their interaction.

This thesis focuses on a natural systems approach. It regards agent-based models as a way to improve our understanding of some aspects of social reality, which is determined by the interaction of human beings.

5.3.2 Top-down vs. bottom up strategies

Bond and Gasser (Bond & Gasser 1988*a*) distinguish from a DAI viewpoint two sub-fields of multi-agent theory: research in *distributed problem solving* and research in *mulit-agent systems*. The first field considers how a single problem can be divided into sub-problems and distributed among a number of agents to develop a solution to the whole problem. Research in multi-agent systems investigates how autonomous agents can coordinate their knowledge and behaviour to solve problems. The distinction of these two sub-fields represents different philosophies in the construction of agent-based systems. Distributed problem solving is a *top-down approach* to the field starting on the macro level refining and dividing the whole problem into smaller pieces. Research in multi-agent systems represents a *bottom-up* strategy to the system design. It starts on the individual level with the definition and construction of single agents.

Following the analysis of chapter 3, which explains the structure of institutional reality in terms of the individual activities of human beings, we use a bottom-up strategy for the construction of the agent-based model. The core part of the model are the agents and their interaction with the environment, which together form the whole system.

5.3.3 Backgrounds to study multi-agent systems

Multi-agent theory is mainly influenced by two scientific fields, by distributed artificial intelligence (DAI) and mainstream computer science. The *DAI perspective* in general has the goal to construct systems which are intelligent. Intelligent in this context (there is no general agreement about what intelligence is) means that the system can solve particular problems at least as good as human beings (Barr & Feigenbaum 1981). Distributed artificial intelligence investigates systems of distributed agents which together can solve problems. It studies how a loosely coupled network of problem solvers can solve problems that are beyond the capabilities of the nodes individually (Durfee, Lesser & Corkill 1992).

Research influenced by mainstream computer science focuses on a *software engineering perspective*. This approach in general has the goal to design more efficient software systems. It does not investigate the intelligence of the system. It creates systems that interconnect separately developed agents, enabling the ensemble to function beyond the capabilities of any singular agent in the setup (Nwana & Ndumu 1999).

Both approaches emphasize the capabilities of the whole system beyond the capabilities of its parts, but they use different methods and focus on different applications. For instance, in the software engineering case object-oriented programing methods will be used to construct information agents. In the DAI case logical deduction methods will be used in the field of distributed problem solving.

5.3.4 Agents as a concept

Neglecting the focus on particular methods and applications, the core element of multi-agent theory is the agent concept. Much confusion arises from the fact that the term 'agent' is used both as technology and metaphor (Nwana & Ndumu 1996). The approach of this thesis is characterized by the focus on the agent concept. In this thesis we call this the *agent-based conceptualization approach* to multi-agent theory.

This approach does not focus on the technical methods, it does not focus on representation and reasoning mechanisms. It uses the term 'agent' as design model (Gilbert et al. 1995). It applies the agent concept as a metaphor for the description of the active entities in some domain. Agent-based models are regarded as conceptual framework for the representation of a domain of interest. The conceptualization will be expressed in a computational language. The language must be expressible and understandable enough to allow the representation of the agent framework. These are all requirements to the agent representation language. To summarize: *The agent-based conceptualization approach uses agents interacting in a multi-agent model as basic concept for the description and representation of a domain.*

5.4 Agent programming languages

An agent language allows programming hardware or software computer systems in terms of the agent concept (Wooldridge & Jennings 1995). It allows developing programs for experimentation with agent-based models. It enables the construction of applications based on multi-agent theory. Agent programming languages are the tool for implementing agent concepts and specifications. We distinguish *special purpose languages for programming agents* from *general purpose programming languages*, which can be applied for the implementation of agent-based models.

Special purpose languages have at least special constructs for the construction of agents. An example of this kind of languages is the *Agent0* programming language applying the agent oriented programming paradigm (Shoham 1993). The idea is to directly program agents based on mentalistic and intentional notions (such as beliefs, desires). Another attempt to construct an agent programming language is the GOLOG/CONGOLOG language (Levesque, Reiter, Lesperance, Lin & Scherl 1996, Shapiro, Lesperance & Levesque 1997). Based on the formal foundation of a modern version of the situation calculus (McCarthy & Hayes 1969) by Raymond Reiter (Reiter 1991) the language is defined in the logic programming language Prolog (Clocksin & Mellish 1981). The core idea of this approach is that many issues in artificial intelligence and program development can be avoided by providing a language based on a formal foundation. The agent programs defined in Prolog syntax have formal semantics with respect to a situation calculus theory.

General purpose programming languages are applicable to the implementation and representation of agent based models. Object oriented programming languages (Meyer

1988) provide a possible foundation for this task, because objects and agents have some common properties (as well as differences: see section 5.6), such as an encapsulated internal state and communication by message exchange.

This thesis uses a general purpose language for the implementation of agents allowing the representation of the model without the necessity to adopt the assumptions about the agent concept made in a special purpose language. We use the functional programming language Haskell for implementing agents. Chapter 6 discusses this issue in detail.

5.5 Applications

There are at least three broad areas of application for multi-agent theory. The boundaries between these areas are not sharp. Sometimes the same area of application is investigated just from different viewpoints. The three areas are:

- Artificial intelligence
- Program development
- Simulation

5.5.1 Applications in distributed artificial intelligence

In the field of artificial intelligence the agent concept can be applied in *Cooperative problem solving*, in the construction of *interface agents* and *information agents*. Research in cooperative problem solving (Durfee et al. 1992) investigates how groups of agents can cooperate to efficiently solve problems. The goal is to overcome complexity issues, to deal with the distribution of problems and to coordinate work. Interface agents can be used to improve the interaction between humans and machines. The metaphor is a personal assistant which cooperates with a user. The user delegates a range of tasks to personalized agents that can act on the user's behalf (Maes 1994). Information agents are WWW-based autonomous software agents that collect and supply information to humans and other computational agents (Decker, Pannu, Sycara & Williamson 1997). They are able to incorporate information from different sources to support its user. Agent theory can be applied to construct *artificial worlds* (Ferber 1999). The goal is to provide software environments to test and evaluate agent theories. Such software environments are called distributed artificial intelligence *testbeds* (Decker 1996). Testbeds are not applications in a narrow sense. They can be used as connecting element between theory and applications.

5.5.2 Applications in the field of program development

Several applications exist in the field of program development (see (Gilbert et al. 1995)): In systems and network management agent technology is used to manage the complexity of systems by distributing the administration task among a set of agents. They can automatically perform simple functions and filter the activities to be performed by the administrators. There is a high demand for improving messaging software by adding more "intelligence" to the system. Agents are applied to filter, sort, distribute, prioritize and organize messages for the user both in electronic and conventional mail systems.

5.5.3 Multi-agent simulation

Multi-agent simulation is the third application of multi-agent theory which is mostly relevant for the topics of this thesis.

The idea of simulation

A powerful way of explaining things, widely used in science, is in terms of models (Gilbert 1995). Computer simulation is a particular type of modelling. A model is a simplification of some structure or system, which is smaller, less detailed and less complex. The important point about a model is that it must be designed to be similar to the target domain in structure and behaviour (Gilbert 1993). The model is usually expressed in terms of a mathematical or logical description.

The idea of a simulation can be expressed as follows: Set up the model which embodies some plausible assumptions and see what happens. Afterwards compare the behaviour of the program (the simulation) with the observed behaviour in reality (Gilbert & Troitzsch 1999). Simulations have inputs and outputs. If the model is a computer program, the simulation consists of running the program with some specified input and observing the progress and output.

Simulations help to check models and theories in complex domains where analytical proofs of correctness are impossible or difficult to achieve. The formal proof is replaced by comparing the results of the simulation with the situation found in the real word. Figure 5.1 shows the idea of a simulation.

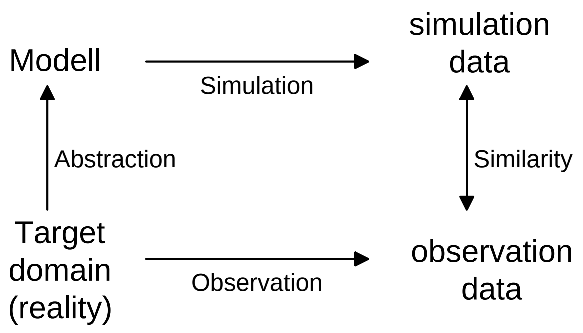

Figure 5.1: The idea of a simulation (after (Gilbert & Troitzsch 1999))

The methodology to construct simulations consists of the following two steps:

1. Specification of the model by abstraction from processes in the real world.

2. Use of the model to generate some output values which are compared to the actual data.

Simulations can be used for different purposes (Gilbert & Troitzsch 1999): The main important use of a simulation is to obtain a better understanding of some features of the real world. Here simulation is a way of explaining things, of explaining how some features of the real world evolve. Another classic use of simulation is for prediction. If we can construct a model which successfully reproduces some behaviour observed in the past, we can simulate the passing of time and predict the behaviour of the system in the future. A third use of simulations is to develop new tools to substitute for human capabilities. For example, expert systems have been developed to simulate the problem solving expertise of some professionals, e.g., of doctors. Simulations have been used for training purposes. Flight simulators are a typical example of this type of simulation. A related area of application for simulations is entertainment. Flight simulators, for instance, can be used in computer games too. Simulations can assist in discovery and formalization. Researchers can build simple models and can discover the consequences of their theories by running the simulation.

The process of formalization is a valuable discipline in its own right (Gilbert & Troitzsch 1999). In order to do a simulation, the ideas, conventionally expressed in textual form, have to be formalized into a specification which can be programmed in a computer. This process of formalization involves being precise about what the theory means. It includes ensuring that the theory is complete and coherent. This is an valuable process for the development of the theory.
There exist various approaches to simulation. We mention here:

- simulations based on differential equations,
- micro-analytical simulations,
- simulations based on cellular automata,
- agent-based simulations.

We will not discuss the approaches (see (Gilbert & Troitzsch 1999) for a comprehensive discussion) and focus on agent-based simulation.

The approach of agent-based simulation

Conventional simulations have two key disadvantages (Gilbert 1993):

- The behaviour of the simulated individuals is not justified in terms of individual preferences, decisions and plans.
- There is no interaction with other individuals.

Multi-agent simulation brings a radically new solution to the very concept of simulation (Ferber 1999, p.36). The development of multi-agent models offers the possibility of simulating autonomous individuals and their interaction. These models involving interacting autonomous agents, can be applied to the simulation of human societies. The goal of social simulation (Gilbert & Doran 1994, Gilbert & Troitzsch 1999) is to reach a better understanding of some features of the social world. Computer simulations in the social sciences is a rather new idea, but with enormous potential. This is because simulation is an excellent way of modelling and understanding social processes (Gilbert & Troitzsch 1999). Explicit models of social phenomena have not been so common, but with the development of powerful computers, computer models of social systems became possible. These models are designed to be run as processes within a computer, simulating processes that exist in the social world.
The following citation describes the *agent-based simulation approach* (Gilbert 1993):

> "The simulation works by cycling through each agent in turn, collecting messages sent from other agents, updating the agent's internal state by checking for any applicable rules, deciding on an action for the agent to

take and finally communicating the messages and the effects of the action to the environment, which responds appropriately. This is repeated for each agent and these cycles continue ... until the the simulation is stopped ..."

The main criterion determining the architecture we develop in this chapter is that it allows the development of agent-based simulation models. That means that it supports the agent-based simulation approach.

5.6 What is an agent?

This section introduces the concept 'agent' as we understand it in this thesis. We first introduce the concept and then discuss key properties of agents. We demarcate the agent concept from other concepts used in computer science and artificial intelligence.

5.6.1 The agent concept

According to the heterogeneity of the field there is no common agreement about a definition of the term 'agent'. This thesis follows the definition by Russell and Norvig regarding an agent as *"anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russell & Norvig 1995, p.31).* Figure 5.2 shows the principal ideas. The environment provides percepts to the agent, which the agent perceives through its sensors. The agent uses the percepts to select actions, which it performs in its environment through its effectors.

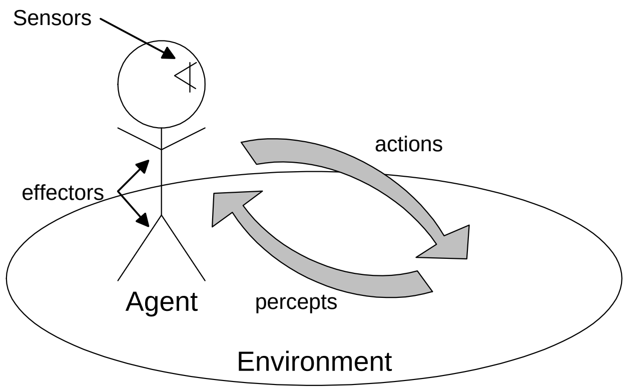

Figure 5.2: An agent embedded into its environment

This general definition allows the application of the agent concept to very different fields. The following examples demonstrate the usefulness of the agent definition and of the agent concept:

- *human beings*: Agents can be human beings that perceive their environment, the external world, through eyes and ears and act in in this environment with their arms and legs.
- *software processes*: If we regard an agent as a software process that encapsulates some state and communicates with other agents via message passing (Agha,

Wegner & Yonezawa 1993), then the agent is a program that acts and perceives by message exchange in its artificial software environment.

- *thermostats*: This definition is also applicable, for instance, to thermostats (Wooldridge 1999). The sensor of the agent detects (perceives) the temperature in its environment, a room. If the temperature is too low, the thermostat agent acts in its environment and switches the heating on.

Agents are situated in some environment and capable of autonomous action (Wooldridge 1999). Autonomy and its embedding into the environment are the two key properties of an agent. We will now discuss the role of autonomy and of the environment.

Autonomy

There is common agreement in the field that autonomy is a required property of agents (Wooldridge 1999). The agent should be able to act autonomously in its environment. Autonomy means that the agent can act based on its own knowledge and perception. A system lacks autonomy if its behaviour is completely determined by its built-in knowledge so that it need not perceive its environment to decide about its activities. A system is completely autonomous if its behaviour is determined by its own experience (Russell & Norvig 1995, p.35). Autonomous agents have control over their actions and internal state. They cannot be directly manipulated from the outside. Influences on the agent are only possible by performing activities that the agent can perceive.

The role of the environment

A fundamental critique of artificial intelligence is that systems constructed according to its methods are *disembodied* (Dreyfus 1997), i.e., that they are not objects (bodies) embedded in their environment. Their decision making process is independent of their environment, i.e, without interaction with the environment. Multi-agent theory avoids this mistake by regarding the environment as integral part of the framework. Often two classes of environments are distinguished: *artificial and real environments* (Russell & Norvig 1995, p.36). Artificial and real environments have different properties that enforce different kinds of agents with different sensors and effectors. Common to all environments is that they provide percepts to the agent and the agent performs actions in them. Agents that are programs and exist in (artificial) software environments are called *software agents*.
There are several properties to categorize environments (Russell & Norvig 1995):

- *Accessible vs. inaccessible.* An environment is accessible if the agent's sensors can detect its complete state with all relevant information to choose actions.

- *Deterministic vs. nondeterministic.* The environment is deterministic if its next state is completely determined by the current state.

- *Episodic vs. nonepisodic.* An environment is episodic if the result of an action does only depend on the current perception-action cycle of an agent. An agent's action does not affect subsequent perception-action cycles.

- *Static vs. dynamic.* An environment is dynamic if it can change during the deliberation process of the agent.

- *Discrete vs. continuous.* An environment is discrete if there exist a fixed number of possible actions and percepts.

Since in this thesis we construct a computational model, we deal with software agents. The special property of a computational model is that the artificial software environment and the real environment it represents share common properties in the way that the software environment represents some essential features of the real environment. The computational model is a simplification of the nondeterministic, dynamic and continuous real environment, which is deterministic, discrete and static (in the sense stated above).

What is not an agent?

The two key properties of agents allow clarifying the agent concept. They allow distinguishing the agent concept from other approaches in artificial intelligence and computer science.

Objects are not agents (Wooldridge 1999). Agents have some common properties with objects as defined in object-oriented programming (Meyer 1988). Objects have methods by which they are able to act in their environment. Object communicate with other objects by message passing. Objects do not fulfill one key property of agents: autonomy. An object has no control over its own behaviour. Other objects can invoke methods of it, and the object itself has no control whether or not the method is executed or not. Nevertheless, agent architecture can be implemented using object oriented technics.

Expert systems are not agents (Wooldridge 1999). Expert systems are capable of solving problems in some domain based on the knowledge represented in the system (Puppe 1991). But expert systems do not directly interact with their environment. They do not fulfill the second key property of agents: Expert systems are not embedded into their environment.

5.6.2 Reactive vs. deliberative agents

The main criteria distinguishing agent architectures is the question how much internal representation of the world the agents should have. Reactive agents have less or no internal representations whereas agents constructed according to the deliberative approach have only symbolic representations. Figure 5.3 shows both approaches. An agent constructed after a reactive approach purely reacts to its current percepts following condition-action rules. An example of this paradigm is the subsumption architecture (Brooks 1986). Deliberative architectures follow the classical AI approach (the Sense-Plan-Act paradigm (Gat 1997)). The agent plans its actions based on its percepts and knowledge. Both approaches can be combined in hybrid architectures. Hybrid architectures combine reactive and deliberative behaviour in different layers (Gat 1997).

We focus on a deliberative approach, since the activities of the agents based on their internal knowledge are the key elements in understanding reality in a cadastre. The institutional phenomena we need to represent exist only based on the individual beliefs and knowledge of human beings (see chapter 3).

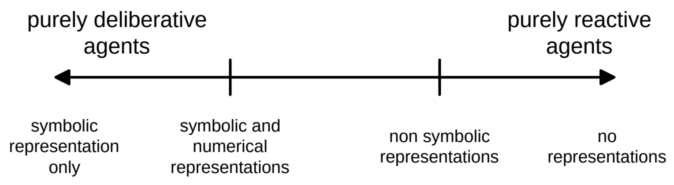
purely deliberative agents
purely reactive agents
symbolic representation only
symbolic and numerical representations
non symbolic representations
no representations

Figure 5.3: Between purely deliberative and purely reactive approaches (after (Ferber 1999))

5.6.3 The agent definition

Adapting the general agent concept, the agent is characterized by its percepts and its actions in the environment. Additionally an internal representation of the environment is necessary, the internal state, which contains the agent's knowledge and beliefs about the world. The state of an agent can be characterized by the percepts its sensors provide, by its internal state and by the actions the agent undertakes. An agent Ag is a tuple of the following form:

$$Ag = (P, I, A) \tag{5.1}$$

where P is a set of percepts, I is the internal state and A is the set of actions. If the internal state is empty ($I = \emptyset$), the agent is called *purely reactive agent*, otherwise it is called *agent with internal state.*

To allow higher level internal capabilities of the agents, such as, planning, goal directed behaviour and collection of experiences, a kind of internal representation of the world is necessary and not possible without internal state. We focus on agents with internal state, since this class of agents is required to realize a deliberative approach.

A goal-based agent

In general, agents with internal state collect information about their percepts and actions and include them into their future decision making process. *Goal-based agents* additionally have explicit representations about desired situations (Russell & Norvig 1995, p.42) which the agent tries to achieve. For a goal-based agent the internal state of the agent I has the following structure:

$$I = B \cup G \tag{5.2}$$

The internal state of the goal-based agent I consists of a set of beliefs about the current and former situations B and a set of goals G, a representation of desired situations which the agent tries to achieve.

Human beings acting in reality in a cadastre have goals and intentions. For the representation goal-based agents are necessary.

Subclasses of goal-based agents

The class of goal-based agents can be further subdivided into:

- Utility-based agents
- Rational agents

Utility-based agents are goal-based agents with a performance measure (Russell & Norvig 1995, p.44). Goals only distinguish desired from undesired situations. A utility function or a performance measure refines this simple distinction. It represents the agent's degree of satisfaction with the achieved situation. If an utility-based agent in every situation does the right thing, i.e., selects always the "best" action with the highest utility, it is called a rational agent (Russell & Norvig 1995, p.33). We will not discuss these classes of agents, because a goal-based agent with internal state is sufficient for the representation of the model.

5.7 The abstract architecture of a multi-agent system

In this section we present an abstract, i.e., domain independent, architecture of a multi-agent system which is suitable for the construction of the model of a cadastre in this thesis. We present an operational view of a multi-agent system, characterizing the architecture according to its operations. The abstract agent architecture of this section combines ideas from (Russell & Norvig 1995), (Ferber 1999) and (Wooldridge 1999).

5.7.1 The definition of a multi-agent system

From the discussion so far follows that a multi-agent system consists of at least two parts: the agent and the environment. There are two possibilities in the definition of the multi-agent system: First the agents can be regarded as part of the environment, or, second, the agent can be defined separated from the environment. We emphasize the embedding of the agent into the environment and consequently assume that the agent is part of the environment.

A multi-agent system is a system that consists of a set of agents that are part of some environment. This notion includes the single-agent scenario as specific case where the environment contains only one agent. Adapting the definition of Ferber (Ferber 1999, p.11) the term 'multi-agent system' refers to a system consisting of the following parts:

- The *environment* E with the following elements:
  - A set of *objects* O. Objects can be perceived, created, destroyed and modified by agents.
  - A set of *agents* Ag (see equation 5.1). Agents are a subset of objects ($Ag \subseteq O$) capable of performing actions - the active entities of the system.
- An assembly of *relations* R which link objects and thus agents to each other.
- A set of *operations* Op enabling the possibility for agents to perceive, manipulate, create, destroy objects of O.
- A set of *operations* U with the task of representing the application of the operations from Op and the reactions of the world to this attempt of modification. The operators from U are called the *laws of the universe.*

The environment E is the set of all objects existing in the system. R defines the static relations between the objects and Op and U define the behaviour of the system.

5.7.2 The structural and the operational part of the architecture

Russell and Norvig distinguish agent architecture and agent program as the main components of an agent (Russell & Norvig 1995, p.35). The agent program is the function that maps the percept to the actions. The agent architecture is the computing device on which the program runs. The architecture describes the structural aspects (the components) of the agent, whereas the agent program represents operational aspects, the agents behaviour. This section does not follow this distinction. We refer to both elements as the agent architecture because it captures both the operations and the components of the whole architecture. The mapping of the percepts to the actions cannot be described independently from the components of the agents. The components of the agent depend on the realization of the mapping function. According to the definition of a multi-agent system, the components of the architecture are represented by the objects of the environment O. The operational aspects of the architecture are represented by the operations of the sets Op and U.

5.7.3 The Sense-Plan-Act paradigm

The classic view of the control system of an agent is, that it should be decomposed into three elements: the *sensing system*, the *planning system* and the *execution system*. This is called the Sense-Plan-Act paradigm (Gat 1997). The control flow between the three components is unidirectional from the sensor to the effector. The execution in such a system is comparable to a computer program. The planner generates a sequence of elementary actions which the system executes.

In this thesis the Sense-Plan-Act paradigm is regarded as the basic concept for the execution model of the multi-agent system. It is an easy and clear concept and it is representable on a computer system without essential problems.

Several issues are known, which are not relevant for the topics of this thesis: The world can change during the planning process so that the plan is not longer applicable if it is generated. Effects of the agent's actions are not completely predictable. So an agent's action can cause the violation of the preconditions for the next action in the plan. The plan can become unexecutable.

This thesis investigates computational models, i.e., artificial agents in artificial software environments, which are deterministic, discrete and static (in the sense introduced in 5.6.1). The issues stated above do not apply for these kinds of environments. The critique applies for artificial systems, which should be embedded into a real environment, which is undeterministic, dynamic and continuous.

5.7.4 The basic operations of the multi-agent system

In the simplest case the multi-agent system consists of at least one agent in the set of objects O. Operations from the set Op that represent the agents activities, must exist. The rules of the universe U must define the reaction of the environment to the agent's actions. The interaction between the agents and the environment determine the dynamics of the multi-agent system, i.e., its behaviour. The essential parts of the architecture are defined by the working together of the agent (operations from the set Op) and reactions of the environment represented by the laws of the universe (operations from the set U). Figure 5.4 shows the basic operations of a multi-agent system. It consist of two parts:

- The activities of the agent (operations from Op).
- The reaction of the environment representing the laws of the universe U.

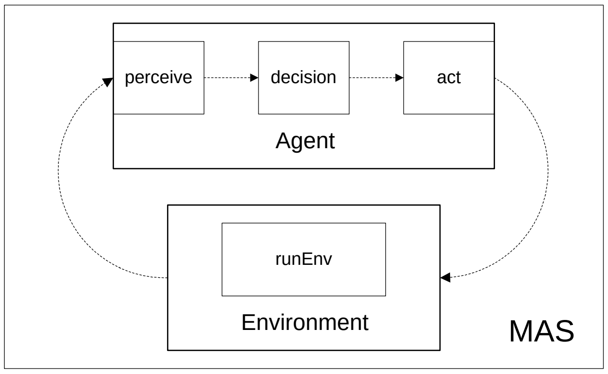

Figure 5.4: The basic operations of a multi-agent system

Following the Sense-Plan-Act paradigm, the activity process of the agent can be divided into three components: the perception subprocess, the decision making subprocess and the action subprocess. An agent can be described by a function *perceive*, a function *decision* and a function *act*.

The perception process

$$perceive : E \rightarrow P \tag{5.3}$$

The function *perceive* represents the perception process of the agent. It maps the environment E to the set of percepts P of the agent. The realization of the function *decision* representing the decision making process of the agent depends on the selected agent architecture, i.e., it is different for purely reactive agents and agents with internal state.

The decision process of a purely reactive agent

A purely reactive agent is characterized by the fact that it directly maps input to output, i.e., percepts to actions. The function *decision* of the reactive agent is a function of the following type:

$$decision : P \rightarrow A \tag{5.4}$$

It transforms a set of percepts P into actions A. A thermostat, for instance, can be modelled by a reactive agent where the *decision* function has the following form:

$$decision(t) = \begin{cases} HeaterOn & \text{if } t < 18 \\ HeaterOFF & \text{if } t > 22 \end{cases}$$

If the temperature is below 18 degree, the heater will be switched on and above 22 degree it will be switched off.

The decision process of an agent with internal state

For agents with internal state the decision function has a more complex form. It includes the built in knowledge and former experiences of the agent into the decision making process.

$$decision : P \times I \rightarrow A \tag{5.5}$$

The *decision* function maps a set of percepts P and the current internal state I of the agent into a set of actions A. The decision function consist of three steps. The first step (the function *updStatePercepts*) updates the internal state of the agent based on its percepts.

$$updStatePercepts : P \times I \rightarrow I \tag{5.6}$$

The second step (function *selActs*) selects actions based on the updated internal state.

$$selActs : I \rightarrow A \tag{5.7}$$

If the agent should improve its knowledge based on its own selected actions, the internal state of the agent will be again updated afterwards based on the agent's selected actions (the third step).

$$updStateActions : A \times I \rightarrow I \tag{5.8}$$

The function *updStateActions* maps the agent's actions and the internal state of the agent to a new version of the internal state. Including the update of the internal state based on the agent's actions, the decision making process has the following form:

$$decision : P \times I \rightarrow (A, I) \tag{5.9}$$

Figure 5.5 shows the decision making process of an agent with internal state.

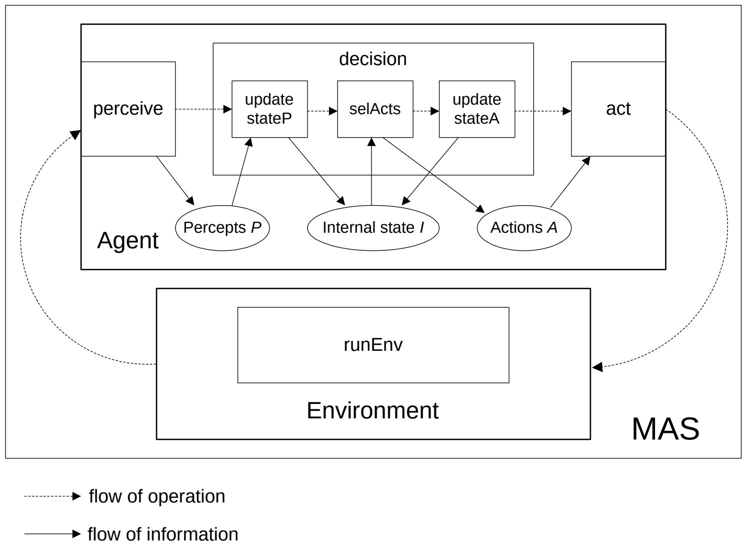

Figure 5.5: Agents with internal state

The action process

The function *act* represents the performance of the actions by the agents. It conveys the agent's actions to the environment.

$$act : E \times A \rightarrow E \tag{5.10}$$

The reaction of the environment

The function *runEnv* represents the reaction of the environment, i.e., the laws of the universe.

$$runEnv : E \rightarrow E \tag{5.11}$$

It maps the environment based on the actions performed by the agents to a new state of the environment. This mapping function realizes the changes on objects (including agents) caused by the agents' actions.

5.8 Communication between agents

Agents can interact through explicit linguistic actions (communication) or by nonlinguistic (physical) actions modifying the world in which they act (Moulin & Chaib-Draa 1996). Communication allows the agents to exchange information and to coordinate their activities. The fundamental requirement for agents to be able to interact on a high level is communication. Communication is required to form a group out of the single agents. Communication protocols and languages are necessary to construct multi-agent systems.

General purpose communication languages have been developed, for instance, the knowledge query and manipulation language (KQML) and the knowledge interchange format (KIF) which are related (Finin, Labrou & Mayfield 1997). They are not directly based on speech act theory (see subsection 3.8.1) but are influenced by the idea of regarding communication as action. They provide the possibility to realize high level communication between agents.

Agents can communicate by exchanging information following two major strategies. Agents can directly exchange messages or exchange messages over a data repository shared by all agents of the system. Direct message exchange is called *message passing*. Indirect communication architectures are designated as *blackboard architectures*, because the common data repository for indirect communication is called a *blackboard* (Ferber 1999, p.129). Most agent communication protocols are based on speech acts (Wooldridge & Jennings 1995). Speech act theory provides the foundation for high level communication. Speech acts can be represented and exchanged between agents with a message exchange architecture.

In this thesis we focus on (direct) message exchange, respectively message passing, which we select as communication method for the model. The message exchange architecture is based on the classical *theory of communication* by *Shannon and Weaver* (Shannon & Weaver 1948). In this model the communication act consist of the sending of some information from a *sender* to a *receiver*. On the sender side the information is encoded using a *language* and decoded on the receiver side. The information is sent through a *medium*, called the *channel*. Figure 5.6 shows the communication model.

In this thesis we select a communication method based on speech act theory which is realized by a message exchange architecture.

5.9 Summary

In this chapter we discussed multi-agent theory as the framework for representing reality in a cadastre. We introduced the main concepts and approaches of the field.

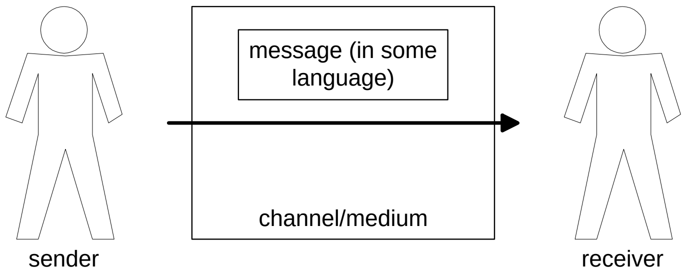

Figure 5.6: The communication model (after (Shannon & Weaver 1948))

We gave a general overview of the field but focused on the topics important for this thesis.

The approach used in this thesis we called the *agent-based conceptualization approach*. This approach regards the agent paradigm as purely conceptual framework, independent of representation and implementation issues and not as technology.

For the programming of multi-agent systems we distinguished specific *agent programming languages* from general purpose languages which can be applied to the development of agent-based systems.

The chapter gave definitions for the terms 'agent' and 'multi-agent system' suitable for the task of this thesis. An *abstract agent architecture* was introduced as foundation for modelling reality in a cadastre. We discussed communication between agents and introduced the *message exchange model* as the basic paradigm for communication between agents in this thesis.

The chapter introduced the basic ideas of simulation and in particular the approach of agent-based simulation. Major contribution of this chapter is that it provides the conceptual framework for the construction of the simulation model. It introduced the agent concept as the central element in the model representing human intentions and behaviour. We developed a general framework that allows the development of an agent-based simulation model of reality in a cadastre in the chapters 7 and 8.

Chapter 6

Algebraic specifications in the functional language Haskell

6.1 Introduction

According to the division of multi-agent research into agent theories, agent architectures and agent languages (Wooldridge & Jennings 1995), this chapter discusses the third point. It discusses the agent programming language used in this thesis. It introduces an algebraic specification methodology represented in the functional programming language Haskell for this purpose. The key idea is that we assume that reality can be modeled in terms of algebras. Haskell provides the tools to algebraically specify and implement the computational model.

The starting point is the idea that the process of developing a multi-agent system can be regarded as a software engineering enterprise (Wooldridge 1997). A multi-agent system can be regarded as a specific software. Methods from the software specification field can be applied to agent-based systems. The software specification framework must be capable of capturing at least the following aspects of an agent-based system (Wooldridge 1997):

- the beliefs agents have,
- the ongoing interaction agents have with their environment,
- the goals the agent will try to achieve,
- the actions that agents perform and the effects of these actions.

Chapter 7 shows that the algebraic specification framework can be successfully applied to the construction of the agent-based model of reality in a cadastre that it is capable of representing the four aspects. This chapter introduces the conceptual and formal background necessary to understand the specification of the model.

The use of functional specifications is new to the field of multi-agent theory. Only Frank (Frank 2000, Frank, Bittner & Raubal to appear) recently used a functional framework for the specification of an agent-based model.

This chapter starts with the discussion of the advantages of formal specification methods for the construction of computational models. It introduces algebraic specifications on both the formal and the intuitive level and shows how to represent an algebraic specification in a functional language. Section 6.4 gives the core argument of this chapter answering the question why algebraic specifications implemented in the functional language Haskell are appropriate for the construction of agent-based models of some aspects of the real world. The chapter ends with an overview of the syntax of the functional programming language Haskell.

6.2 Formal specifications

Software development is the process of looking for a process that establishes that a program correctly implements a concept that exists in someone's mind (Liskov & Zilles 1978). The program that correctly implements the concept can be regarded as a computational model of that concept. If the concept represents some part of reality, the process of developing the program is the construction of a computational model of this part of reality. We regard the construction of an agent-based computational model as a software development process and argue in this section that formal specifications are advantageous for this purpose.

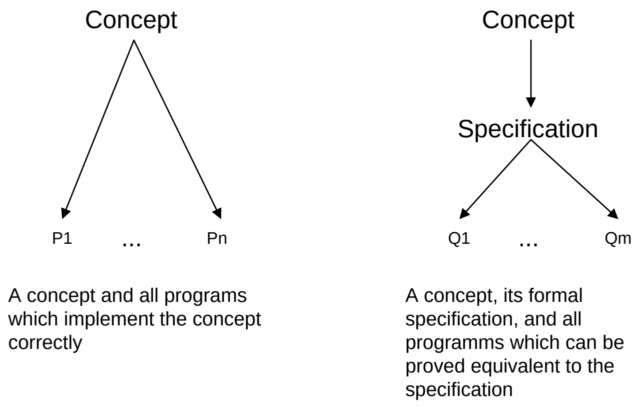
Concept
P1
...
Pn
A concept and all programs which implement the concept correctly
Concept
Specification
Q1
...
Qm
A concept, its formal specification, and all programms which can be proved equivalent to the specification

Figure 6.1: Concept, specification and programs (after (Liskov & Zilles 1978))

A concept can usually be implemented by many programs. Formal specifications are established between the informal concept and the program (see figure 6.1). They provide a mathematical description of the concept to be implemented. The key advantage is that the correctness of a program can be proved with respect to the formal specification, i.e., is can be proved that a class of programs satisfies the formal specification. The formal description of the concept in the formal specification provides the advantage that it is easier to understand than the program. The specification written in a formal language with mathematically defined semantics allows a clearer and more exact description of the concept.

The methodology can fail in two ways (Liskov & Zilles 1978): First, the proof itself can be incorrect, i.e., it can state a program as correct with respect to the specification, which is in fact not. Second, the specification can be incorrect with respect to the concept, i.e., the specification does not correctly express the meaning of the concept. The first issue can be solved by better proof methods. The second issue inherently remains because there is no formal way to prove that a specification correctly represents a concept that exists in someone's mind, respectively that the specification correctly represents some part of reality.

Frank and Kuhn (Frank & Kuhn 1995) formulate requirements for formal specification languages, which hold for computational models on a formal foundation too:

- *expressing semantics.* The specification language must be able to express complex real world situations in terms with exact defined semantics. The semantics, i.e., the meaning of the specification, must provide a clear mathematical description of the concepts to be represented.
- *ease of understanding.* In order to communicate the concepts to be represented, i.e., to serve as a communication medium, a specification must be easy to read and to write. It should support mastering real world situations and provide abstraction mechanisms which are easy to understand.
- *rapid prototyping.* A specification language with a rapid prototyping capability allows the construction of executable specifications. This allows observing the behaviour of the specification and to detect deviations form the intended behaviour. The correspondence of the specification with the intended real world concepts cannot be formally proven. A rapid prototyping capability of the specification is the way to deal with this issue.

Rapid prototyping is the key property making formal specifications applicable for the construction of computational models.

6.3 Algebraic specifications

This section introduces *algebraic specifications* (Guttag, Horowitz & Musser 1978, Liskov & Zilles 1978, Loeckx et al. 1996, Breu 1991, Horebeek & Levi 1989, Ehrlich et al. 1989, Sannella 1997) as methodology for writing formal specifications. Roughly speaking an algebra is a collection of carrier sets and operations between theses sets. Algebraic specifications allow specifying algebras. Strictly speaking algebraic specifications are language independent. But they require a formal language which is capable of expressing algebraic style specifications. Algebraic specifications are a style to provide a mathematical description of concepts representing ideas in someone's mind or parts of the real world in a abstract manner.

This thesis uses the algebraic specification style for the representation of the agent-based model. We assume that algebras and the real world have structural similarities. Algebraic models are therefore relevant for the construction of models of reality.

6.3.1 Informal description

There are two fundamental design strategies in the software design process (Guttag et al. 1978): A top-down strategy is the transformation of a software design by a step-by-step refinement process into an executable program. The design of data types is the complementary design strategy. It treats operations that primarily execute on a single data type as unit. A data type specification (*Abstract Data Type*) is a representation independent formal definition of each operation of a data type (Guttag et al. 1978). An Abstract Data Type is a class of data structures described by its available operations and their properties (Horebeek & Levi 1989).

An algebra is an abstract mathematical structure consisting of a family of sets of objects (the *sorts*) and a number of *functions*. A function is a mapping from a cross product of values (the *domain* of the function) to a single value (the *range* of the function) (Liskov & Guttag 1986). Domain and range have values from the set of sorts. If an algebra contains more than one sort, it is called *many sorted* or heterogeneous. Algebras with only one sort are named single sorted or homogeneous. A function of arity 0 is named a constant. There exist two kinds of functions: *observers* and *constructors*. Constructors return values of the sort being specified, observers map values of that sort onto other sorts (Liskov & Guttag 1986). Constructors construct or change objects whereas observers report the state of an object.

An algebraic specification is a mathematical description of an Abstract Data Type where Abstract Data Types are modeled by means of algebra. It is the description (notation) of a many sorted algebra (Horebeek & Levi 1989). An algebraic specification consist of a syntactic and a semantic part. The syntactic part contains the set S of sorts, the set O of operations applicable to elements of S. The semantic part comprises the set E of axioms defining the behaviour of the operations from the set O (Ehrlich et al. 1989).

6.3.2 Formal description

A *signature* of an algebraic specification consists of the set of sorts of the participating data and the set of operators declared on the sorts. A signature is a pair

$$\Sigma = (S, O)$$

with S the set of sorts and O the set of operators. Each $o \in O$ has the form:

$$o : s_1 \times \ldots \times s_n \rightarrow s$$

with $s_1, \ldots s_n, s \in S$. In the case $n = 0$, o is called a constant.

An *algebraic specification* is a triple $D = (\Sigma, X, E)$ where Σ is a signature, X is a set of variables and E is a set of axioms. We omit in the notation the set of variables assuming that it contains exactly the variables of E. We make the signature explicit and write:

$$D = (S, O, E)$$

An algebraic specification is a triple consisting of the set S of sorts, the set O of operators and the set E of axioms.

The models of algebraic specifications are *heterogeneous algebras* with the given signature Σ, which fulfill the given axioms E. Algebras are interpretations of signatures, which assign to each sort S of Σ a carrier set and to each $o \in O$ a function on the given carrier set. Algebraic specifications specify classes of algebras.

6.3.3 Examples

We give two examples for algebraic specifications, the specification of a stack and the specification of a two-dimensional vector.

A stack contains elements of a particular type. It implements the 'last in first out' principle, i.e., an element can be pushed on top of the stack and only the element on top of the stack is removable by the 'pop' operation. The element which comes last into the stack comes first out of the stack. The operation 'top' returns the top element of the stack. The following specification represents a stack:

```
sorts: Stack, Nat
operations: empty : -> Stack
            zero : -> Nat
            push : Nat x Stack -> Stack
            pop : Stack -> Stack
            top : Stack -> Nat
axioms: top(push n s) = n     (1)
        pop(push n s) = s     (2)
        top(empty) = zero     (3)
        pop(empty) = empty    (4)
```

According to the formal notion of the previous subsection this defines an algebraic specification $D = (S, O, E)$ with $S = \{Stack, Nat\}$, $O = \{empty, zero, pop, push\}$ and E is the set of the equations (1)-(4).

A two-dimensional vector is a tuple of two elements. On a vector the following operations are defined: *make* constructs a vector out of two single elements. *getX* gives the first component of a vector and *getY* returns the second element of a vector. *add* and *sub* represent vector addition and subtraction.

```
sorts: Vec,Float
operations: make : Float x Float -> Vec
            getX : Vec -> Float
            getY : Vec -> Float
            add : Vec x Vec -> Vec
            sub : Vec x Vec -> Vec
axioms: add a, b = make (getX a + getX b) (getY a + getY b) (1)
        sub a, b = make (getX a - getX b) (getY a - getY b) (2)
```

The algebraic specification is defined by $D = (S, O, E)$ with $S = \{Vec, Float\}$,$O = \{make, getX, getY, add, sub\}$ and E consist of the equations (1) and (2).

6.4 Computational models of reality in Haskell

Haskell (Thompson 1996, Bird & Wadler 1988) is a functional programming language that supports an algebraic specification style. We can construct algebraic specifications and, since Haskell is a formal language, a Haskell program correctly implements the algebra specified, i.e., is an algebraic model. This means that the computational model constructed has the properties of the algebraic specification, which is intended to represent some aspects of the real world.

The key point is that we assume that algebras represent the structure of the real world on an abstract level. An algebraic specification expresses the structural properties of some part of the real world in question and the Haskell programming language allows constructing programs out of the specification, which form algebras that have the structural properties specified, i.e., are similar to the real world. Figure 6.2 shows the relationship between reality, algebraic specifications and computational models analogous to the connection between concepts, specifications and programs given in section 6.2.

The Haskell program can be tested, i.e., its correspondence to the part of reality it represents can be checked. This is exactly the approach of simulation that compares the output of programs, i.e., simulations, with data observed in reality. If both correspond, the model, respectively the program, is correct.

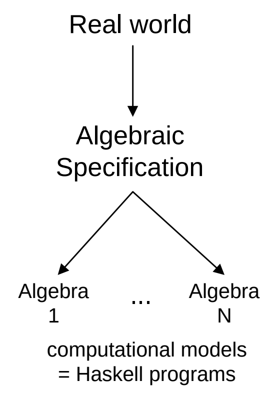

Figure 6.2: Computational models of reality in Haskell

For the special case of agent-based models algebraic specifications support the bottom-up strategy (see subsection 6.3.1), which is characteristic for multi-agent systems (see subsection 5.3.2). Algebraic specifications regard the operations on the object level as primary. They allow the construction of models in terms of the individual objects (the agents) and their operations.

6.5 Specification, Representation and Implementation

A computational model in Haskell consists of three parts: the specification, the representation and the implementation. Specifications define the signatures of operations and representation independent properties of these operations, i.e., they are the realization of algebraic specifications in Haskell. The representation describes the particular carrier sets on which the operations specified are carried out, i.e., it assigns carrier sets to the sorts of the algebraic specification. Implementations define the functions that correspond to the operations defined by the algebraic specification. They are carried out on a particular data type, i.e., on a particular representation.

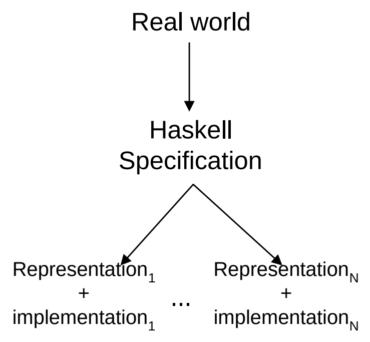
Real world
Haskell Specification
Representation₁ + implementation₁
...
Representation$_N$ + implementation$_N$

Figure 6.3: Specification, representation and implementation

The three concepts together describe how to connect specifications and computational models. The specification part defines an algebraic specification, and representation and implementation represent a particular algebra (or program) that is correct with respect to the specification (see figure 6.3). The model construction process in Haskell can be described as a three-stage process: first construct the algebraic specification of some part of reality, then define representations and implementations to realize a computational model that corresponds to the specification.

6.6 Haskell

This section introduces the main concepts as well as the syntax of Haskell (Thompson 1996, Bird & Wadler 1988) as far as necessary to understand the computational model of this thesis.

6.6.1 Properties of Haskell

In a functional programming language everything is a function that returns a value. Programming in a functional language consists of building function definitions and using the computer to evaluate the expressions (Bird & Wadler 1988). The basic control structure in a functional program is recursion. One distinguishes pure functional languages from others. Pure functional languages allow no side effects, i.e., the only effect of a function onto the whole program is its result.

If arguments in a functional programming language are only evaluated if their value is needed than this language is said to be lazy. Otherwise it is called strict. Haskell is a lazy functional language.

Referential transparency is an important property of Haskell. It means that an expression always describes the same value. The value of an expression is independent of any context. Expressions denoting the same value can be replaced by each other. This allows mathematical reasoning based on substitution. Haskell supports a strong type system. In a strongly typed language every expression has a particular type. The compiler checks that only expressions of the correct type are applied to an object. Types need not always be provided by the programmer. Functional languages support a type inference system that derives the type of an expression if no or little type information is given explicitly (for instance, from the expression $x = 1 + 1$ the type inference mechanism can deduce that x is of type Int). The strong type system and referential transparency simplify the writing of programs avoiding sources of mistakes due to type conflicts and side effects.

The most obvious disadvantage is that functional specifications restrict the axioms to a constructive form. General properties, such as transitivity cannot be expressed by constructive axioms. Constructive axioms restrict the left hand side of an axiom to be a simple expression or an expression containing only constructor operations. Nevertheless, functional languages have high expressive power for the application to a broad variety of problems. The restriction to constructive axioms is the precondition to reach the executability of the specification, i.e., it is the price which must be paid in order to enable the construction of computational models.

6.6.2 An example

Algebraic specifications can be translated into Haskell (Frank 1997*a*). The example shows the functional specification corresponding to the algebraic specification of a two dimensional vector presented in subsection 6.3.3.

```
class Vector vec where
    make :: Float → Float → vec
    getX, getY :: vec → Float
    add, sub :: vec → vec → vec

    add a b = make ( getX a + getX b) (getY a + getY b)
    sub a b = make ( getX a − getX b) (getY a − getY b)
```

The specification provides a representation independent description of the properties of the sort *vec* representing the two dimensional vector. The algebraic specification corresponds to a class in Haskell.

6.6.3 The syntax of Haskell

In this thesis a specific implementation of the Haskell standard is used: *Hugs* (Jones & Peterson 1999). This subsection introduces the basics of the Haskell syntax which are necessary to understand the specification done in this thesis.

Functions

The syntax of functions in Haskell deviates from the usual mathematical syntax. An expression of the form

```
    f :: a → b → c
```

describes a function with the signature:

$$f : a \rightarrow (b \rightarrow c) \tag{6.1}$$

which does not correspond to the definition of a function we used earlier in this chapter (see subsection 6.3.1) with the following signature:

$$f : (a \times b) \rightarrow c \tag{6.2}$$

The process of transforming a function with a structured argument list into a sequence of functions with a single argument is known as *currying* (Bird & Wadler 1988). Both styles of definitions have their advantages. In Haskell typically the curryed style of definition is used (as in equation 6.1). The following example shows the curryed and the uncurryed version of the function *min*:

```
    min x y = if x ≤ y the x else y −−curryed version
    min (x,y) = if x ≤ y the x else y −−uncurryed version
```

Expressions defining functions must start with a lower case letter. New functions can be designed by conditional equations or by pattern matching as the following examples of the factorial function (equation 6.3) shows:

```
    fac n = if n=0 then 1 else n∗fac(n−1)

    fac 0 = 1
    fac n = n∗fac(n−1)
```

Both definitions are equivalent. They define the following function:

$$fac(n) = \begin{cases} 1 & if\, n = 0 \\ n \times fac(n-1) & otherwise \end{cases} \tag{6.3}$$

Using pattern matching the compiler always applies the first matching equation for the evaluation of an expression. The pattern matching style allows writing in a more mathematical and clearer style.

Higher order functions

A source of the expressive power of a functional language is the possibility of higher order functions. The arguments of a function can be themselves functions, and functions can be the result of other functions. Higher order functions provide a powerful representation mechanism allowing that functions are handled in the same way as other values.

Important examples of higher order functions are the functions *map*, *filter* and *foldl*:

map :: $(a \to b) \to [a] \to [b]$
map sqrt [1,4,9] = [1,2,3]

The function *map* applies a function of type $(a \to b)$ to a list of values of type a. The result is a list containing values of type b, the results of applying the function to the elements of the input list.

filter :: ($a \to$ **Bool**) $\to [a] \to [a]$
filter (**even**) [1,2,3,4] = [2,4]

The function *filter* applies a boolean function of type $(a \to b)$ to a list of values of type a. The result is a list of values of type a which contains only the values from the input list, which evaluate the input function to *True*.

foldl :: ($a \to b \to a) \to a \to [b] \to a$
foldl (+) 1 [1,1,1] = 4

The operation *foldl* applies the operation $(a \to b \to a)$ to the elements of the list $[b]$ from the left hand side. The second argument is the result of the application of the operation to the previous value of the list. The computation starts with the initial value given by argument a. A similar operation *foldr* works starting with the most right element of the list.

Data types and data type constructors

Haskell supports the basic data types: integer (*Int*,*Integer*), boolean (*Bool*), floating point numbers (*Float*,*Double*) and character (*Char*). The expression $[a]$ denotes a list of values of type a. The data type *String* is defined as list of characters ([*Char*]). A product type (tuple) is written by a comma separated series of types enclosed in parenthesis. The expression (*Int*, *Bool*) defines a tuple consisting of a value of type integer and a value of type boolean. The keyword *type* defines a type synonym, i.e., an alias for an existing data type (e.g.: *type ID = Int*). User defined data types are declared by the keyword *data* together with a type constructor. A type constructor is a function that constructs new data types out of existing data types. For example, the expression

data *Book* = *B* **String String**

defines a new data type *Book* with the constructor function *B* applied to two string values (denoting title and author). Constructors are written in capital letters. A sum type is represented by a '|' separated series of values:

```
data Week = Mon | Tue | Wed | Thu | Fri | Sat | Sun
```

Classes, instances, and data representations

Polymorphism is the capability of an operation to be applied to arguments of varying types. This allows the translation of algebraic specifications into classes in a functional language. Classes are the counterpart of algebraic specifications in a functional language.

A class consists of a list of operations applied to arguments of specific types. The class definition consists of the signatures of the operations and of the axioms constraining the behaviour of the operations. Classes are parameterized with the types they define. A class is a collection of types over which the operations are defined. An example of a class definition was presented in subsection 6.6.2.

Each member data type of a class is called an instance of that class. An instance combines a class with a data type. It provides the definition of the operations of the signature of the class. An instance defines the operations for one particular data type. For the vector example of subsection 6.6.2 we define the instance for the type *Coord*:

```
data Coord = Coord Float Float

instance Vec Coord where
  make x y = Coord x y
  getX (Coord x y) = x
  getY (Coord x y) = y
```

Class, *data* and *instance* correspond to the concepts of specification, representation and implementation as introduced in section 6.5. Figure 6.4 shows the ideas in correspondence to figure 6.3.

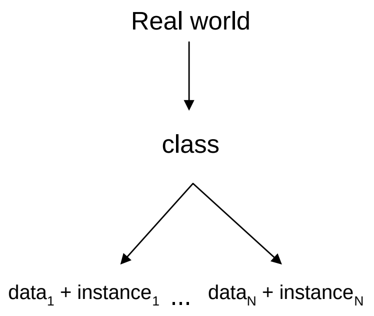

Figure 6.4: Class, data and instance

The lifting of functions

The lifting of functions transfers an operation working with one data type into an operation working with different data types.

```
type Time = Int
data Times f = Timing f Time deriving Show
op :: Int → Int
op a = a*a
```

```
lift   :: ( t → t)  → (Times t → Times t)
lift  f (Timing x t) = Timing (f x) (t+1)
myop a = lift (op) a
```

For instance, we can define an operation *op* which works with integer values. We can lift (function *lift*) the operation to work on the data type *Times* without changing the operation itself. We define the operation *myop* working on the lifted data type using the operation *op*. In the example the intended interpretation is to add a time the operation takes into the result, to evaluate the time the operation takes.

6.7 Summary

This chapter dealt with the agent programming language used in this thesis. We introduced the language and the methodology to represent the agent-based model of this thesis. The construction of a multi-agent model can be seen as a software development process. A widely used technique for the software design is algebraic specification which we described as methodology for specifying the agent-based model of this thesis. As representation language we introduced the functional programming language Haskell which supports an algebraic specification style.

The core idea is that an algebraic specification style expressed in a functional language provides a sophisticated specification framework for agent-based models. An algebraic specification describes algebras. We assume that algebras capture real world structures on an abstract level, which makes them suitable for models of reality.

We introduced the model construction process as a three stages process consisting of specification, representation and implementation. Specifications correspond to *class* definitions, representations to *data* type definitions and implementations correspond to *instance* definitions.

This chapter introduced the formal background necessary for the representation of the agent-based model of reality in a cadastre. In the next chapter we construct the model using the representation mechanism introduced here.

Chapter 7

The construction of the agent-based model

7.1 Introduction

Chapter 4 analyzed the structure of reality in a cadastre and developed an ontology of the domain. In this chapter a computational model of reality in a cadastre will be constructed based on the ideas introduced in chapter 5 and 6. Reality in a cadastre is mainly influenced by institutional concepts. People act according to theses institutional concepts. The main issue for the task of this chapter is the representation of human intentions and behaviour because institutional reality does not exist independent of human beings and their minds. We use agents for the representation of human beings and their minds.

The chapter emphasizes the description of the conceptual ideas. It follows the agent-based conceptualization approach as basic assumption (see section 5.2). It does not focus on the technical issues of the representation mechanism, it focuses on the conceptual framework. It uses a formal language, Haskell, to express the conceptual ideas in an algebraic specification style (see chapter 6). For the development of the model we follow the three stages process (see section 6.5) and introduce the specifications, the representations and implementation details as far as necessary. We introduce the operations and data structures necessary for the simulation in chapter 8.

The model of this chapter can be regarded as ontology in the computer science sense (see section 3.2), which is a language (Haskell) dependent conceptualization of the domain cadastre. The model construction process is based on the following architectural assumptions with the goal to achieve a most simple framework:

1. The architecture can be characterized as an instance of the Sense-Plan-Act approach (see subsection 5.7.3). We do not focus on the most human like reasoning process. We use this approach as it is sufficient to represent reality in a cadastre.

2. We apply an agent-based framework to represent human beliefs about assigned status functions in the internal state of the agent. According to the general agent architectures described in section 5.7 we use a goal-based agent architecture with internal state.

To apply the abstract architecture (see section 5.7) to the model construction process, we have to perform two tasks:

1. Find realizations for the operations from Op and U.

2. Identify the data structures representing the elements of the environment E.

The operations from the set Op implement rules on the individual level, in particular the decision making process of the agents including constitutive rules. Operations from the laws of the universe U represent rules on the world level that correspond to change which is caused by the whole set of agents, i.e., by the combination of all activities of the agents.

Data structures have to be found for agents (we regard the agent's actions and beliefs as part of the agent) and land pieces, which represent the elements of the environment E.

We start with the discussion of the assumptions about the model we make without loss of generality. We discuss the conceptual ideas of the model, which are: the realization of documentation, the realization of facts and rules in terms of data structures and operations, the agent interaction model based on message exchange and the goal generation process, which determines the dynamics of the model. Next we introduce the model following the three stages structure of the model construction process. We define sorts for the elements of the environement E and specify the operations on these sort (from Op and U). Then we introduce the representations (i.e., data structures corresponding to the sorts) for E and third discuss the implementations of the operations from Op and U.

7.2 Assumptions about the model

This section discusses the basic assumptions we make about the model. These assumptions restrict the complexity of the domain without loss of generality. We make these assumptions to achieve a minimal complexity of the model. We claim that these assumptions do not influence the core of reality in a cadastre that we want to represent. There is no obstacle and limitation of the model that avoids the extension of the model to a more comprehensive part of reality.

We regard communication actions as primitives. Strictly speaking communication actions are speech acts which are themselves institutional facts (see subsection 3.8.1). According to Searle, speech acts are the most basic facts that are essentially constitutive for every other institutional fact. For the purpose of the model strictly speaking communication actions have to be represented capturing their physical and institutional part. We regard communication actions as primitives in the same way as physical actions. This does not affect the structure of institutional facts in cadastral reality because this structure is based on the common foundation of language. The institutional facts discussed here are on a higher level of institutional reality. Without loss of generality communication actions can be described as actions on the physical level, on which the structure of institutional reality is grounded in a cadastre.

According to the abstract architecture the agent is characterized by the perception decision and action process (see equations 5.3, 5.9 and 5.10). We assume that according to the laws of the universe the environment provides the percepts to the agent. Consequently be do not have an explicit perception operation. We do not distinguish between the decision about the actions and the execution of the actions. Thus there is no explicit action function.

7.3 Documentation in the model

The documentation in the cadastral registry plays an important role for the cadastral system because it permanently represents the events that change the institutional status of objects and subjects (see section 4.5). We distinguish documentation of the legal situation from the documentation of transactions. Documentation is represented in the model by the knowledge of agents, i.e., the documentation of the legal situation and of the transactions is completely realized within the internal state of the agents.

Since we assume the agent representing the cadastral registry has complete knowledge of the institutional situation (see section 3.5), the current knowledge of the registry agent represents the currently existing institutional facts, i.e., the current legal status of land pieces or human beings which count as parcels or owners of parcels. Status beliefs refer to the actions that created the beliefs, and thus store information about legal transactions.

In reality the cadastral registry maintains the set of legal transactions and their legal consequences. The legal transactions are caused by communication actions, i.e., by speech acts. Thus in the model we can represent the cadastral registry by a set of speech acts and by the institutional status, which is assigned to the speech acts. The agent representing the cadastral registry maintains beliefs, which consist of speech acts (represented by messages) and status assigned to the speech acts.

For agents who do not have complete knowledge about the current legal situation, copies from the cadastral registry, i.e., messages from the registry agent, count as proofs of events, i.e., of legal transactions. For instance, an owner has to prove his ownership if he wants to sue another person constraining his right. The court responsible for the complaint does not have complete knowledge about owners. In this case a message from the cadastral registry proofs the right of the owner.

7.4 The components of the model

Applying the ontology of reality in a cadastre (see section 4.11), the structure of the system corresponds to physical phenomena, i.e., to the categories of physical phenomena that exist in the domain in question. According to the ontological categories of phenomena (see subsection 4.8), within the category of events there are communication events and physical events (caused by physical and communication actions of the agents). Within the category of objects there are land pieces and the system of documentation and within the category of subjects there are agents. Since we assume that the system of documentation is realized within the internal state of one registry agent, no explicit representation of documentation is necessary.

The structure of the agent-based model for representing reality in a cadastre is determined by the following elements:

- Agents
- Communication actions between agents
- Physical actions of agents
- Land

The agents' internal state represents the building blocks of institutional reality, i.e., status beliefs, rights and constitutive rules.

7.5 Facts and rules vs. data structures and operations

According to the distinction of facts and rules for their creation and existence (see subsection 4.10), facts describe what is in the real world and rules describe how the world possibly evolves. Rules describe how facts can be created and destroyed. Facts represent a state of the world (current, former or future state) whereas rules represent how the world can change between these states. With respect to the two levels of reality the world changes on the physical and on the institutional level. Physical change affects the existence and properties of physical phenomena, institutional change affects only a particular property of human beings, their mind. For the model we distinguish general change of the world state from internal change of the agents minds. Strictly speaking the change affecting the mind of the agent is a specific kind of physical change. For the model there is no qualitative difference between both.

Facts, i.e., phenomena and beliefs in the agent's minds determine the state of the world. In the model *we represent the state of the world by data structures.* Data structures comprise agents, beliefs of the agents, actions and land pieces.

In the model *we represent the rules for the change of the world state by operations.* We distinguish rules that define possible change on the individual level from rules that define possible change on the world level. Rules on the individual level change individuals whereas rules on the world level affect the whole world. The class of rules of the first class, we are especially interested in, are constitutive rules. We implement constitutive rules as operations updating the internal state of the agent.

7.6 Agent interaction

In reality of a cadastre persons perform physical and communication actions. In this thesis an agent interaction model based on a (direct) message exchange (see section 5.8) is selected. Interaction between agents is realized in this approach by the exchange of information pieces codified in specific formats. These information pieces are messages. We distinguish two kinds of messages: communication messages and physical messages. Figure 7.1 shows the transition from reality into the framework of the message exchange model. In reality there are human beings performing and perceiving actions. In the model there are agents which exchange messages with each other and with other parts of the environment.

Accordingly agents can perceive communication and physical events, i.e., they can receive communication messages and physical perception messages. We assume that physical and communication actions as well as physical and communication percepts realized in a message exchange approach represent all aspects of agent interaction in the model.

7.7 Goal generation based on rights

A goal-based agent performs actions to reach its goals. The generation of these goals in general is a two stages process. First the agent needs to investigate all its possibilities to act. From these possibilities the agent selects the situations it tries to achieve, i.e., its goals. Based on these goals the agent selects its activities. The possibilities of the

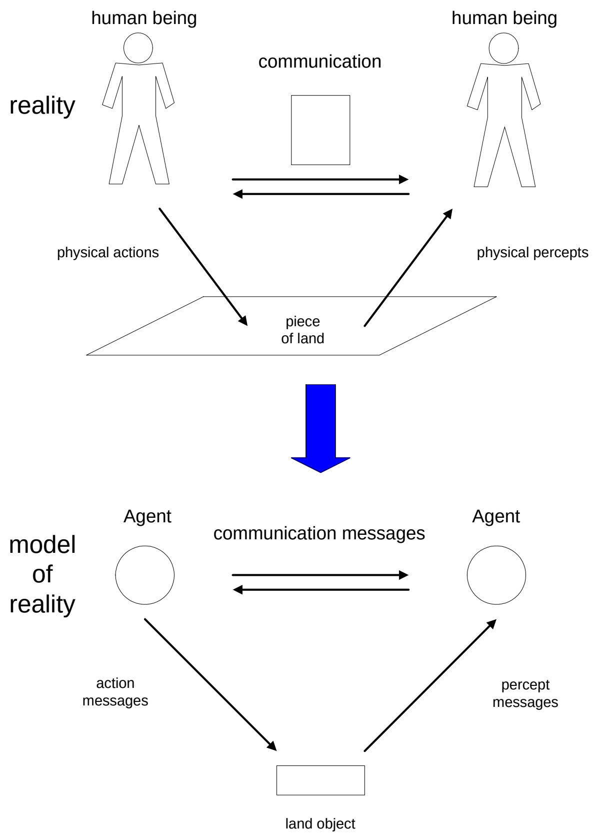

Figure 7.1: From reality to the agent-based model

agent are determined by its physical capabilities and its rights. This does not mean that the agent must select goals corresponding to its rights, in principle it can also decide to violate its rights.

Rights describe the conventional power connected to status functions. They give rules for possible activities on the institutional level (see section 3.6). In section 4.6 we identified two types of rights relevant for the model of a cadastre: duties and powers (in the legal sense). Powers (in the legal sense) define which activities are allowed. Duties define what activities are required. For instance, the power (in the legal sense) to use the land is connected to ownership, as well as the duty to pay taxes. In general the agent is able to freely choose activities according to its powers (in the legal sense). In opposition duties define the obligation to perform particular activities.

The execution of activities according to powers (in the legal sense) creates duties (see section 4.6). After an agent decides to perform a particular activity according to its powers (in the legal sense) the resulting social processes in a cadastre are highly determined by the duties of the participants. For instance, if an owner offers his parcel and another person accepts his offer, the owner afterwards has the duty to transfer ownership of the parcel to the buyer, i.e., the duty to apply the ownership transfer at the cadastral registry. The judge at the registry has now the duty to perform the ownership transfer.

Duties can be modeled as a specific class of goals, because they represent situations the agent tries to achieve. To describe social processes in a cadastre the modelling of goal generation based on duties is a core element.

Agents generate goals based on powers (in the legal sense). The agent chooses a particular activity from the set of possible activities defined by its legal powers. These selected activities can be represented as goals too, because they also describe situations which the agent tries to achieve. We call these kinds of goals *objectives*. In this sense objectives are powers (in the legal sense) to perform activities, selected for execution.

Duties and objectives are two classes of goals, which, from the representational point of view, do not differ. Both represent situations the agent tries to achieve. Their meaning is different, one represents goals the agents is enforced to select by its obligations, the other represents goals the agent selects based on its own decision. For this reason in the model we will separate duties from objectives. We treat duties and objectives as different data structures.

For the model it is sufficient to work with predefined objectives to represent the social processes we are interested in. Consequently no complete goal generation mechanism is necessary. It is sufficient to generate only duties during the execution. Respectively, no explicit representation of powers (in the legal sense) is required, which would be necessary to generate objectives of the agents.

7.8 The specification

This section gives the specification of the agent-based model[1] directly applying the abstract architecture of the model (see chapter 5). We introduce the agent level of the specification (corresponding to the set of agent operations Op) and the world level of the specification (corresponding to the laws of the universe U). We combine both parts in the execution model of the system.

7.8.1 The specification of the agent level

The specification of the agent consists of the specification of the agent as a whole and of the specification of its internal structure. We assume a state-based model, i.e., the model changes through discrete states called world states (ws).

The abstract agent

The class *AbsAgent* (see code listing 7.1) defines the general properties of an agent. An agent a is determined by its unique identifier (its name) aid and by the current state of the world ws. The agent is characterized by two operations: the operations *newAgt* creates a new agent a with the identifier aid. The operation *doAgt* represents the activity cycle of one agent a in the world state ws.

Listing 7.1: Specification of an abstract agent

```
class AbsAgent a aid ws | a → ws, a → aid where
  newAgt :: aid → a
  doAgt :: ws → a → a
```

[1]In this chapter we describe the most important parts of the model. For the complete listing of the code see appendix A

The internal operations of the agent

The specification of the internal operations of the agent consists of a domain independent part, directly transforming the abstract agent architecture into a specification (operations from the class *AgentInternals*) , and of a domain dependent, model specific part (operations from the classes *MyAgentInternals*, *UpdStateInMsgs* and *UpdStateOutMsgs*).

The domain independent part of the specification. The domain independent internal operations of the agent are specified by the class *AgentInternals* (see code listing 7.2), which defines the general activity cycle of the agent.

Listing 7.2: The specification of the internal operations of the agent

```
class MyAgentInternals aid ws p i a ⇒ AgentInternals aid ws p i a where
  updStatePercepts :: aid → ws → [p] → i → i
  selActs :: aid → ws → i → [a]
  updStateActions :: ws → [a] → i → i
  decision :: aid → ws → [p] → i → ([a], i)

  decision aid ws p i =
     ( selActs aid ws ( updStatePercepts aid ws p i),
       updStateActions ws (selActs aid ws ( updStatePercepts aid ws p i))
         (updStatePercepts aid ws p i ) )
  updStatePercepts aid ws p = updStateInMessages aid ws p. updStatePhy aid ws p
  selActs aid ws i = (actDuties aid ws i) ++ (actObjectives aid ws i )
  updStateActions = updStateOutMessages
```

The activity cycle of the agent is determined by its decision making process, since we do not explicitly represent the perception and action process (see section 7.2). An agent is characterized by its percepts p, its internal state i and its actions a (see equation 5.1). The decision making process of the agent creates a set of actions and an updated version of the internal state based on the agent's percepts and its internal state (see equation 5.9). It is specified by the operation *decision*, which maps for an agent *aid* in world state *ws* a set of percepts p and the internal state i to a tuple $([a], i)$ consisting of the agent's actions and of the updated version of the internal state.

The decision making process first updates the internal state of the agent based on its percepts (see equation 5.8) represented by the operation *updStatePercepts*, second it generates the actions of the agent based on the updated internal state (see equation 5.7) represented by the operation *selActs* and third updates the internal state of the agent based on the agent's actions (see equation 5.8), which is represented by the operation *updStateActions*.

The domain dependent part of the specification. The axioms of the specification *AgentInternals* (see listing 7.2) define the operations of the abstract agent architecture in terms of model specific operations. The model specific properties, which are relevant here, are the action selection based on different types of goals (duties and objectives) and the update of the internal state based on physical and communication percepts and actions. The specification 7.3 defines the domain specific operations.

Listing 7.3: The domain dependent part of the specification

```
class ( UpdStateInMsgs aid ws p i, UpdStateOutMsgs ws a i) ⇒
```

```
        MyAgentInternals aid ws p i a | i→ aid, i→ws, i → p, i → a
where
  actDuties :: aid → ws → i → [a]
  actObjectives :: aid → ws → i → [a]
  updStatePhy :: aid → ws → [p] → i → i
  updStateInMessages :: aid → ws → [p] → i → i
  updStateOutMessages :: ws → [a] → i → i

  updStateInMessages aid ws p = updSubjObjInMsgs aid ws p. updEvtInMsgs aid ws p.
                                      updRegMsgs aid ws p
  updStateOutMessages ws a = updSubjObjOutMsgs ws a . updEvtOutMsgs ws a
```

The actualization operation of the internal state based on the percepts of the agent (operation *updStatePercepts*) updates the internal state based on the physical percepts (operation *updStatePhy*) and of the communication percepts (operation *updStateInMessages*) of the agent. The update of the state according to the actions (operation *updStateActions*) takes place in terms of the agent's selected communication actions (by the operation *updStateOutMessages*). The action selection operation (operation *selActs* selects activities based on the duties of the agent (operation *actDuties*) and based on the agent's objectives (operation *actObjectives*).

The update operations of the internal state represent the realization of constitutive rules in the model. The internal state of the agent *aid* is updated based on its communication percepts *p* by operations of the specification *UpdStateInMsgs* (see listing 7.4) and based on its communications activities *a* by the operations of class *UpdStateOutMsgs* (see listing 7.5)

Listing 7.4: Update of the internal states based on the agent's percepts

```
class UpdStateInMsgs aid ws p i where
  updRegMsgs :: aid → ws → [p] → i → i
  updEvtInMsgs :: aid → ws → [p] → i → i
  updSubjObjInMsgs :: aid → ws → [p] → i → i
```

Listing 7.5: Update of the internal states based on the agent's actions

```
class UpdStateOutMsgs ws a i where
  updEvtOutMsgs :: ws → [a] → i → i
  updSubjObjOutMsgs :: ws → [a] → i → i
```

The operation *updRegMsgs* updates the internal state of the agent by information from the registry agent, which has, by definition, complete and correct knowledge about the legal situation. The operations *updEvt∗* and *updSubjObj∗* represent the two levels of constitutive rules (see section 4.12), i.e., status assignment to events and to subjects and objects.

7.8.2 The specification of the world level

Following a state based approach for the execution model of the system, the world, i.e., the system, changes from one discrete state to the other. A state represents the world at one moment in time. There is one initial state, the state in the beginning where no change has occurred. In the model world states are represented by integer values starting with zero. A world state is characterized by one execution cycle of the model.

The class $Worlds$ specifies the operations on the world level (see listing 7.6). The operation $doWorld$ represents one simulation run and creates a world history. In each cycle of the simulation $doWorld$ calls the operation $runEnv$.

Listing 7.6: The specification of the world

```
class Worlds w where
  incWS :: w → w
  performActs :: w → w
  sendEnvReacts :: w → w
  sendMsgs :: w → w
  doAgts :: w → w
  runEnv :: w → w
  doWorld :: [w] → Int → [w]

  doWorld whist count = if (count > 0) then
     doWorld (whist++[w']) (count−1) else whist where
     w'= runEnv (last whist)
  runEnv = incWS. doAgts. performActs. sendEnvReacts. sendMsgs
```

An execution cycle on the world level of the model corresponds to the operation $runEnv$ (see also equation 5.11 of the abstract architecture), which we characterized as the reaction of the environment to the agents activities. The operation $runEnv$ maps the world w to a new state of the world. It first sends all messages from each agent to the receivers (operation $sendMsgs$), second, it sends the reactions of the environment if physical actions fail (operation $sendEnvReacts$), third, it performs the successful physical actions (operation $performActs$). Fourth, it calls the activity function of each agent (operation $doAgts$). Last it increments the world state (operation $incWS$).

7.8.3 The execution model

We distinguish the world level and the agent level of the execution model. The agent level consists of the execution cycles of each agent representing its decision making process (operations form Op) and the world level comprises the execution cycle representing the reaction of the environment (operations from U).

The action cycle of the agents consists of the following three steps corresponding to the decision making process of the agent:

1. update the beliefs and duties of the agent based on its percepts,
2. decision about communication and physical actions to be performed,
3. update of the beliefs of the agent based on the actions of the agent.

The execution cycle on the world level consists of three steps, the first two steps correspond to the reaction of the environment, the third step starts the activity cycle of each agent:

1. Send all messages from all agents from the previous world state to the intended addressees.
2. Perform all physical actions of all agents on the objects, i.e., on land in the model. Generate environment reactions, i.e., physical percepts for the agents if activities fail.

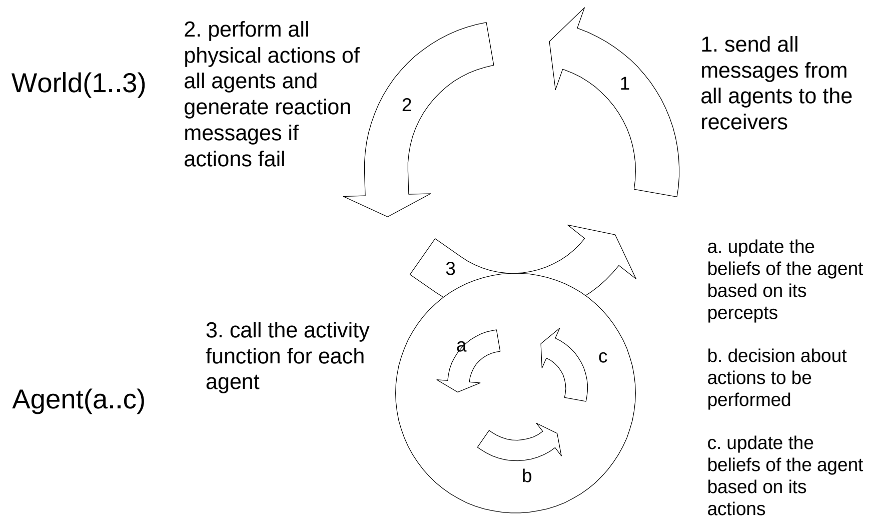
World(1..3)
2. perform all physical actions of all agents and generate reaction messages if actions fail
2
1
1. send all messages from all agents to the receivers
3
a. update the beliefs of the agent based on its percepts
3. call the activity function for each agent
a
c
b. decision about actions to be performed
Agent(a..c)
b
c. update the beliefs of the agent based on its actions

Figure 7.2: The execution model of the system

3. Call the activity function of each agent.

Figure 7.2 shows the execution model of the system.

7.8.4 The flow of operation in the specification

This subsection shows the flow of operation (see figure 7.3) according to the execution model discussed in subsection 7.8.3.

The execution starts with the empty world data structure. The functions $initWorldOT$ and $initWorldU$ (see chapter 8) initialize the world data according to the case studies, i.e., the intended simulation. Then the main execution function $doWorld$ is called and the simulation evolves according to the specification. The execution of the model ends with the operation $showWorld$ displaying the results of the simulation on the screen.

7.9 Representation

In this section we present the data structures for the computational model. We distinguish data structures for the message exchange, i.e., messages and data structures for the components of the environment E, comprising the agents and their internal state as well as pieces of land.

7.9.1 Identifiers and world states

It has to be possible to uniquely identify entities in the model. For this purpose the model implements identifiers. Data types are defined for identifiers as follows:

type *AgentID* = **String**
type *LandID* = **String**

These two kinds of identifiers represent unique names for Agents and pieces of land. Identifiers are implemented as strings. The concrete representation is not important, crucial is the uniqueness property.

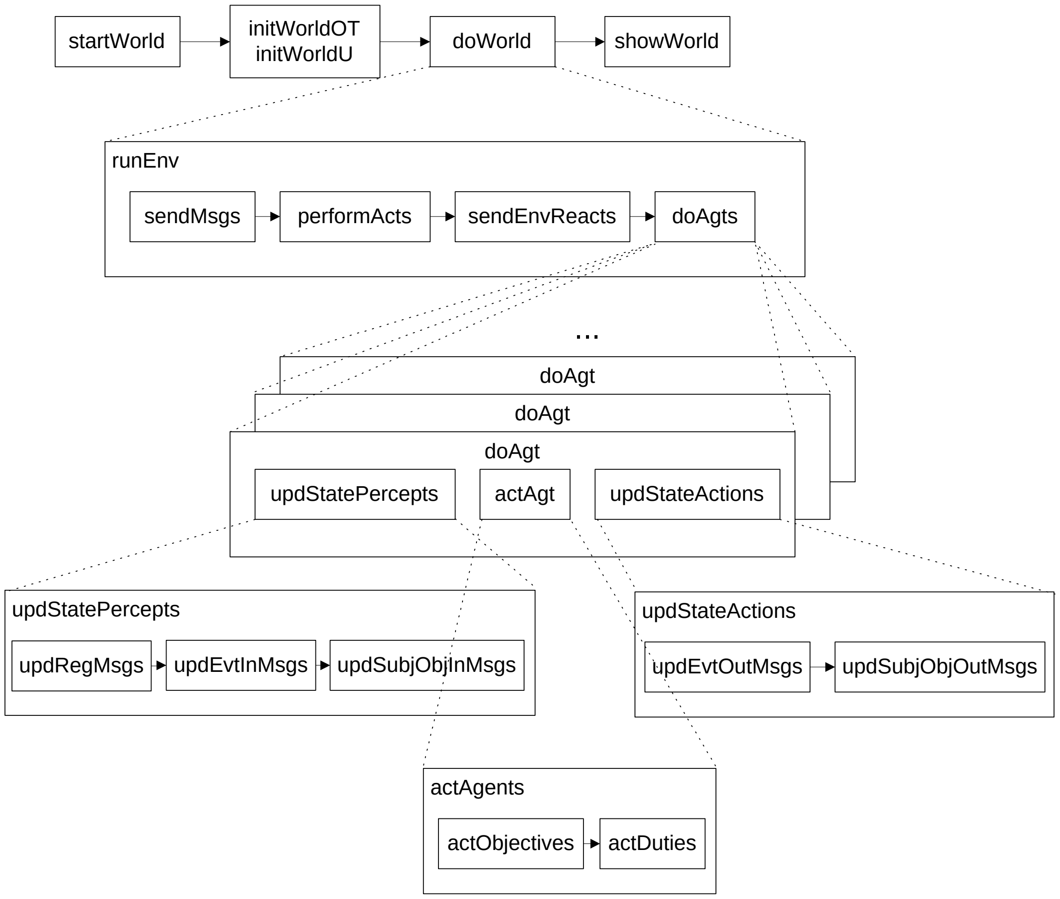

Figure 7.3: The flow of operation in the specification

The state of the world is represented by an integer value.

type *WorldState* = **Int**

7.9.2 Messages

The basic element in message exchange is the structure of the messages to be transferred. We distinguish messages that transport activities (*Act*) from messages that transport percepts (*Percept*). We further distinguish communication messages from physical messages. Messages are data structures of the following form:

data *Act* = *PAct Action* | *CAct Message*
data *Percept* = *PPercept PhyPercept* | *CPercept Message*

The data type *Act* describes actions as either physical actions oder communication action. The data type *Percept* describes percepts as either physical percepts or communication percepts.

Communication messages

A communication message can be the content of an agent's communication action or a communication percept. In the model is a data structure of the following form:

data *Message* = *Message AgentID AgentID Content MessageType Message*

The first parameter is an agent identifier identifying the sender of the message by its name or address. The second parameter determines the address of the receiver of the message. The message transports information encoded in the third argument, the message content. The content of a message can be the sale of a parcel, the query about the owner of a parcel or a complaint against another person:

```
data Content = Sell AgentID AgentID LandID
             | Query_Owner AgentID LandID
             | Complaint AgentID AgentID LandID
```

Messages are typed with the fourth argument. The type defines the purpose of the message. Valid message types are, for instance (not a complete list):

```
data MessageType = Offer | Query | Answer_Query |
                   Application | Judgement
```

A message can be an offer to sell a parcel. It can be a query regarding the ownership of a parcel or the answer to such a query. A message can be the application of ownership transfer or it can be a judgement deciding a complaint.

Communication actions are speech acts (see subsection 3.8.1). They have the purpose to achieve some effects on the addressee. They affect and change the beliefs of the addressee. In general communication actions can be defined as operations that change the internal state of the other agents. Messages can be categorized according to the speech act types given in subsection 3.8.1. An offer is a commissive speech act. The offeror promises to do something in the future (e.g., apply for ownership transfer) if the transferee accepts the offer. A query and an application are directive speech acts. An agent asks another agent to give some information or to act in a particular way with respect to the content of the application. The answer to a query is a representative speech act. An agent informs another agent about some state of affairs. The judgement is an example of a declarative speech act explicitly creating a new institutional fact.

The last parameter in the message data type is another message. It plays an important role since it realizes the proof of a status. For instance, an owner suing has to prove his ownership at the court. In his complaint messages he provides the proof of his ownership as a message. This message is the speech act that created his ownership right, i.e., the message he originally received by the registry agent, which caused the transfer of ownership to him.

Physical action messages

Physical action messages are conveyed by the agents to the environment, which reacts to these actions. Physical actions are represented by data structures of the following form:

```
data Action = ActUseLand LandID AgentID
            | ActAbandonLand LandID
            | ActEvictLand LandID
```

The data type represents three activities regarding land use. An agent can start to use a piece of land determined by a land identifier and the agent can abandon the land use. An agent can use physical power against another agent: An agent can evict land from another agent.

Physical activities of persons have effects on physical objects and subjects. We investigate physical actions concerning land use. Physical actions represented by land use have effects on one category of objects, on land pieces. Additionally they can affect subjects, i.e the user of a land by evicting the land.

Physical percept messages

Physical percept messages represent the case that actions of agents fail. In this case the environment does not realize the actions of the agents, it generates reaction messages (of type *EnvReaction*) which are conveyed to the acting agents. For instance, an agent tries to use a piece of land another agent already uses. The environment generates a physical percept message showing the agent that his action failed, i.e., environment reactions are synonymous with physical percepts. Physical percept messages are data structures of the following form:

data *PhyPercept* = *LandUse AgentID AgentID LandID*
type *EnvReaction* = *PhyPercept*

The physical percept $LandUse$ shows an agent that the land he tries to use, is already used by another agent. The reaction messages of the reactions of the environment are of type $PhyPercept$, i.e., the environment reacts by providing physical percepts to the agents.

7.9.3 The internal state of the agent

According to the components of the model analyzed in subsection 7.4, the internal state must provide representations of *status beliefs* and *rights*. Since we apply a goal-based agent (see subsection 5.6.3), the agent's internal state consists of knowledge about the current situation and former situations (B), which are the status beliefs, and of knowledge about situations the agent tries to achieve (G), which are the goals of the agent (see equation 5.2). The goals of the agent comprise duties and objectives (see section 7.7).
The internal state of the agent is represented by a data structure of the following form:

data *IntState* = *IntState* [*Status*] [*Duty*] [*Objective*]

Status beliefs

Status beliefs represent the agent's knowledge about the institutional status of physical phenomena, i.e., beliefs about assigned status functions. The model represents status beliefs as data structures of the following form (incomplete list):

```
data Status = Owner AgentID LandID Message WorldState
            | Legal_person AgentID WorldState
            | Parcel LandID WorldState
            | Buyer AgentID Message WorldState
            | Seller AgentID Message WorldState
            | Contracting Message WorldState
            | Transferring Message WorldState
```

An agent can have beliefs about who owns a piece of land. He can know legal persons. Agents can believe that a particular piece of land is a parcel. In the case that two persons enter into a sales contract the agent can believe that one of the persons is the seller and the other person is the buyer. An agent can have beliefs about the institutional status of events, e.g., that an event counts as contraction or as ownership transfer.

The beliefs of the agent change over time. They depend on the state of the world. Every status belief has a reference to the world state (parameter $WorldState$) in which it was created, i.e., to the moment in time in which the event occurred that constituted the institutional fact.

Events create status beliefs. Every dynamic status belief needs a reference to the event that created it. Respectively dynamic status beliefs contain the messages that caused these beliefs. This captures the fact that physical and institutional reality are connected. The message is the physical foundation for the status belief. The message has the property that it has a particular institutional meaning, i.e., a status function assigned.

Goals

Duties are data structures of the following form (incomplete list):

```
data Duty = AnswerApplication Message WorldState
          | DoApplication Message WorldState
```

In a specific world state an agent representing the cadastral registry can have the duty to answer an application to transfer ownership. An owner of a parcel, who sold his parcel, has the duty to apply for the ownership transfer. A duty refers to the event, i.e., to the message which created it. It is valid and enforces activities in the world state identified by $WorldState$.
Objectives are data structures of the following form (incomplete list):

```
data Objective = SellParcel LandID AgentID WorldState
               | UseLand LandID WorldState
```

An agent can have the objective to sell a parcel to another agent or an agent can have the objective to use a piece of land. Objectives refer to a world state, i.e., to the moment in time they are valid.

7.9.4 The agent

The state of an agent is characterized by its percepts, its actions and its internal state (see equation 5.1). Since in the message exchange architecture of this model percepts and actions are messages, the general structure of the agent can be described as consisting of an inbox, an internal state, and of an outbox.

$$Agent =< inbox >< internal\, state\, of\, the\, agent >< outbox >$$

This gives the following data structure for the agent:

```
data Agent = Agent AgentID [Percept] IntState [Act]
```

An agent has a unique identifier allowing its identification in the environment. The inbox of the agent consists of a list of percepts, the outbox consists of a list of activities. The inbox contains all messages sent to an agent in the current world state. The outbox contains the actions the agents selected for execution in the current world state.

7.9.5 Land

It is sufficient to represent land pieces by a unique *land identifier*. Further in order to capture physical activities in the case of land use, land has to include the reference to the current user of the land as an agent identifier (see section 7.9.1). A piece of land is a data structure that comprises the following elements:

data *Land* = *Land LandID AgentID*

The first parameter of type $LandID$ is the unique identifier of the land. The second parameter of type $AgentID$ represents the agent currently using the land.

7.9.6 The environment

It is now possible to specify the components, i.e., the data structures of the environment, that are capable of representing reality in a cadastre. The environment is represented by the following data structure $World$:

data *World* = *World* [*Agent*] [*Land*] *WorldState*

The world consists of agents and land pieces. The current state of the world is represented by the $WorldState$.

7.10 Implementation

The implementation combines the algebraic specifications with representations and realizes the functions on the representations. This section discusses implementation issues to the extent necessary to understand the model. We assume that for the understanding of the model it is sufficient to focus on the implementation of constitutive rules, which are a core part of the model, and on the implementation of the decision making process.

7.10.1 Constitutive rules

We describe the translation of constitutive rules into operations and show how to implement the operations.

Transforming constitutive rules into operations

According to Searle's theory the assignment of status functions to phenomena can be described by *constitutive rules* of the following form (see section 3.4.5):

X counts as Y in C.

The (perhaps) physical fact X counts at the institutional level as Y in a context C. These rules can be hierarchically nested. In the model we use operations to represent these rules:

$$r : X \times C \longrightarrow Y$$

These operations form a recursive structure. For instance, there are two rules (with empty context C_1 and C_2 for simplicity reasons):

A human being (X_1) counts as legal person (Y_1).

$$r_1(X_1, C_1) = Y_1$$

A legal person (Y_1) counts as owner (Y_2).

$$r_2(Y_1, C_2) = Y_2$$

The following non recursive structure replaces the second definition:

A human being (X_1) who is a legal person (C_2') counts as owner.

$$r(X_1, C_2') = Y_2 \, with \, C_2' = C_1 \cup Y_1$$

Operations representing constitutive rules have as parameter a physical fact (always an event represented by a message) and as second parameter the context that comprises other status functions that are assigned to the physical fact. *The non recursive structure of constitutive rules* can be introduced for all status functions assigned in the model of a cadastre with the goal to simplify the formalization.

The implementation of constitutive rules

According to the two levels of constitutive rules (see section 4.12) in the specification there are two operations for the update of the internal state based on the percepts (see listing 7.4) and of the actions (see listing 7.5) each. The operations *updEvtInMsgs* and *updEvtOutMsgs* create status beliefs for events and the operations *updSubjObjInMsgs* and *updSubjObjOutMsgs* create status beliefs for objects and subjects. The events that cause the creation of status beliefs are always communication actions, i.e., messages. These operations call for each message following operations, which are the realization of constitutive rules:

```
updStateEvtMsg :: ws → [s] → m → s
updSubjObjMsg :: ws → [s] → Bool → m → ([s],[d],[o])
```

The operation *updStateEvtMsg* takes the current world state *ws*, the set of current status beliefs of the agent [*s*], a message *m* and creates a new status belief *s*. The operation *updSubjObjMsg* takes the same input parameter and an additional boolean parameter, which determines whether the function is called with a message from the percepts or actions. The result of the operation is a tuple consisting of a set of status beliefs, a set of duties and a set of objectives. In the case that it is called with a message from the percepts the operation creates status beliefs, duties and objectives for the agent. In the case that the message is an action, the operation only creates the knowledge of the agent according to its own activities.

An example

The following example shows the construction of status beliefs in the case that a person applies for ownership transfer of a parcel. The physical phenomenon that causes the assignment of status is the message with the content saying "this is the application to transfer ownership of a parcel from the seller to the buyer". If the receiver of the message is the agent representing the cadastral registry and the sending person is a legal person than this message counts as application, otherwise no status is assigned.

```
updStateEvtMsg ws sl (Message s z (Sell  s  b  l) Application rmsg) =
   if ((isRegAgent sl z) && (isLegalPerson sl s)) then
         (Applying (Message s z (Sell  s  b  l) Application rmsg) ws)
      else NoStatus
```

If a message has the status assigned that it is an application of ownership transfer, then the sender of the message counts as 'applicant' and the receiver of the message, who is the registry agent, is the application receiver. To the status 'application receiver' the duty to decide the application is connected; it means the duty either to accept the application and transfer the ownership or to reject the application.

```
updSubjObjMsg ws sl inflag (Message s z (Sell s b l) Application rmsg) =
   if (isApplying sl (Message s z (Sell s b l) Application rmsg)) then sdlist
      else([NoStatus],[NoDuty], [NoObjective])
   where
      sdlist = ([(Applicant s (Message s z (Sell s b l) Accept_Offer rmsg) ws),
         (Application_receiver z (Message s z (Sell s b l) Accept_Offer rmsg) ws)],
         duty, [NoObjective])
      duty = if (inflag==True) then
         [AnswerApplication (Message s z (Sell s b l) Application rmsg) ws] else [NoDuty]
```

7.10.2 The implementation of the decision making process

We connect the specification of the internal operations of the agent (see code listings 7.2 and 7.3), with the representations by the definition of instances.

```
instance AgentInternals AgentID WorldState Percept IntState Act
instance MyAgentInternals AgentID WorldState Percept IntState Act where ...
```

We get the following functions that implement the operations from the specification:

```
updStatePercepts :: AgentID → WorldState → [Percept] → IntState → IntState
selActs :: AgentID → WorldState → IntState → [Act]
updStateActions :: WorldState → [Act] → IntState → IntState
decision :: AgentID → WorldState → [Percept] → IntState → ([Act], IntState)
```

The functions apply for an agent with the identifier $AgentID$ in some world state $WorldState$. The function *updStatePercepts* takes a set of communication and physical messages and the current state of the agent consisting of status belief, duties and objectives and creates new status beliefs, duties and objectives. The function *selActs* takes status beliefs, duties and objectives and creates physical and communication actions. The function *updStateActions* maps a set of action messages and updates the internal state. The *decision* function connects the three functions and maps a set of percept messages and the internal state to a new version of the internal state and a set of action messages.

7.11 Summary

In this chapter we developed the agent-based model of reality in a cadastre. We constructed a multi-agent system for the simulation of social processes in a cadastre. We introduced the specifications, the representations and some important aspects of the implementation. The communication mechanism used in the system is message exchange. We applied the abstract agent architecture developed in section 5.7 to the construction of the model.

According to the simulation approach (see figure 5.1) in this chapter we constructed the simulation model based on the theoretical basis developed in the previous chapters. The major result of this chapter is a computational model, which allows the execution of simulations of social processes in a cadastre. The model combines the results from the analysis of reality in a cadastre (chapter 4) with an agent-based framework (chapter 5) and an algebraic specification approach (chapter 6) for the model construction process. We have shown that it is possible to construct a computational agent-based model based on the ontological assumptions we made about reality in a cadastre. The remaining task for this thesis is the validation of the model with appropriate case studies.

This chapter concluded the second part of this thesis, the construction of the agent-based model. The next chapters discuss the third part, agent-based simulation, with the intention to prove the correctness of the model.

Chapter 8

Agent-based simulation of social processes in a cadastre

8.1 Introduction

In chapter 7 we developed the computational model of reality in a cadastre. We gave the specifications, introduced representations and discussed implementation aspects. This chapter tests the validity of the model by agent-based simulation. That means to let the program run with appropriate input data. After the simulation ends the output of the model can be compared with the situation in reality.

The chapter focuses on the two characteristic processes, which we introduced in section 4.7:

1. The transfer of ownership of a parcel between two persons.

2. The conflict between two persons regarding the use of a piece of land.

We assume that together these processes represent an essential part of reality in a cadastre as ownership and ownership transfer are the central elements influencing the legal status of parcels. The transfer of ownership represents the normal flow of operation where all people act according to the legal rules. We simulate conflicting cases where a person does not recognize the legal rules by the example of land use: The owner of a parcel sues a person that unauthorized uses his parcel to mobilize the power of the state (in the execution process) to enforce his rights.

We discuss social processes based on the Austrian legal system. On the level of the simulation it is necessary to focus on a specific legal system to achieve a realistic model. The results of the simulation have to correspond to reality regulated by the Austrian law. We have to check the validity of the model and its correctness with respect to the Austrian cadastral system we introduced in chapter 2.

The simulation has inputs and outputs. The input consists of the initial world state and the specification of the constitutive rules (see subsection 7.10.1). The initial world state comprises the representation of the subjects and objects existing in the model. Subjects and objects in the model are land pieces and agents. Main important element in the initial world state is the specification of the beliefs of the agents representing their objectives and the initially existing institutional facts.

The output of the simulation is the set of all world states executed during the simulation. This chapter has for each case study of the following structure:

1. Definition of the input data
2. Discussion of the output

The chapter concludes with the assessment of the results of the simulation, i.e., with the discussion to which extent the model correctly represents reality in a cadastre.

8.2 Case study 1: The transfer of ownership of a parcel

This section discusses the input and output of the ownership transfer simulation. We present only parts of the output of the simulation here. For the complete output see appendix B. For the presentation of the initial world state we use the output of the simulation with 0 cycles, i.e., where no action occurred yet. All elements of the output correspond to data structures defined in the model.

8.2.1 The input of the simulation

Objects and subjects: land pieces and agents

Within the category of subjects there are three agents, the registry agent ”RegAg” representing the cadastral registry, the agent ”A” who wants (has the objective) to sell his parcel and the agent ”B” who wants to buy the parcel. The agents know each other and the parcel ”P”.

```
 Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0)
 (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Legal_person "B" 0) (Legal_person "A" 0) (Owner "A" "P" <m> 0)
 (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[SellParcel "P" "B" 0]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )
```

The registry agent "RegAg" and the owner of the parcel "A" both have the belief `(Owner "A" "P" <m> 0)` meaning that both believe that "A" is the owner of parcel "P". The belief was created in world state 0 and `<m>` is the message that caused the construction of this belief. We do not include the message `<m>` into the output of the program for clearness reasons. Agent "A" has the objective `SellParcel "P" "B" 0` meaning that "A" wants to sell his parcel "P" in the first world state 0 to "B".

For the simulation of ownership transfer is sufficient to have one piece of land with the name "P" and currently no land user.

```
Land "P" ""
```

The constitutive rules

Within constitutive rules we distinguish event rules (rules directly assigned to events) and subject/object rules (rules assigned to subjects and objects participating in an event). See subsection 4.12 for a discussion. Figure 8.1 shows the hierarchy of constitutive rules necessary to represent this case study.

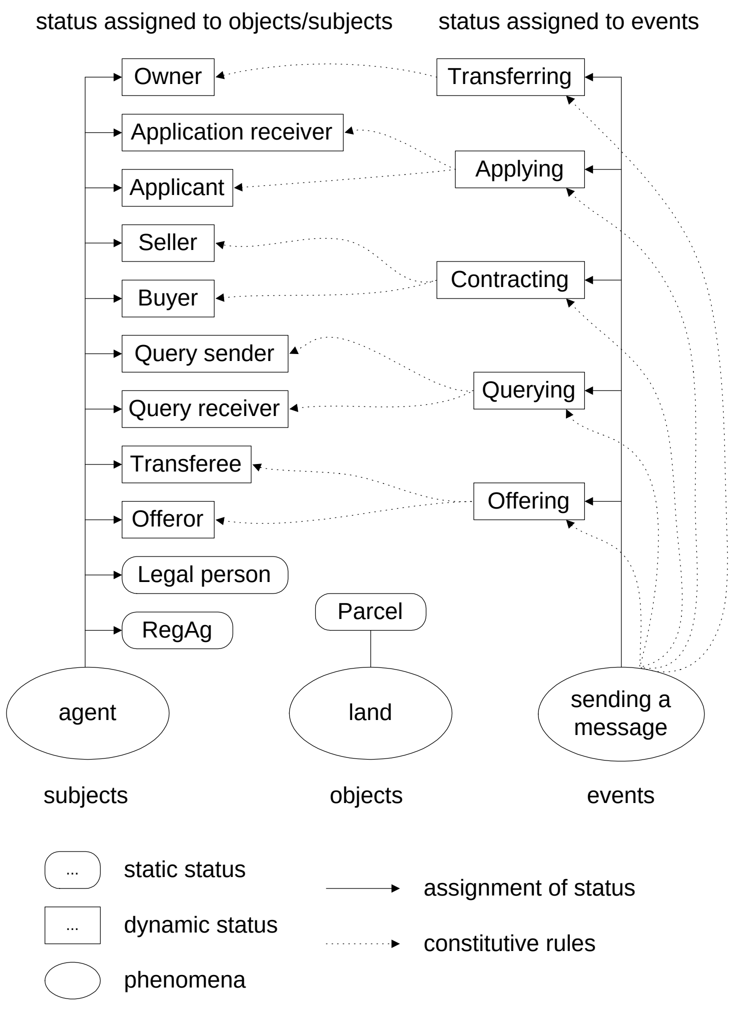

Figure 8.1: Constitutive rules and status assigned for ownership transfer

Event rules. Event rules necessary for the representation of an ownership transfer are the following:

```
updStateEvtMsg ws sl (Message s z (Sell s b l) Offer rmsg) =
    if ((isLegalPerson sl s) && (isLegalPerson sl b) && (isParcel sl l))
       then (Offering (Message s z (Sell s b l) Offer rmsg) ws) else NoStatus
```

A message counts as offering, if its content is an offer, if it is sent from one legal person to another legal person and if the concerning piece of land is a parcel.

```
updStateEvtMsg ws sl (Message s z (Query_Owner aid l) Query rmsg) =
    if ((isRegAgent sl z) && (isLegalPerson sl s)) then
         (Querying (Message s z (Query_Owner aid l) Query rmsg) ws) else NoStatus
```

A message counts as querying, if its content is a query, if the receiver of the message is the registry agent and the sender is a legal person.

```
updStateEvtMsg ws sl (Message s z (Sell s b l) Accept_Offer rmsg) =
    if ((isOfferor sl z) && (isTransferee sl s)) then
        (Contracting (Message s z (Sell s b l) Accept_Offer rmsg) ws)
       else NoStatus
```

A message counts as contracting, if the receiver is an offeror and the sender is a transferee and the content of the message is the acceptance of an offer.

```
updStateEvtMsg ws sl (Message s z (Sell s b l) Application rmsg) =
    if ((isRegAgent sl z) && (isLegalPerson sl s)) then
         (Applying (Message s z (Sell s b l) Application rmsg) ws)
       else NoStatus
```

A message counts as applying (of ownership transfer), if its content is an application, if the receiver is the registry agent and the sender is a legal person.

```
updStateEvtMsg ws sl
    (Message s z (Sell s b l) Accept_Application rmsg) =
       if ((isRegAgent sl s) && (isLegalPerson sl z)) then
          (Transferring (Message s z (Sell s b l) Accept_Application rmsg) ws)
          else NoStatus
```

A message counts as transferring (of ownership of a parcel) if the content of the message is the acceptance of an application, if the sender is the registry agent and the receiver is a legal person.

Subject/object rules. Subject/object rules necessary for the representation of an ownership transfer are the following:

```
updSubjObjMsg ws sl inflag (Message s z (Sell s b l) Offer rmsg) =
    if (isOffering sl (Message s z (Sell s b l) Offer rmsg)) then sdlist
       else ([NoStatus],[NoDuty],[NoObjective])
    where
       sdlist = ([(Offeror s (Message s z (Sell s b l) Offer rmsg) ws)] ++
          [(Transferee b (Message s z (Sell s b l) Offer rmsg) ws)], duty, objective)
      duty = if (inflag==True) then
                 [(AskRegAgent (Message s z (Sell s b l) Offer rmsg) ws)] else [NoDuty]
       objective = if (inflag==True) then
                    [(AnswerOffer (Message s z (Sell s b l) Offer rmsg) (ws+2))] else
                       [NoObjective]
```

A message makes the sender of a message to an offeror and the receiver of the message to a transferee if the message counts as offering. It creates for the receiver in the current world state the duty to ask the registry agent whether the sender of the message is the owner of the parcel. In the second next world state the receiver has the objective to answer to the offer.

```
updSubjObjMsg ws sl inflag (Message s z (Query_Owner aid l) Query rmsg) =
    if (isQuerying sl (Message s z (Query_Owner aid l) Query rmsg)) then sdlist
       else ([NoStatus],[NoDuty],[NoObjective])
    where
        sdlist = ( [(Query_sender s (Message s z (Query_Owner aid l) Query rmsg) ws)] ++
                   [(Query_receiver z (Message s z (Query_Owner aid l) Query rmsg) ws)],
                     duty, [NoObjective])
        duty = if (inflag==True)
           then [AnswerQuery (Message s z (Query_Owner aid l) Query rmsg) ws] else [NoDuty]
```

A message assigns the status query sender to the sender of the message and the status query receiver to the receiver of the message if the message counts as querying. It creates for the receiver the duty to answer the query.

```
updSubjObjMsg ws sl inflag (Message s z (Sell s b l) Accept_Offer rmsg) =
    if (isContracting sl (Message s z (Sell s b l) Accept_Offer rmsg)) then sdlist
       else ([NoStatus],[NoDuty],[NoObjective])
    where
        sdlist = ([(Seller s (Message s z (Sell s b l) Accept_Offer rmsg) ws),
           (Buyer b (Message s z (Sell s b l) Accept_Offer rmsg) ws)], duty, [NoObjective])
        duty = if (inflag==True) then
           [DoApplication (Message s z (Sell s b l) Accept_Offer rmsg) ws] else [NoDuty]
```

A message assigns the status seller to the agent which is mentioned in the message content as seller and assigns the status buyer to the agent mentioned in the message as buyer if the message counts as contracting. It creates for the seller the duty to apply for ownership transfer of the concerning parcel.

```
updSubjObjMsg ws sl inflag (Message s z (Sell s b l) Application rmsg) =
    if (isApplying sl (Message s z (Sell s b l) Application rmsg)) then sdlist
       else([NoStatus],[NoDuty], [NoObjective])
    where
        sdlist = ([(Applicant s (Message s z (Sell s b l) Accept_Offer rmsg) ws),
           (Application_receiver z (Message s z (Sell s b l) Accept_Offer rmsg) ws)],
           duty, [NoObjective])
        duty = if (inflag==True) then
           [AnswerApplication (Message s z (Sell s b l) Application rmsg) ws] else [NoDuty]
```

A message that counts as applying assigns the status applicant (of ownership transfer) to the sender and the status application_receiver to the receiver. It creates for the receiver the duty to answer to the application, i.e., to grant the application or reject the application.

```
updSubjObjMsg ws sl inflag (Message s z (Sell s b l) Accept_Application rmsg) =
    if (isTransferring sl (Message s z (Sell s b l) Accept_Application rmsg)) then sdlist
       else ([NoStatus],[NoDuty], [NoObjective])
    where
        sdlist = ([(Owner b l (Message s z (Sell s b l) Accept_Application rmsg) ws)],
           [NoDuty], [NoObjective])
```

If a message counts as transferring (ownership of a parcel), then it creates the status owner of the parcel for the agent mentioned as buyer in the message.

8.2.2 The output of the simulation

The output of the simulation is the list of all phenomena existing and occurring during the simulation in the model, i.e., the land piece, the registry agent, the agent "A" and the agent ”B”. In the following we present only the parts of the output which change, i.e., the state of agents that construct new beliefs, generate new goals or act in some way.

World state 0

```
Agent( AID=A
INBOX=[]
PHYPERCEPS=[]
STATUS=(Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0)
(Legal_person "B" 0) (Legal_person "A" 0) (Owner "A" "P" <m> 0)
(RegAg "RegAg" 0) (Parcel "P" 0)
DUTIES=[]
OBJECTIVES=[SellParcel "P" "B" 0]
ACTIONS=[]
OUTBOX=[Message "A" "B" (Sell "A" "B" "P") Offer NoMessage] )
```

In world state 0 the agent ”A” has the objective to sell the parcel. He decides to send a message containing an offer to the agent ”B”. He updates his internal state, which creates his knowledge about this activity and the assigned status.

World state 1

```
Agent( AID=B
INBOX=[Message "A" "B" (Sell "A" "B" "P") Offer NoMessage]
PHYPERCEPS=[]
STATUS=(Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1) (Querying <m> 1)
(Offeror "A" <m> 1) (Transferee "B" <m> 1) (Offering <m> 1) (Legal_person "B" 0)
(Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
DUTIES=[AskRegAgent (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 1]
OBJECTIVES=[AnswerOffer (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 3]
ACTIONS=[]
OUTBOX=[Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage] )
```

Agent ”B” receives the offer in world state 1 and updates his internal state with knowledge about the offer. If he wants to accept the offer, ”B” generates the objective to answer to the offer. ”B” needs to know whether the offeror is the legal owner of the parcel, and generates the duty to ask the registry agent. According to his duty ”B” sends a message with a query to the registry agent. He updates his internal state with the knowledge that he is asking (querying) the registry agent.

World state 2

```
Agent( AID=RegAg
INBOX=[Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage]
PHYPERCEPS=[]
STATUS=(Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2)
(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0)
(Legal_person "A" 0)
DUTIES=[AnswerQuery (Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage) 2]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[Message "RegAg" "B" (Query_Owner "A" "P") Answer_Query NoMessage] )
```

In world state 2 the registry agent receives the query and updates his internal state based on the message. He knows the owner of the concerning parcel and sends this information to the asking agent ”B”.

World state 3

```
Agent( AID=B
INBOX=[Message "RegAg" "B" (Query_Owner "A" "P") Answer_Query NoMessage]
PHYPERCEPS=[]
STATUS=(Seller "A" <m> 3) (Buyer "B" <m> 3) (Contracting <m> 3)
(Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1) (Querying <m> 1)
(Offeror "A" <m> 1) (Transferee "B" <m> 1)  (Offering <m> 1) (Legal_person "B" 0)
(Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
DUTIES=[]
OBJECTIVES=[AnswerOffer (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 3]
ACTIONS=[]
OUTBOX=[Message "B" "A" (Sell "A" "B" "P") Accept_Offer NoMessage] )
```

Agent ”B” receives this message in world state 3. He updates his internal state and knows now that the offering agent is the legal owner of the parcel. According to his objective he decides to accept the offer and sends a message with the acceptance of the offer to the agent ”A”. ”B” updates his internal state and knows now that he concluded a contract with the seller.

World state 4

```
Agent( AID=A
INBOX=[Message "B" "A" (Sell "A" "B" "P") Accept_Offer NoMessage]
PHYPERCEPS=[]
STATUS=(Applicant "A" <m> 4) (Application_receiver "RegAg" <m> 4) (Applying <m> 4)
(Seller "A" <m> 4) (Buyer "B" <m> 4) (Contracting <m> 4) (Offeror "A" <m> 0)
(Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A"  0)
(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
DUTIES=[DoApplication (Message "B" "A" (Sell "A" "B" "P")Accept_Offer NoMessage)4]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[Message "A" "RegAg" (Sell "A" "B" "P") Application NoMessage] )
```

Agent ”A” gets the acceptance of the offer in world state 4. He updates his internal state and generates the duty to apply for ownership transfer. According to his duty arising from the contract he sends a message to the registry agent containing an application of ownership transfer. He updates his internal state now knowing that he is the applicant (of the ownership transfer).

World state 5

```
 Agent( AID=RegAg
 INBOX=[Message "A" "RegAg" (Sell "A" "B" "P") Application NoMessage]
 PHYPERCEPS=[]
 STATUS=(Owner "B" "P" <m> 5) (Transferring <m> 5) (Transferring <m> 5)
 (Applicant "A" <m> 5) (Application_receiver "RegAg" <m> 5) (Applying <m> 5)
(Query_sender "B" <m> 2)
 (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m> 0)
 (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[AnswerApplication (Message "A" "RegAg" (Sell "A" "B" "P")
 Application NoMessage) 5]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "RegAg" "A" (Sell "A" "B" "P") Accept_Application NoMessage,
 Message "RegAg" "B" (Sell "A" "B" "P") Accept_Application NoMessage] )
```

The registry agent receives the application message in world state 5. He generates the duty to answer the application. He knows that the sender of the message is the owner of the parcel and decides to accept the application. He sends messages with this information to "A" and "B". He updates his internal state now knowing that "B" is the new owner of the parcel.

World state 6

```
Agent( AID=A
INBOX=[Message "RegAg" "A" (Sell "A" "B" "P") Accept_Application NoMessage]
PHYPERCEPS=[]
STATUS=(Owner "B" "P" <m> 6) (Transferring <m> 6) (Applicant "A" <m> 4)
(Application_receiver "RegAg" <m> 4) (Applying <m> 4) (Seller "A" <m> 4)
(Buyer "B" <m> 4) (Contracting <m> 4) (Offeror "A" <m> 0)
(Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A" 0)
(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
DUTIES=[]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[] )

Agent( AID=B
INBOX=[Message "RegAg" "B" (Sell "A" "B" "P") Accept_Application NoMessage]
PHYPERCEPS=[]
STATUS=(Owner "B" "P" <m> 6) (Transferring <m> 6) (Seller "A" <m> 3)
(Buyer "B" <m> 3) (Contracting <m> 3) (Query_sender "B" <m> 1)
(Query_receiver "RegAg" <m> 1) (Querying <m> 1) (Offeror "A" <m> 1)
(Transferee "B" <m> 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0)
(RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
DUTIES=[]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[] )
```

"A" and "B" receive the acceptance of the ownership transfer in world state 6. They update their internal state with the knowledge that the buyer is now the new owner. The simulation ends.

8.3 Case study 2: Conflicts regarding land use

This section discusses the input and output of the conflict resolution simulation.

8.3.1 The input of the simulation

Subjects and objects: land pieces and agents

Within the category of subjects there are four agents, the court agent representing the court responsible for the complaint, the sheriff responsible for the judgement execution, Agent "A" who wants (has the objective) to use his parcel and agent "B" who unauthorized uses the parcel.

```
Agent( AID=CourtAg
INBOX=[]
PHYPERCEPS=[]
```

```
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0)
 (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0)
 (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0)
 (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6,UseLand "P" 0]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0)
 (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0)
 (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage] )
```

All agents have knowledge about each other and about the parcel "P". In the initial world state the registry agent sends a message to "A" creating "A"'s ownership of parcel "P". "B" has in world state 0 the objective to use the parcel and "A" has in world state 1 the same objective. This constellation causes the complaint of the authorized user. In world state 6 and 7 there is the analogous situation. Again "B" avoids land use by the owner "A" . The difference to world state 1 is that the owner in world state 7 is holder of an execution title (after the judgement), which allows the start of the execution process and thus the second sub-process of the simulation starts.

As in the first case study, for the simulation it is sufficient to have one piece of land with no land user in the beginning.

```
Land "P" ""
```

The constitutive rules

Again we distinguish between event and subject/object rules. Figure 8.2 shows the hierarchy of constitutive rules necessary to represent this case study.

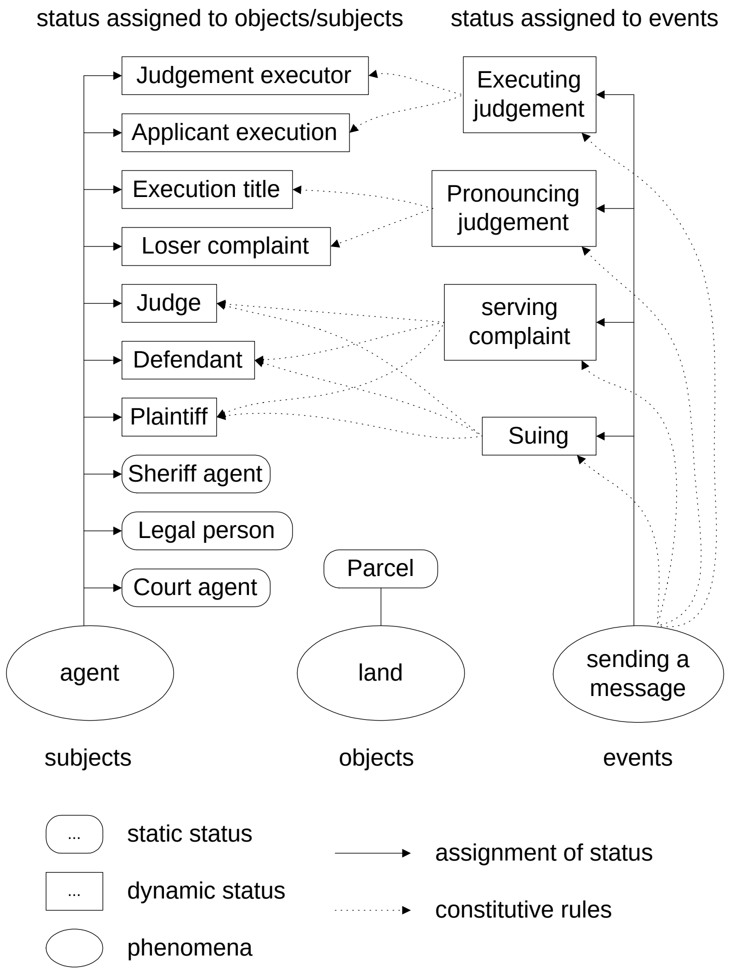

Figure 8.2: Constitutive rules and status assigned during the conflict resolution

Event rules. Event rules necessary for the representation of conflicts regarding land use are the following:

```
updStateEvtMsg ws sl (Message s z (Complaint p d l) Legal_Action proofmsg) =
   if ((isCourtAgent sl z) && (isLegalPerson sl s) && (s==plaintiff)) then
      (Suing (Message s z (Complaint p d l) Legal_Action proofmsg) ws)
      else NoStatus
```

A message counts as suing if its content is a complaint, if the sender is a legal person and the receiver is the court agent. The sender of the message must be mentioned as plaintiff in its content.

```
updStateEvtMsg ws sl (Message s z (Complaint p d l) Serve_Complaint proofmsg) =
   if ((isCourtAgent sl s) && (isLegalPerson sl z) && (z==d)) then
      (ServingComplaint (Message s z (Complaint p d l) Serve_Complaint proofmsg) ws)
      else NoStatus
```

A message counts as serving a complaint if its content is a complaint, if the sender is the court agent and if the receiver is a legal person. The receiver must be mentioned as defendant in the content of the message.

```
updStateEvtMsg ws sl (Message s z (Complaint p d l) Judgement proofmsg) =
    if ((isCourtAgent sl s) && (isLegalPerson sl z) &&
       (( isPlaintiff  sl  z ) || ( isDefendant sl  z))) then
          (PronouncingJudgement (Message s z (Complaint p d l) Judgement proofmsg) ws)
       else NoStatus
```

A message counts as pronouncing a judgement if the sender is the court agent and the receiver is a legal person and if the receiver of the message is either the plaintiff or the defendant.

```
updStateEvtMsg ws sl (Message s z (Complaint p d l) Apply_Execution proofmsg) =
    if (( isSheriffAgent  sl  z) && (isLegalPerson sl s) && (s==p)) then
       (ExecutingJudgement (Message s z (Complaint p d l) Apply_Execution proofmsg) ws)
       else NoStatus
```

A message counts as starting of a judgement execution if the receiver of the message is the sheriff agent, the sender is a legal person, which is mentioned in the message as plaintiff.

Subject/object rules necessary for the representation of conflicts regarding land use are the following:

```
updSubjObjMsg ws sl inflag (Message s z (Complaint p d l) Legal_Action proofmsg) =
    if (isSuing  sl  (Message s z (Complaint p d l) Legal_Action proofmsg)) then sdlist
       else ([NoStatus],[NoDuty], [NoObjective])
    where
       sdlist  = ([(Defendant d (Message s z (Complaint p d l) Legal_Action proofmsg) ws),
          ( Plaintiff  p (Message s z (Complaint p d l) Legal_Action proofmsg) ws),
          (Judge z (Message s z (Complaint p d l) Legal_Action proofmsg) ws)],  duty,
          [NoObjective])
       duty = if ( inflag==True) then
          [(DoServeComplaint (Message s z (Complaint p d l) Legal_Action proofmsg) ws),
           (DoJudgement (Message s z (Complaint p d l) Legal_Action proofmsg) (ws+2))] else
           [NoDuty]
```

If a message counts as suing, it assigns the status defendant to the person mentioned as defendant in its content and the status plaintiff to the person mentioned as plaintiff in its content. Additionally it assigns the status judge (in this legal action) to the receiver of the message. For the receiver it creates the duty to serve the complaint to the defendant in this world state and to pronounce a judgement in the second next world state.

```
updSubjObjMsg ws sl inflag (Message s z (Complaint p d l) Serve_Complaint proofmsg) =
    if (isServingComplaint sl (Message s z (Complaint p d l) Serve_Complaint proofmsg)) then
       sdlist  else ([NoStatus],[NoDuty],[NoObjective])
    where
       sdlist  = if ( inflag==True) then
          ([( Defendant d (Message s z (Complaint p d l) Serve_Complaint proofmsg) ws),
          ( Plaintiff  p (Message s z (Complaint p d l) Serve_Complaint proofmsg) ws),
          (Judge s (Message s z (Complaint p d l) Serve_Complaint proofmsg) ws)],
          [(DoAnswerComplaint (Message s z (Complaint p d l) Serve_Complaint proofmsg) ws)],
          [NoObjective])
          else
             ([NoStatus],[NoDuty],[NoObjective])
```

A message that counts as serving the complaint creates the beliefs that the person mentioned as plaintiff in its content, is the plaintiff and that the person mentioned as defendant in its content, is the defendant. The sender of the message counts as the judge. For the receiver of the message it creates the duty to answer the complaint.

```
updSubjObjMsg ws sl inflag (Message s z (Complaint p d l) Judgement proofmsg) =
   if (isPronouncingJudgement sl (Message s z (Complaint p d l) Judgement proofmsg)) then
      sdlist else ([NoStatus],[NoDuty],[NoObjective])
   where
      sdlist = ([(Loser_Complaint d (Message s z (Complaint p d l) Judgement proofmsg) ws),
         (Execution_Title p d land (Message s z (Complaint p d l) Judgement proofmsg) ws)],
         duty,[NoObjective])
      duty = if ((isDefendant sl z) && (inflag==True)) then
         [(DoAbandonLand (Message s z (Complaint p d l) Judgement proofmsg) ws)] else
         [NoDuty]
```

A message counting as pronouncing a judgement, creates the belief that the person mentioned in the messages as defendant, is the loser of the legal action. The person mentioned as plaintiff in the message is the winner of the legal action, which creates the execution title against the loser. If the receiver of the message is the defendant, it creates the duty to abandon the illegal land use.

```
updSubjObjMsg ws sl inflag (Message s z (Complaint p d l) Apply_Execution proofmsg) =
   if (isExecutingJudgement sl (Message s z (Complaint p d l) Apply_Execution proofmsg)) then
      sdlist else ([NoStatus],[NoDuty],[NoObjective])
   where
      sdlist = ([(Applicant_Execution s (Message s z (Complaint p d l)
         Apply_Execution proofmsg) ws), (JudgementExecutor z (Message s z (Complaint p d l)
         Apply_Execution proofmsg) ws)], duty,[NoObjective])
      duty = if (inflag==True) then [DoExecution (Message s z (Complaint p d l)
         Apply_Execution proofmsg) ws] else [NoDuty]
```

If a message counts as starting of a judgement execution, then it assigns the status applicant of the execution to the sender of the message and the status judgement executor (in this case) to the receiver of the message. For the receiver it creates the duty to perform the judgement execution.

8.3.2 The output of the simulation

The output of the simulation is the list of all phenomena existing and occurring during the simulation in the model, i.e., the land piece, the registry agent, the agent "A" and the agent "B", the court agent and the sheriff agent.

Again we present only the parts of the output which change, i.e., the state of agents that construct new beliefs, generate new goals or act in some way.

World state 0

```
Agent( AID=A
INBOX=[Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage]
PHYPERCEPS=[]
STATUS=(Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0)
(RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
(SheriffAg "SheriffAg" 0) (Parcel "P" 0)
DUTIES=[]
OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
ACTIONS=[]
```

```
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0)
 (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6,UseLand "P" 0]
 ACTIONS=[ActUseLand "P" "B"]
 OUTBOX=[] )
```

In world state 0 ”A” receives the message from the registry agent that creates his ownership right of parcel ”P”. Agent ”B” has the objective to use the parcel and generates a physical action starting the land use

World state 1

```
 Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0)
 (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
 ACTIONS=[ActUseLand "P" "A"]
 OUTBOX=[] )

Land "P" "B"
```

In world state 1 ”A” has the objective to use the piece of land ”P” and generates the action to start the land use. The land piece ”P” is already used by ”B”.

World state 2

```
 Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[LandUse "A" "B" "P"]
 STATUS=(Defendant "B" <m> 2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2)
 (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0)
 (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,DoComplaint "P" "B" 2]
 ACTIONS=[]
 OUTBOX=[Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action
 (Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage)] )
```

In world state 2 agent ”A” receives the physical percept that ”P” is already used by ”P”. He generates the objective to sue ”B” and generates the complaint message to the court. With the complaint he conveys the proof of his ownership right as second message, which originally created his right.

World state 3

```
Agent( AID=CourtAg
INBOX=[Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action
(Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage)]
PHYPERCEPS=[]
STATUS=(ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3)
(Judge "CourtAg" <m> 3) (Suing <m> 3) (CourtAg "CourtAg" 0)
(SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0)  (Parcel "P" 0) (Legal_person "A" 0)
(Legal_person "B" 0)
DUTIES=[DoServeComplaint <m> 3,DoJudgement <m> 5]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Serve_Complaint NoMessage])
```

In world state 3 the court agent receives the complaint and he updates his internal state accordingly. He generates the duty to serve the complaint to the plaintiff and to pronounce a judgement in the second next world state. He generates the message serving the complaint to ”B”.

World state 4

```
Agent( AID=B
INBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Serve_Complaint NoMessage]
PHYPERCEPS=[]
STATUS=(Defendant "B" <m> 4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4)
(ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0)
(Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
DUTIES=[DoAnswerComplaint (Message "CourtAg" "B" (Complaint "A" "B" "P")
Serve_Complaint NoMessage) 4]
OBJECTIVES=[UseLand "P" 6]
ACTIONS=[]
OUTBOX=[Message "B" "CourtAg" (Complaint "A" "B" "P") Answer_Complaint NoMessage])
```

In world state 4 the agent ”B” receives the message informing about the complaint. It generates the duty to answer to the complaint. According to this duty ”B” generates a message to the court agent with his answer to the complaint.

World state 5

```
Agent( AID=CourtAg
INBOX=[Message "B" "CourtAg" (Complaint "A" "B" "P") Answer_Complaint NoMessage]
PHYPERCEPS=[]
STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5)
(PronouncingJudgement <m> 5) (PronouncingJudgement <m> 5) (ServingComplaint <m> 3)
(Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3)
(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0)
(Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
DUTIES=[DoJudgement <m> 5]
OBJECTIVES=[]
ACTIONS=[]
OUTBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Judgement NoMessage,
Message "CourtAg" "A" (Complaint "A" "B" "P") Judgement NoMessage] )
```

In world state 5 the court agent receives the answer to the complaint from ”B”. Since the court agent has the duty to perform a judgement and ”B” cannot justify his behaviour, he accepts the complaint and generates two messages pronouncing the judgement to ”A” and ”B”.

World state 6

```
 Agent( AID=A
 INBOX=[Message "CourtAg" "A" (Complaint "A" "B" "P") Judgement NoMessage]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6)
 (PronouncingJudgement <m> 6) (Defendant "B" <m> 2) (Plaintiff "A" <m> 2)
 (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0)
 (CourtAg "Cour tAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Judgement NoMessage]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6)
 (PronouncingJudgement <m> 6) (Defendant "B" <m> 4) (Plaintiff "A" <m> 4)
 (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0)
 (RegAg "RegAg" 0) (Legal _person "A" 0) (Legal_person "B" 0)
 (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[DoAbandonLand (Message "CourtAg" "B" (Complaint "A" "B" "P")
 Judgement NoMessage) 6]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[ActAbandonLand "P",ActUseLand "P" "B"]
 OUTBOX=[] )
```

In world state 6 "A" and "B" receive the judgement message. "A" knows now that he has now an execution title against "B". For the loser of the complaint the message creates the duty to abandon the land. "B" physically acts and abandons the land "P".

To simulate the execution process agent "B" gets in world state 6 again the objective to use the parcel. "B" acts according to his objective and performs the action starting again the land use.

World state 7

```
 Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6)
 (PronouncingJudgement <m> 6) (Defendant "B" <m> 2) (Plaintiff "A" <m> 2)
 (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0)
 (CourtAg "CourtAg" 0)(RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[ActUseLand "P" "A"]
 OUTBOX=[] )
```

In world state 7 the "A" again has the objective to use his parcel "P" and generates a physical action to use the land.

World state 8

```
 Agent( AID=A
INBOX=[]
PHYPERCEPS=[LandUse "A" "B" "P"]
STATUS=(Applicant_Execution "A" <m> 8) (JudgementExecutor "SheriffAg" <m> 8)
(ExecutingJudgement <m> 8) (Loser_Complaint "B" <m> 6)
(Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6)
(Defendant "B" <m> 2) (Plaintiff "A" <m> 2 ) (Judge "CourtAg" <m> 2)
(Suing <m> 2) (Owner "A" "P" <m> 0)  (Transferring <m> 0)
(CourtAg "CourtAg" 0) (RegAg "RegAg" 0)(Legal_person "A" 0)
(Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
DUTIES=[]
OBJECTIVES=[ApplyExecution "P" "B" 8]
ACTIONS=[]
OUTBOX=[Message "A" "SheriffAg" (Complaint "A" "B" "P") Apply_Execution
(Message "CourtAg" "A" (Complaint "A" "B" "P") Judgement NoMessage)] )
```

In world state 8 ”A” preceives that ”B” uses parcel ”P”. Since ”A” has the execution title against ”B” he decides to send a message to the sheriff agent containing the application of the judgement execution. The message contains the message that created his execution title as proof of his right to apply for judgement execution. ”A” updates his internal state with the knowledge about the application.

World state 9

```
Agent( AID=SheriffAg
INBOX=[Message "A" "SheriffAg" (Complaint "A" "B" "P") Apply_Execution
(Message "CourtAg" "A" (Complaint "A" "B" "P") Judgement NoMessage)]
PHYPERCEPS=[]
STATUS=(Applicant_Execution "A" <m> 9) (JudgementExecutor "SheriffAg" <m> 9)
(ExecutingJudgement <m> 9) (CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0)
(RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
DUTIES=[DoExecution <m> 9]
OBJECTIVES=[]
ACTIONS=[ActEvictLand "P"]
OUTBOX=[Message "SheriffAg" "A" (Complaint "A" "B" "P") Accept_Execution
NoMessage])
```

In world state 9 the sheriff agent receives the message containing the application He updates his internal state and generates the duty to do the judgement execution. He performs the physical action ’evict land’. Additionally he sends a message to the authorized user with the information that he accepts the application and evicted the land from the unauthorized user.

World state 10

```
Agent( AID=A
INBOX=[Message "SheriffAg" "A" (Complaint "A" "B" "P") Accept_Execution NoMessage]
PHYPERCEPS=[]
STATUS=(Applicant_Execution "A" <m> 8) (JudgementExecutor "SheriffAg" <m> 8)
(ExecutingJudgement <m> 8) (Loser_Complaint "B" <m> 6)
(Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6)
(Defendant "B" <m> 2) (Plaintiff "A" <m> 2 )  (Judge "CourtAg" <m> 2)
(Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0)
(RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0)
(SheriffAg "SheriffAg" 0) (Parcel "P" 0)
DUTIES=[]
OBJECTIVES=[]
```

```
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""
```

In world state 10 the execution applicant "A" receives the information that the judgement was executed. The piece of land "P" is again unused. The simulation ends.

8.4 Assessment of the results

The flow of operation in the simulations captures the social processes we introduced in section 4.7. We represented social processes in reality, the interaction between human beings and their relation to parcels, in terms of the interaction between agents and the change in the internal states of the agents.

The model correctly represents the social processes in reality (with respect to the two case studies). Based on the simulation we can asses that the model we constructed in chapter 7 and tested by agent-based simulation in this chapter is valid with respect to the two case studies we used: the transfer of ownership of a parcel between two persons and the conflict between to persons regarding land use and its resolution by the organizations of the state.

By comparing the flow of operation in the simulation with the situation in reality represented by the two scenarios we can asses that the simulation corresponds to the processes in reality with respect to the two case studies. Because we selected characteristic and typical case studies capturing a broad variety of processes and assume that other processes are quite similar to the two we presented, *we conclude that the computational agent-based model based on the ontology institutional reality developed in chapter 3 correctly and successfully represents reality in a cadastre.*

8.5 Summary

The goal of this chapter was the validation the agent-based model we constructed in chapter 7 with respect to the reality of the Austrian legal system we discussed in chapter 2.

We selected two characteristic social processes from reality, namely the transfer of ownership of a parcel between two persons, and conflicts and their resolution in the case of land use. We discussed both simulations in detail. We presented the input and the output of the simulation. The input of the simulation consists of

- representations of subjects, objects and events: land pieces, agents and messages they exchange.
- representations of the initial institutional situation, by the initial beliefs of the agents.
- representations of constitutive rules as operations in the agents' decision making process.

The output of the simulation consists of a complete history of the world states the simulation produced. Based on the input and the output we described the flow of operation during the simulation.

The chapter has two major results: First it has shown the correctness of the model with respect to the two case studies we investigated. From this fact we derive the second, more general result: We conclude that the ontology of institutional reality we developed in this thesis is correct for the field of cadastre. Reality in a cadastre can be analyzed based on the following three basic assumptions:

- Institutional reality can be explained in terms of the individual knowledge and interaction of the participating actors.
- Searle's concepts of institutional and physical facts characterizing status functions assigned by constitutive rules to physical phenomena (subjects, objects, events) are the core element determining the knowledge and the interaction of these actors.
- The monopoly of force determines the structure of institutional reality in a cadastre. It explains how particular actors in the society define institutional rules, apply these rules and enforce the consequences of these rules.

Chapter 9

Application of agent-based simulation: Assessing costs

9.1 Introduction

In this chapter we outline the ideas of the application of agent-based simulation. Simulations can be used to assess costs of processes. It is possible to count the individual costs related to the activities of each agent within the simulation of these processes. Counting the costs of each agent allows investigating the distribution of the costs between the participating agents.

For the cadastre the cost assessment is one foundation for comparing different systems. If it is possible to analyze costs of similar processes in different systems, the comparison of the systems can be done on an objective basis. The cost assessment allows investigating the internal distribution of the costs, in particular between the private parties acting (the legal persons performing legal transactions) and the public authority managing the activities (in particular the cadastral registry).

Costs arising in the interaction between the cadastral registry, legal persons and other actors are mainly transaction costs (North 1997). The approach of this chapter is the analysis of transaction costs in the field of cadastre.

Within the agent-based simulation model the cost analysis is based on the distinction of the following three sorts of costs:

1. decision costs, which are internal costs,

2. activity costs, which are external costs,

3. perception costs, which are external costs.

Decision costs are related to the decision making process of each agent about activities to perform. In this sense they are internal to each agent and represent the costs of its internal activities. Activity costs and perception costs are related to the activities of the agent in reality. In this sense they are external to the agents.

The chapter starts with the investigation transaction costs in the field of cadastre. It continues with the discussion of perception, decision and activity costs. In the following sections we integrate the cost assessment into the model. We implement the cost assessment based on two different technical solutions. Next we apply the ideas to the simulation of the case studies from chapter 8. A short summary concludes the chapter.

9.2 Transaction costs

North observes that it is costly to interact and that traditional (micro-) economic theories do not deal with these costs (North 1997). He discusses that institutions exist in order to reduce the costs of transactions. The main criterion for the measurement of the efficiency of institutions is the question how they effectively reduce the cost of transacting in relation to the cost of the maintenance of the institutions themselves.

North distinguishes two sorts of transaction costs: measurement costs and enforcement costs. First it is costly to measure the valuable attributes of what is being exchanged. The costs to acquire this necessary information are called measurement costs. The second sort of transaction costs is related to the enforcement of agreements. Efficient exchange of commodities (or exchange at all) can only happen, if individuals do not cheat. In modern society a third party is involved in the exchange, which enforces individual agreements, the state.

Institutions reduce uncertainty in the exchange by limiting the choices people have. The support of measurement reduces the uncertainty about the properties of commodities. The enforcement reduces the uncertainty about the behaviour of the people involved in the exchange.

Third party enforcement means the development of the state as a coercive force able to monitor property rights and enforce contracts. The monitoring of property rights, which is necessary for the enforcement, is a third source of transaction costs. We identify three sorts of transaction costs:

- *measurement costs*,
- *monitoring costs*,
- *enforcement costs*.

In oder to analyze the transaction costs of a system it is necessary to discuss both, the costs of the institutions, of their maintenance and transactions, and the cost arising for the individuals and their interactions. We perform the analysis by distinguishing the public costs (of the state) from the private costs (of the individuals).

9.3 Transaction costs in a cadastre

The efficiency of a cadastral system as the system of institutions dealing with the exchange of real estate, can be assessed by the analysis of transaction costs occurring between the actors within the system. The overall costs of the system consist of *costs independent of transactions* and *transaction dependent costs*. Transaction independent are the costs of the definition of the property rights and the creation and maintenance of the organizations according to the institutions. Transaction independent costs are transferred to the individuals and organizations of the society by taxes. Transaction dependent costs are related to the activities of the actors, such as, ownership transfer and conflict resolution. Transaction independent costs are beyond the scope of this analysis. We focus on transaction dependent costs. We discuss transaction dependent costs related to the two case studies under consideration: ownership transfer and conflict resolution. In the following we investigate public cost for the representatives of the

	private costs	public costs
		creation and maintenance of the institutions (transaction independent costs)
ownership transfer	exchange costs	
	measurement (queries)	measurement (answer queries)
	monitoring (application of ownership transfer)	monitoring (registration)
conflict resolution	enforcement (complaint, application of judgement execution)	enforcement (judgement, execution process)

Table 9.1: The cost analysis of a cadastre

state, i.e., the cadastral registry, courts and sheriffs and the private costs for individuals, i.e., legal persons. Transaction dependent costs are transferred to the individuals by charging the participants of the transactions.

Analyzing costs in the interaction of actors in the cadastre we must distinguish *exchange costs* from transaction costs. Exchange costs are related to the exchange itself, not to acquiring information, monitoring and enforcement. In the case of ownership transfer, exchange costs occur by the contracting of private persons consisting of the making of an offer and the acceptance of this offer.

The registration process in the registry is related to monitoring costs, i.e., to the monitoring of the legal situation by the state. These costs have a private and a public part. The private person applying for the ownership transfer is charged by the cadastral registry. For the cadastral registry costs arise from the checking of the validity of the transfer, from the changing of the registry and from the delivering of its decisions to the legal persons involved.

Measurement costs in the field of cadastre are mainly connected to the determination of the legal properties of a parcel, in particular, its legal status. Measurement costs are related to the queries and answers of the registry according to ownership of a parcel. These costs are distributed between the registry and private legal persons. For the cadastral registry costs arise from the processing of the query and the generation of the answers to the query. For a private person costs arise from the query for which he is charged by the cadastral registry.

Enforcement costs are related to the conflict resolution. The private costs consist of the complaint and of the judgement application by the suing person. The public costs are distributed between the court for the legal proceeding and the sheriff for the judgement execution. Table 9.3 shows cost analysis of the cadastre according to the two case studies.

9.4 Perception costs, decision costs and activity costs

For the simulation of the cost distribution in the cadastre we introduce perception, decision and activity costs. The idea is to express and represent the different transaction costs occurring during the simulation in terms of these three sorts of costs. These costs can be positive, i.e., they takes resources or negative if they provide resources.

Agents perform physical activities in their environment to which we can assign costs. Their physical activities take time and perhaps an amount of money, for instance, if the person is employed and must be paid. Additionally costs can be assigned to the communication actions between the participants. which are time-consuming and involve other costs, such as mail costs. These costs are always related to the activities of agents. They can be characterized as *activity costs*.

In the cadastre activity costs of legal persons can be, for instance, the costs of a person sending an offer to another person or the cost to perform an application at the cadastral registry. The costs of the state in this situation are, for instance, the performance of ownership transfer, i.e., the registration of the transfer in the registry. Costs can be assigned, for instance, to the physical enforcement of rights by activities of the sheriff.

Costs can also be assigned to the percepts of agents. For instance, if an agent sends a query to the registry agent, this is a positive cost for the sender, who has to pay for the query. For the registry agent, who receives the payment, the cost is negative. These costs we designate as *perception costs*.

Common to these activities is that they are external to the human being in the sense that they change the environment external to the agents or provide external influences to the agents. We designate these costs as *external costs*.

Costs can be assigned to the decision making process of persons and, to a much larger extent, to the decision making process of organizations. These costs are different to the external costs in the sense that they are internal to the human being or to the organization. They do not affect the environment. We designate this kind of cost as *internal costs*. Internal costs are related to the decision making process of the human being or organization. They can be characterized as *decision costs*.

Internal costs are of particular importance for the cadastre because the decision making process of the registry as well as its efforts to maintain the rights in the registry can be characterized (and represented) as internal costs. Also the process of the court to decide a complaint can be characterized by internal costs. Figure 9.1 shows the conceptual ideas.

9.5 Assumptions about the cost distribution

In order to simulate the cost distribution in the model we have to make assumptions about the individual costs occurring during the interaction of the agents. The detailed economical analysis of these costs in reality is beyond the scope of the thesis. Our scope is to show how to simulate and assess costs based on these basic assumptions. Here we make assumptions about the costs, which are not economically justified.

We discuss the simulation of the cost distribution according to the case study of ownership transfer. We assume that costs correspond to the time actors need to make their decisions and to perform activities. Thus we give the amount of working

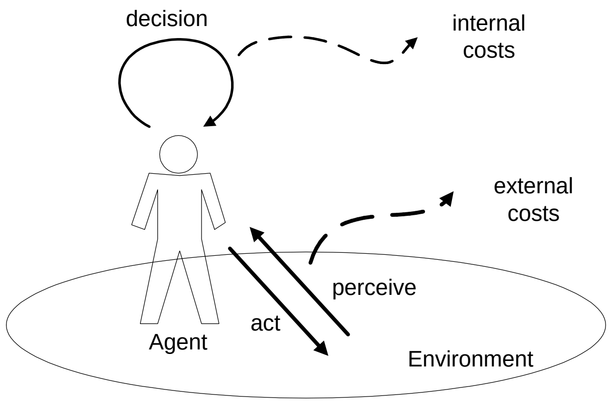

Figure 9.1: Internal and external costs in the agent-based model

hours needed. We assume writing and processing of communication messages, which are usually letters in the field of cadastre, takes two working hours, so we assign the cost 2 to a communication action. Physical activities, where persons act somewhere outside are more costly than communication activities. We assign the cost 5 to physical activities. Physical and communication percepts are cost less. We assign the cost 1 to the decision making process of each agent. Exceptions from these general assumptions are the application of ownership transfer and a query concerning the ownership of a parcel. The action 'application of ownership transfer' costs 10, the action 'query' costs 4. Accordingly the perception of an application or query provides costs of -10 and -4, i.e., it provides resources to the registry agent which correspond to the cost of 10 and respectively 4 working hours.

We assume that the organizations of the state make no profit. That means that the amount of money they charge for their activities are exactly the costs arising by performing these activities. We assume that, if assigned to an application, the cost is of 10 for the applicant, so the cost for registering this ownership transfer is also 10, i.e., that the process of registering ownership transfer costs 10 working hours. Figure 9.2 shows the cost distribution based on these basic assumptions according to the example of ownership transfer.

9.6 Counting costs

This section discusses the realization and implementation of the different types of costs. In the model of reality in a cadastre (see chapter 7) activities of the agents are realized according to the message exchange model. That means actions and percepts are messages, i.e., data structures. Communication actions and percepts are represented by data structures of type $Message$ and physical actions are represented by data structures of type *Action* (see subsection 7.9.2). Physical percepts are realized by data structures of type $PhyPercept$.

We assign costs to each of these data structures. This allows the assignment of costs to each type of activity. Consequently we assume that costs correspond to the type of the messages.

To calculate the internal costs, the approach of assigning costs to data structures is not usable. The decision costs are related to the decision operations. It is necessary

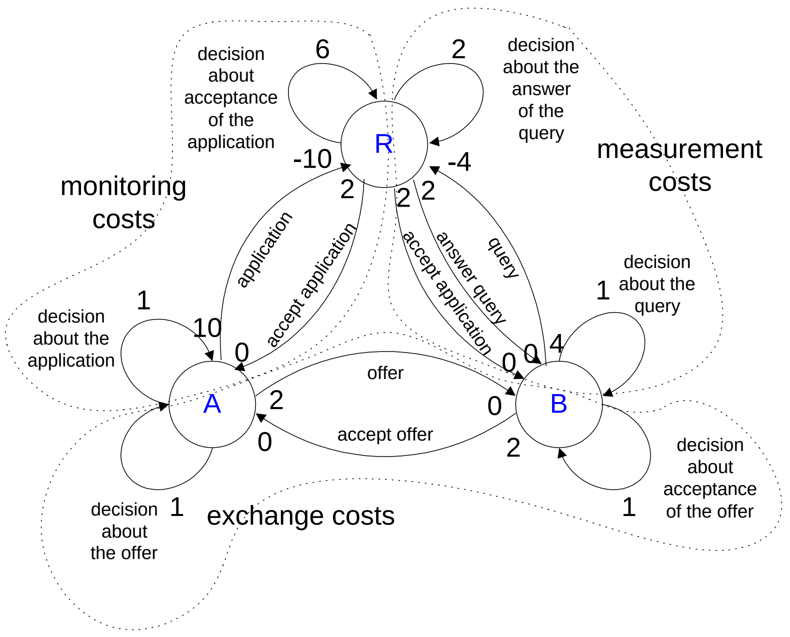

Figure 9.2: Basic costs in the ownership transfer based on the cost assumptions

to assign costs to operations. We choose the following solution: We lift each decision operation corresponding to a duty or objective and assign a cost as part of the lifted function result (see subsection 6.6.3 for the description of the lifting operation).

In each world state we calculate the costs for each agent and store it as part of the agent data structure. After the simulation ends we collect and summarize the costs from the world history and present it as part of the output of the simulation.

The costs are represented in the model as integer values. Costs can be positive or negative. The following data structure represents costs in the model:

type *Cost* = **Int**

According to the analysis in each world state for each agent we can identify exchange costs, monitoring costs, measurement costs and enforcement costs. These sorts of costs are represented by the following data structure:

data *CostData* = *CostData Cost Cost Cost Cost*

In each world state we summarize the sorts of costs and store them as part of the agent data structure in order to enable the analysis of the cost distribution based on the information stored in the output of the simulation, the world history. The agent extended by the cost data is a data structure of the following form:

data *Agent* = *Agent AgentID* [*Percept*] *IntState* [*Act*] *CostData*

After the end of the simulation we can analyze the cost distribution for the different costs for each agent by the following operation:

getAbsAgentCosts :: [*World*] → [(*AgentID*, *CostData*)]

The operation calculates the individual costs for each agent based on the output of the simulation. In the next two subsections we investigate how to calculate the individual costs for each agent during the simulation.

9.6.1 Counting external costs

Agents put their activities into the outbox after each activity cycle. In each world state the percepts are provided to the agents in their inboxes. The approach we use here is to collect all percepts and activities that occurred during one simulation cycle from the agent data structure and assign costs to each activity.

This is implemented in the model in the following way: The class *ExtCosts* (listing 9.1) defines operations that assign costs c to each activity or percept e and count the four sorts of costs based on a list of activities or percepts.

Listing 9.1: Specification of external cost assessment

```
class Num c ⇒ ExtCosts e c | e → c where
    cost :: e → c
    gesCost :: [ e] → c
    isExchangeCost :: e → Bool
    isMonitoringCost :: e → Bool
    isMeasurementCost :: e → Bool
    isEnforcementCost :: e → Bool
    genExchangeCosts :: [e] → c
    genMonitoringCosts :: [e] → c
    genMeasurementCosts :: [e] → c
    genEnforcementCosts :: [e] → c

    gesCost [] = 0
    gesCost e = foldl (+) 0 (map cost e)
    genExchangeCosts = gesCost. (filter isExchangeCost)
    genMeasurementCosts = gesCost. (filter isMeasurementCost)
    genMonitoringCosts = gesCost. (filter isMonitoringCost)
    genEnforcementCosts = gesCost. (filter isEnforcementCost)
```

In order to assign costs to particular physical and communication messages we define instances for percepts and actions. These instances define the basic perception and activity costs. The implementations 9.2 and 9.3 are both incompletely presented to express the ideas. For the complete code see appendix A.

According to the assumptions made in section 9.5 we assign the cost 5 to the physical activity $ActUseLand$. We assign the cost 2 to the communication action $Offer$ and assume that an application (of ownership transfer) costs 10 and a query to the cadastral registry costs 4. The action $ActUseLand$ belongs to exchange costs as well as the offer. The application action is a monitoring cost and the query action belongs to measurement costs. The listing 9.2 shows the implementation.

Listing 9.2: Implementation of external costs for activities

```
instance ExtCosts Act Cost where
    cost (PAct (ActUseLand _ _ )) = 5
    cost (CAct (Message aid aid' content Offer rid)) = 2
    cost (CAct (Message aid aid' content Application rid)) = 10
    cost (CAct (Message aid aid' content Query rid)) = 4
    isExchangeCost (PAct (ActUseLand _ _ )) = True
    isExchangeCost (CAct (Message aid aid' content Offer rid)) = True
    isExchangeCost _ = False
    isMonitoringCost (CAct (Message aid aid' content Application rid)) = True
    isMonitoringCost _ = False
    isMeasurementCost (CAct (Message aid aid' content Query rid)) = True
    isMeasurementCost _ = False
```

Assuming that in general no costs are assigned to percepts of agents the perception of a land use has the cost 0. Receiving an application or a query causes a negative cost of −10 or −4. Applications belong to monitoring costs and queries belong to measurement costs. The listing 9.3 shows the implementation.

Listing 9.3: Implementation of external costs for percepts

```
instance ExtCosts Percept Cost where
  cost (PPercept (LandUse _ _ _)) = 0
  cost (CPercept (Message aid aid' content Application rid)) = −10
  cost (CPercept (Message aid aid' content Query rid)) = −4

  isExchangeCost _ = False
  isMonitoringCost (CPercept (Message aid aid' content Application rid)) = True
  isMonitoringCost _ = False
  isMeasurementCost (CPercept (Message aid aid' content Query rid)) = True
  isMeasurementCost _ = False
  isEnforcementCost _ = False
```

For an agent it is now possible to calculate exchange, monitoring, measurement and enforcement costs with the operation *genExtCosts* (listing 9.4) which summarizes the different costs for percepts and actions.

Listing 9.4: Generation of external costs

```
genExtCosts :: Agent → Agent
genExtCosts (Agent aid percepts intstate actions (CostData ex mon mea enf)) =
  (Agent aid percepts intstate actions (CostData ex' mon' mea' enf')) where
     ex' = ex + (genExchangeCosts percepts) + (genExchangeCosts actions)
     mon' = mon + (genMonitoringCosts percepts) + (genMonitoringCosts actions)
     mea' = mea + (genMeasurementCosts percepts) + (genMeasurementCosts actions)
     enf' = enf + (genEnforcementCosts percepts) + (genEnforcementCosts actions)
```

9.6.2 Counting internal costs

Internal costs can be calculated by assigning costs to the decision making process of each agent. Applying the idea of lifting to this issue we change the decision operation of the agents in a way that it includes cost information about the operation. We define the parameterized data type *OpCosts* that adds cost data to a parameter f:

```
  data OpCosts f = OpC f CostData
```

The task is now to lift operations that work with parameters f to operations that work with the data type *OpCosts f*. We define the lifting operation in the following way:

```
  liftCost :: f → CostData → OpCosts f
  liftCost f cost = ( OpC f cost)
```

It takes an argument f and a cost data structure *CostData* and lifts it to the data type *OpCost f*, i.e., it wraps the cost data around the parameter f. According to the lift operation introduced in subsection 6.6.3, this is the lifting operation of an operation of arity 0, i.e., the lifting of a constant (here f). We are able to use this simplified version of the lifting operation, since we assume that the cost of the operation does not depend on its parameters, i.e., no costs are assigned to these parameters. This is a limitation of the realization here, not of the lifting approach.

We assign the costs to operations that represent the agents' decision making process. The lifted version of the decision operation *decision* (see listing 9.5 and compare to listing 7.2) has as result sets of activities (the activities the agent decided to perform) and cost information assigned ([(*OpCosts*[*a*])]). The decision making operation is defined in terms of the *selActsL* operation, which combines the decision making operation according to duties *actDutiesL* and objectives *actObjectivesL*. For each duty and objective these operations call the function *doDutyL* and *doObjectiveL*.

Listing 9.5: Specification of the decision operation including cost data

```
class (MyAgentInternalsL aid ws p i a, AgentInternals aid ws p i a) ⇒
   AgentInternalsL aid ws p i a where
      selActsL :: aid → ws → i → [( OpCosts [a])]
      decisionL :: aid → ws → [p] → i → ([( OpCosts [a])], i)

class ( UpdStateInMsgs aid ws p i, UpdStateOutMsgs ws a i) ⇒
   MyAgentInternalsL aid ws p i a | i→ aid, i→ws, i → p, i → a where
      actDutiesL :: aid → ws → i → [( OpCosts [a])]
      actObjectivesL :: aid → ws → i → [( OpCosts [a])]

class ActdutiesL s d aid ws a | a → d, a → s, a → aid where
   doDutyL :: [s] → aid → ws → d → ( OpCosts [a])

class ActobjectivesL s o aid ws a  where
   doObjectiveL :: [ s] → aid → ws → o → (OpCosts [a])
```

We assign internal (operation) costs to the lifted versions of the functions *doDuty* (see listing 9.6) and *doObjective* (see listing 9.7). For both operations we only give examples, the complete instance definition can be found in appendix A. The following part of the implementation of the *doDutyL* operation assigns the measurement cost 2 to the decision operation of an agent to answer a query and the monitoring cost of 1 to the decision operation of an agent to apply ownership transfer.

Listing 9.6: Lifted version of the *doDuty* operation

```
instance ActdutiesL Status Duty AgentID WorldState Act where
   doDutyL sl aid ws (AnswerQuery mid ws') =
      liftCost ( answerQuery sl aid ws mid) (CostData 0 0 2 0)
   doDutyL sl aid ws (DoApplication mid ws' ) =
      liftCost ( doApplication sl aid ws mid) (CostData 0 1 0 0)
```

The following part of the implementation of the *doObjectiveL* operation assigns the exchange cost 1 to the decision operation of an agent to make an offer.

Listing 9.7: Lifted version of the *doObjective* operation

```
instance ActobjectivesL Status Objective AgentID WorldState Act where
   doObjectiveL sl aid ws ( SellParcel land aid' ws') = erg where
      erg = if (ws==ws') then liftCost (doOffer sl aid aid' land ws) (CostData 1 0 0 0) else
         (OpC [] (CostData 0 0 0 0))
```

9.7 Assessing the results of the simulation

We are able to simulate the cost distribution by assigning basic costs to percepts, decisions and actions of the agents. The approach we described gives a powerful tool for the analysis of transaction costs occurring in social processes of a cadastre. We focused on the description of the core ideas and made idealized assumptions about the basic costs. We used the transfer of ownership as example to simulate the cost distribution.

9.7.1 The transfer of ownership

Applying the mechanism introduced above we get the following results for the simulation of ownership transfer:

```
 ********Costs*******
["(RegAg,Exchange=0,Monitoring=0,Measurement=0,Enforcement=0)",
 "(A,Exchange=3,Monitoring=11,Measurement=0,Enforcement=0) ",
 "(B,Exchange=3,Monitoring=0,Measurement=5,Enforcement=0) "]

Sim>
```

Since we assume that the registry charges applications and queries to the extent of its costs, the overall costs for the registry agent are 0. These costs are transferred to the private persons involved in the exchange. A has exchange costs of 3 consisting of the decision cost to make an offer 1 and of the activity cost of sending this offer 2. Its monitoring costs consist of the decision to apply for ownership transfer 1 and of the activity of applying ownership transfer 10. B has the exchange costs of 3 consisting of the decision to accept the application 1 and of the activity of accepting the application 2. It has the measurement cost 5 consisting of the decision to query the registry agent 1 and of the activity of querying 4.

9.8 Summary

In this chapter we outlined the ideas how two apply agent-based simulations to the assessment of costs. Costs arising in social processes in a cadastre are transaction costs. The basic idea of this chapter is to assess costs based on the individual interaction of the actors involved, i.e., to assign costs to the individual interaction. We identified different types of costs: transaction independent costs of the institutions and transaction dependent costs consisting of exchange costs, monitoring costs, measurement costs and enforcement costs.

In oder to simulate these costs in the model we distinguish internal costs and external costs of the agents. With these kinds of costs it is possible to independently discuss the cost of the decision making process of the agents and the costs of the agent's activities and percepts. With this approach we are able to simulate and assess costs occurring during transactions. The approach does not address the transaction independent costs of the system of institutions.

The degree of correspondence to the processes in reality relies on the assumptions about the basic costs assigned to atomic agents decisions, percepts and actions. We made artificial assumptions since the economical analysis of these costs is beyond the scope of this thesis.

We presented the principal idea of the application and assume that the model is extensible to the point that it is possible to discuss realistic systems. The evaluation of a cadastral system is one foundation for comparing systems regarding their costs and efficiency.

An important point is the distribution of the cost between private persons and public organizations. Both have to be assessed to evaluate the overall costs of a system.

The model developed allows the assignment of costs to all individual agents participating in the processes simulated. In this chapter we made the assumption that all costs arising in the institutions are charged to customers, i.e., to the private persons acting according to the institutional rules.

Chapter 10

Conclusions and future work

In this chapter we discuss the results and contributions of this thesis. We start with the summary of the content of the thesis. Then we discuss its major results and derive from them the contributions to different research fields. Additionally we discuss the restrictions of this thesis and conclude with an outlook on fields of future work.

10.1 Summary

This thesis is interdisciplinary work and has foundations in different scientific fields. These foundations are:

- Cadastre and legal background,
- Searle's theory of institutional reality,
- Multi-agent theory and Multi-agent systems,
- Algebraic specifications,
- The functional programming language Haskell.

Based on this background we developed the main content of this thesis, which we discussed distinguishing three stages of work:

1. *The ontological analysis of reality in a cadastre.* We introduced the *legal background of the Austrian cadastre* (chapter 2) and developed an *ontology of institutional reality* (chapter 3). We applied the results of both chapters to the *analysis of reality in a cadastre* (chapter 4).

2. *The construction of a computational model of reality in a cadastre.* We discussed the relevant aspects of multi-agent theory and multi-agent simulation and introduced an *abstract architecture of a multi-agent system* (chapter 5). As approach for the construction of the agent-based model and as agent programming language we introduced *algebraic specifications* and the *functional programming language Haskell* (chapter 6). We applied the abstract architecture to the *construction of model model* in Haskell (chapter 7).

3. *Agent-based simulation.* We performed the *validation of the model* with appropriate case studies by agent-based simulation (chapter 8). We discussed how simulations can be applied to the *analysis and assessment of costs* occurring in a cadastre (chapter 9).

We will discuss the three stages in the following.

10.1.1 Cadastre and legal background

We introduced the legal background of this thesis: the cadastre, i.e., the part of reality determined by the rules of the legal system relevant for the cadastre. We focused on a concrete system, namely the Austrian cadastral system. The Austrian legal system is an instance of a title registration system. Because it cannot be completely described without the discussion of the relationship between the cadastral system and the legal system in general we focused on the embedding of the cadastre into the general legal system.

10.1.2 The ontology of institutional reality

The ontology of institutional reality is mainly based on Searle's theory of institutional reality. Searle's theory describes reality as consisting of a physical part and an institutional part. The physical part is characterized by physical facts which correspond to subjects, objects, events and their relationships. The institutional part is characterized by institutional facts, which are constructed by the assignment of status functions to these physical phenomena. The building blocks necessary to explain the structure of institutional reality are the following:

- (physical) phenomena
- (institutional) status
- constitutive rules (defining under which conditions status is assigned to phenomena)
- (institutional) rights (connected to status) of agents to perform activities
- (physical) powers of agents to perform activities

We assume that the building blocks of institutional reality do not exist in external reality in the same way physical phenomena exist. Status, constitutive rules and rights exist only in the human mind. This leads to the following (individual-based) view of reality, where the building blocks of institutional reality are:

- (physical) phenomena
- beliefs of the individuals
- (physical) powers of the individuals

Searle assumes that collective intentionality is involved in the assignment of status functions, i.e institutional facts exists by human agreement or acceptance. We replaced collective intentionality by the monopoly of force, which gives particular organizations of the society the authority to create and change institutional facts. Thus collective intentionality is not involved in the creation of institutional facts. Collective intentionality is only involved in the creation of institutions, which exist by collective acceptance.

We simplified Searle's theory and assume that collective intentionality can be explained without presupposing a biological capability. Collective intentionality can be explained in terms of the single intentionality combined with a general economic principle. Collective acceptance of institutions is created if there is some economic benefit for the individuals, which causes the acceptance of the rules by the majority of the human beings.

The ontology of institutional reality developed in this thesis is characterized by the following three assumptions:

- Institutional reality can be explained in terms of the individual knowledge and interaction of the participating actors.
- Searle's concepts of institutional and physical facts characterizing status functions assigned by constitutive rules to physical phenomena (subjects, objects, events) are the core element determining the knowledge and the interaction of these actors.
- The monopoly of force determines the structure of institutional reality. It explains how particular actors in the society define institutional rules, apply these rules end enforce the consequences of these rules.

10.1.3 The analysis of reality in a cadastre

Reality in a cadastre

Starting the analysis of reality in a cadastre we defined what we understand by this term: By reality in a cadastre we understand the part of the real world, which is influenced by the activities and the content of the cadastral registry. It comprises the information system cadastral registry fully embedded into its environment. The environment consists of human beings and organizations and their interaction with each other and with objects, mainly land parcels, i.e., real estate.

Social processes in a cadastre

In the analysis we focused on two specific social processes in cadastral reality: the transfer of ownership of a parcel between two persons and the conflict between two persons and its resolution by the organizations of the state. The first process represents the normal flow of operation in the cadastre where people act according to the legal rules. The second process represents a conflicting case where people violate the legal rules.

Transfers of rights on parcels are the central element in the work of a cadastre. The first case study represents the ownership transfer of a parcel between two persons as the most important case of right changes. The second case study models conflicts, i.e., if a person does not respect the legal rules of the cadastre. It introduces an abstract notion of land use that can stand for a broad variety of activities the owner can perform on his parcel. The owner tries to use his parcel. A conflict occurs because an unauthorized person already uses the parcel. The owner sues against the persons and after the judgement he starts an execution to mobilize the power of the state to realize his right.

Three aspects in the analysis and their interrelationships

We analyzed reality in a cadastre based on the ontology of institutional reality. It was found that there are three essential aspects in the analysis of reality in a cadastre:

- Ontological categories of phenomena
- Levels of reality
- Facts and rules for their creation and existence

According of the ontology there exist three categories of phenomena in a cadastre: subjects, objects and events. Subjects are the active entities in the world that are able to perform actions. Actions cause events. Events are always based on the participating objects and subjects. Events change, create and destroy objects and subjects. Objects are passive entities in the world changed by events.

According to Searle's theory we distinguished two levels of reality: the physical and the institutional level. Facts at the physical level exist independent of human observers in external reality whereas facts at the institutional level are observer relative. Both levels are closely linked. Facts on the institutional level are always based on facts on the physical level.

We distinguished facts and rules for their creation and existence. Rules define the conditions for the creation and existence of facts. The possibility of facts must be distinguished from actually existing facts. Facts describe how the world is, i.e., a world state, whereas rules describe how the world can evolve, i.e., how it can change between world states. But both are part of reality and must therefore be part of the system.

The three aspects in the analysis of reality in a cadastre are interrelated. None of the aspects can be omitted. They represent different views on the issues in the cadastral domain. Every combination of the three aspects is meaningful. Objects, subjects and events exist on the physical level as well as the corresponding status functions on the institutional level. There are rules and facts for all categories on both levels of reality. In chapter 4 these interrelationships are represented in a systematic way.

Objects and subjects relevant for the cadastre

From the analysis follows that the phenomena relevant for the representation of reality in a cadastre are human beings, land pieces and the system of documentation. Status functions are assigned to these phenomena. We assume that the system of documentation only indicates status, i.e., no status is assigned to the documentation.

10.1.4 The abstract architecture of a multi-agent system

We presented an abstract, i.e., domain independent, architecture of a multi-agent system intended to be applicable to the representation of the cadastral domain. We started with definitions of the terms 'agent' and 'multi-agent system' suitable for our task. We regard the agents as part of the environment, i.e., the environment is defined as the set of all objects and agents in the system.

As an agent we regard an entity which is capable of acting in its environment and perceiving its environment. A multi-agent system consists of four components. The first component is the environment consisting of a set of objects. A subset of the objects are agents. The second part is a set of relations between objects. The third part is a set of operations enabling activities of the agents', in particular the agents' actions and decision operations. The fourth part of the system is called laws of the universe which represent the reaction of the environment to the agents activities.

We presented the agent architecture from an operational point of view, i.e., we characterized it based on the structure of the operations the agent performs. According to the structure of the decision making process we distinguished agents without internal state from agents with internal state. Agents without internal state just map percepts to actions based on condition-action rules. An agent with internal state has explicit representations of the environment in its current state and of former states. We assume that the decision making process of an agent with internal state follows the Sense-Plan-Act paradigm. It consists of three steps:

1. update the internal state based on the percepts of the agent,

2. decision based on the internal state which action to perform,

3. update the internal state with the knowledge about the action performed.

A specific class of agents with internal state are goal-based agents. The internal state of the goal-based agent contains additionally to the representations of the current and former world states, representations about desired world states, i.e., world states the agent tries to achieve.

10.1.5 Algebraic specifications in Haskell

The fundamental assumption of this thesis is that the real world can be described in terms of algebras, i.e., by sets and operations between sets. These algebras can be described on an abstract level by algebraic formal specifications. On the level of the specification we can determine the desired properties of the algebras, which are the models of the real world. The algebraic specifications embody our assumptions about the real world.

Expressing algebraic specifications in the functional programming language Haskell has the advantage that we are able to construct computational models, which form algebras that have the desired properties of the corresponding algebraic specifications. This means that the computational model, which is formally correct with respect to the specification, embodies our assumptions about the properties of the real world.

Haskell provides support for the development of computational models of algebraic specifications by the concepts of specification, representation and implementation. The specification is the realization of algebraic specifications in Haskell and is expressed by classes. Representations, i.e., data structures, define the carrier sets of the algebras. Implementations define the functions between these carrier sets and are realized in Haskell by instances.

10.1.6 The construction of the agent-based model

The construction of the model follows the *agent-based conceptualization approach.* It uses an agent-based model as conceptual framework not as paradigm for the construction of more efficient software systems or of intelligent artificial systems.

For the realization a state based model was selected. The world changes from one state to another starting with the initial state 0.

The model construction process follows the algebraic approach and defines specifications of the models and gives representations and implementations for the operations of the specification.

Specification

The specification part translates the abstract agent architecture into a Haskell specification and defines the execution model of the system. We distinguish the agent level and the world level of the execution model. Both together form the execution cycle of the system in each world state.

The agent level corresponds to the operations defining the activities of the agents, i.e., to the operations of the agents to modify their environment, in particular to interact with each other. The agent level defines the activity cycle of each agent in one world state. It implements the three step structure of the decision making process of the agent.

The world level corresponds to the reaction of the environment to the agents' activities and implements the laws of the universe. The operations on the world level consist of three steps:

1. Perform the (physical) activities of each agent, i.e., realize physical events based on the agent's activities. If activities fail, generate percepts to the acting agents providing the information that its action failed.

2. Send the messages of each agent to the addressees, i.e perform communication events based on the agent's activities.

3. Call the activity function of each agent.

Representations

Applying the analysis of reality in a cadastre, the model has to contain representations of the agents and of the objects, which together form the environment. Additionally the representation of the system of documentation is necessary. We assume that documentation is sufficiently modeled by constructing one particular agent, which represents the cadastral registry and maintains the system of documentation by its internal beliefs.

The agents. The state of an agent is characterized by its percepts, its internal state and its actions. We assume that the basic model of interaction in the model is message exchange. Consequently we assume that percepts and actions of the agents are (different types of) messages. Since we distinguish physical and communication activities we distinguish physical and communication messages.

This interaction model motivates a structure of the agents consisting of an inbox, an internal state and an outbox. The inbox contains the percepts of the agent in the current world state, its outbox contains all activities the agent decided to perform in the current world state.

The agent's internal state. Since we assume that a goal-based agent architecture is necessary to represent agents capable of acting in reality in a cadastre, the agent's internal state consists of representations of its beliefs about the current state of the world and of goals. The representation of the current state of the world comprises the agent's beliefs about subjects and objects existing in the world and events occurring as well as of the institutional status assigned to these phenomena. Goals are related to rights, which are connected to institutional status. We distinguish between goals the agent tries to achieve based on its own possibilities given by its rights and goals the agent tries to achieve based on its obligations. These types of goals we designate as objectives and duties.

We have shown that duties and predefined objectives are sufficient to represent the social processes under consideration. Positive rights and the free decision of activities under these rights, i.e., goal generation, has not been implemented. A set of predefined objectives were sufficient for the model.

The environment Following the analysis, the category of objects relevant for the cadastre are pieces of land. The model has to comprise representations of pieces of land and of agents. We model pieces of land by their identifiers. We model the environment by a data structure *world*, which consists of a set of agents and a list of land pieces.

Implementation

The implementation provides the realization of the operations from the specification on the specific representations. The aspect of the implementation, which is of central relevance for the model of reality in a cadastre, is the transformation of constitutive rules into the model. We implement constitutive rules as operations updating the internal state of the agents. These operations map some phenomenon represented in the model and a specific context to status beliefs and duties assigned. Status beliefs are created by communication actions between agents, i.e., assigned to communication messages the agent receives.

10.1.7 Validation of the model by agent-based simulation

According to the two case studies the validation of the model means to define the input of the simulation and to assess its output. The input comprises the agents, their internal states and the realization of the constitutive rules. The output of the simulation is the set of all world states, i.e., the world history.
The simulation of the two case studies has to comprise the following agents as input:

- the registry agent
- the court agent
- the sheriff agent
- the seller and the buyer of a parcel
- the authorized user and the unauthorized user of a parcel

The registry agent represents the cadastral registry and its behaviour. The court agent models the work of the court during the legal action. The sheriff agent represents the power of the state with its monopoly of force. Seller, buyer, authorized user and unauthorized user are representations of legal persons acting in reality.

The assessment of the results of the simulations allows the following conclusions: First, the correctness of the model has been shown with respect to the two case studies we investigated. From this fact we derive the second, more general result: We conclude that the ontology of institutional reality we developed in this thesis is correct for the field of cadastre.

10.1.8 Agent-based simulation for the assessment of costs

We applied the simulation model to the assessment of costs in the cadastre. We assume that the costs occurring in the cadastre are mainly transaction costs. The basic idea of the approach is to assess transaction costs based on the individual interaction of the actors involved, i.e., to assign costs to the individual interaction. We identified different types of costs: transaction independent costs of the institutions and transaction dependent costs consisting of exchange costs, monitoring costs, measurement costs and enforcement costs.

In oder to simulate these costs in the model we distinguish internal costs and external costs of the agents. With these kinds of costs it is possible to independently discuss the cost of the decision making process of the agents and the costs of the agent's activities and percepts. With this approach we are able to simulate and assess costs occurring during transactions. The approach does not address the transaction independent costs of the system of institutions.

10.2 Results

It is possible to explain social reality in a cadastre in terms of the individual interaction of the human and organizational actors. We applied Searle's theory of institutional reality to the field of cadastre, but differed from his theory in one crucial point: We are able to explain the structure of reality in a cadastre without presupposing collective intentionality as biological property of human beings. Insofar we made less assumptions than Searle's theory only presupposing the individual capabilities of human beings to assign status to phenomena, to define institutional rules and to negotiate these rules. Social reality is a property of the whole system that emerges from the interaction of the individuals in the society.

It is possible to construct a computational model of a cadastre based on the ontology of institutional reality. Combined with the first point, this is the main result of this thesis. It has three aspects. First the ontology of institutional reality based on Searle's theory allows computational model construction. Second the ontology is sufficient and powerful enough to represent a complex part of reality, a cadastre. Third the fact that we successfully constructed and validated the model allows the conclusion that a theory of the institutional part of social reality is sufficient to explain the structure of reality in a cadastre.

It is possible to represent social processes of a cadastre. We developed a framework for the simulation of social processes of reality in the model and tested it by representing two non-trivial cases of processes from cadastral reality.

The extension of the scope from the cadastral registry to *reality in a cadastre* is helpful for the analysis of the cadastral domain. We have seen that this view on the cadastral domain allows the discussion of a broader variety of issues, because they often occur outside the registry, but nevertheless with strong impact to the cadastral system.

It was necessary to model social reality in an agent-based framework. The model construction based on Searle's theory was only possible with an appropriate representation of human intentions and behaviour. Agents were the architecture used for this purpose. The agent-based approach is useful to construct models of parts of the real world. This shows the potential of agent-based models for the investigation of social reality.

The *agent-based conceptualization approach* has been successfully applied. We have shown the usefulness of an agent-based approach as a purely conceptual framework independent of the formalization and implementation of the model. Using a formal language we focused on models of the real world, not on the formal properties of the framework. We called this the *agent-based conceptualization approach.*

The algebraic approach has been successfully applied. The assumption that it is possible to model the real world in terms of algebra was an essential foundation of the model construction process. It allows the conclusion that the algebraic approach captures important aspects of reality.

The analysis of reality in a cadastre has shown its correspondence to reality. The analysis based on the distinction of the three essential aspects of reality was the foundation for the construction of the model. We have shown the usefulness of the analysis by its application to the model, which correctly represents processes in reality. This allows the conclusion that for the cadastral domain the analysis performed here corresponds to the reality it describes.

Functional specifications lead to a clear and understandable representation. We have shown the applicability of functional specifications to agent-based models. Functional languages are operation oriented and therefore in particular suited for the description of change. An agent-based approach focuses on the activities of agents. Thus the description of change plays a crucial role. The suitability for representing change makes functional languages to an appropriate tool for agent-based models.

A method for cost assessment has been implemented. We outlined the idea of the application of simulations to assess costs and cost distribution between the agents. We implemented an example application based on the case studies of this thesis for the assessment of transaction costs in a cadastre.

The foundation of a simulation environment for the evaluation of real-world cadastral systems has been developed. The model presented here is extensible to more comprehensive parts of the cadastral law, which can be expressed in the proposed framework. A possible application of this environment is the evaluation of cadastral systems regarding their stability to fraud and mistakes by agent-based simulation of critical cases. By agents it is possible to simulate the behaviour of people making mistakes or cheating.

10.3 Contributions

There are contributions to several scientific fields and we will discuss them for each field. A general contribution is the union of work from very different scientific fields. This shows the connections of these fields and the importance and power of interdisciplinary work.

Contributions to philosophy. The successful application of our ontology of institutional reality based on Searle's theory to the construction of a computational model of the cadastral domain contributes to research in the ontology of social reality. It has been shown that a (computational) model generation based on this theory is possible. The successful validation of the model by simulating processes from a real world cadastre supports the conclusion that our ontology correctly expresses the structure of institutional reality. Thus Searle's theory is an important approach to improve our understanding of the world around us. We improved his theory by making less assumptions about the real world. We omitted the presupposition of collective intentionality as biological capability of human beings.

We have seen the practical dimension of Searle's theory because it was the crucial foundation for the application in the cadastral domain. This work shows the usefulness and importance of doing "practical" ontology. Ontologies of specific domains, such as the cadastre, can improve the design process of information systems. Domain ontologies contribute to solve specific tasks and we have demonstrated this in the cadastral domain.

Contributions to social science. Cadastral systems are a separated part of reality but nevertheless belong to social reality. We contributed to social science by the successful simulation of social processes in a cadastre. This shows the applicability and importance of agent-based social simulation to the investigation of social reality. We analyzed the cadastre as part of social reality on a sophisticated formal foundation. We have seen the relevance of computational and formal model construction as a useful tool to validate theories about social concepts. This supports the assumption that it is helpful and possible to investigate a broader variety of phenomena in social reality on a formal foundation based on computer simulations.

Contributions to multi-agent theory. We contributed to multi-agent theory by the development of an abstract architecture of a multi-agent system and by the application of the agent-based approach to the model of a cadastre. The power of agent-based models for the representation of social reality, as well as the solution of practical tasks has been shown. Agents were the key element in the representation of human intentions and behaviour. We introduced the agent-based conceptualization approach. The key idea is the understanding of agent-based approaches as domain independent conceptual framework for the description of phenomena in reality. It carefully separates the conceptual model from the formalization and possible application in software development.

Contributions to the legal domain and the cadastre. The contribution to the legal domain in general and the cadastral domain in particular has a theoretical and a practical dimension. First we investigated the cadastre embedded into its environment and developed a framework for the computational representation of the cadastral law based on Searle's theory. This framework is extensible to larger parts of the cadastral law and perhaps to other legal domains. Second we developed the foundations of a simulation environment for the evaluation of cadastral systems. Based on the evaluation the efficiency of a real world cadastre can be further improved.

Contributions to the algebraic specification and functional programming domain. At the Institute for Geoinformation the use of functional specification for the model and information system design has been proposed. The benefits of this approach have been shown in several domains. This thesis supports this view because it applied a functional specification for the construction of an agent-based model based on a philosophical theory and for social simulation.

10.4 Restrictions

Due to the lack of space and time and the complexity of the domain this thesis constrains its focus in several respects.

The model only represents Austrian law. We decided to use a concrete system to reach a more realistic model. Neither the structure of the ontology of reality in a cadastre nor the agent-based model are affected by this constraint. Only the input of the simulation, the objects and agents with their status beliefs and constitutive rules have to be adapted to the specific domain. This has no effects on the general structure of the approach and is therefore no constraint on its generality.

The thesis omits the discussion of spatial problems. This thesis is mainly interested in the change of the legal status in the cadastral domain. This can be described without regarding spatial change of parcels. Nevertheless there are interesting issues connected to the physical shape of boundaries and their institutional status. These issues have a strong impact on the cadastral system. They are excluded here from the discussions and left to future work.

The thesis assumes a static system of institutions. We investigated change and processes within the institutions, not the change of the institutions themselves. Topic of this theses was to model how people interact according or violating the legal rules of a cadastre. The cadastre is an existing system of institutions, which can be modelled assuming that the institutional rules do not change.

The thesis assumes that a functioning system of institutions exists. This assumption does not hold in any case. Countries of the third world do not always have a working land registration system, i.e., a system of land exchange, which is determined by a functioning legal system. In such a case not all aspects determining reality can be derived from the legal rules, so the analysis of the cadastral law, as it was the foundation for the model construction process in this thesis, is not sufficient.

The model represents only a small part of the legal rules of the cadastre. This is not a principal limitation of the approach. In another Ph.D thesis at the Institute for Geoinformation (Navratil 2001), which focuses directly on the formalization of law texts, a more comprehensive part of the Austrian cadastral law is represented. The integration of this work into the framework of this thesis is possible without fundamental problems.

The validity of the model is only checked with two case studies. It is a principal problem that the correctness of the model regarding reality cannot be formally proven. The solution to this issue is the testing of the specification with appropriate test cases. We selected characteristic non-trivial and expressible case studies for this purpose. Due to the lack of space and time we restricted the presentation here to two case studies.

The agent-based model is incomplete. The model works with predefined objectives, which are a specific kind of goals. The goal generation of the agent based on its own beliefs is a difficult problem in multi-agent theory and not essentially necessary for the representation of the processes discussed here. We only implemented an incomplete method for the generation of duties and objectives. It would be necessary for the development of a complete simulation tool for the evaluation of cadastral systems to realize a more general goal generation approach. Such an approach must be based on a sophisticated theory of human decision making. The completion of the agent-based model is left to future work.

10.5 Future work

According to the restrictions discussed in the last section there are several areas of future work.

The extension of the approach to spatial issues. The distinction of bona fide and fiat boundaries matches Searle's distinction of physical and institutional facts. The addition of spatial representations, of (bona fide) boundaries and beliefs about (fiat) boundaries in the agents' minds would improve the model. It allows the discussion of several phenomena, in particular the examination of conflicts regarding parcel boundaries between people when they have different beliefs about the boundaries.

The investigation of the applicability of the approach to other legal domains. There seems to be no principal obstacle to apply the agent-based framework of this thesis to other domains. The question is whether Searle's theory and the here proposed model are sufficient to describe legal domains in general. The idea is to achieve a general computational framework for the representation in the legal domain.

The completion of the agent model. The main obstacle on the way to a simulation environment for reality in a cadastre is the lack of a goal generation mechanism to make the model independent from external defined goals, i.e., objectives. The goal is to provide a mechanism that independently generates the goals that determine the behaviour of the agent based on the rights the agent has, according to his beliefs about the status of phenomena.

The representation of a more comprehensive part of the cadastral law. This task can be solved by the embedding of the work of Gerhard Navratil in his thesis (Navratil 2001) into the framework of this thesis.

The representation of different legal systems. It is interesting to model other cadastral systems in this framework, especially an instance of a deed recording system in contrast to the Austrian system as an instance of a title registration system. This gives the possibility for the comparison of the very different philosophies of these systems on a common foundation.

The development of an evaluation software for cadastral systems. Based on several previously introduced topics for future work the goal is to provide a practical tool for the evaluation of real world cadastral systems. This can help to find the weak points of the system and can improve the efficiency of the system. Therefore it can support the improvement of a real world cadastre.

Bibliography

ABGB (1811), *Allgemeines Bürgerliches Gesetzbuch*, JGS 946/1811.

Agha, G., Wegner, P. & Yonezawa, A., eds (1993), *Research Directions in Concurrent Object-Oriented Programming*, MIT Press.

Al-Taha, K. (1992), Temporal Reasoning in Cadastral Systems, Ph.d. thesis, University of Maine.

Austin, J. (1962), *How to do things with words*, Oxford University Press, Oxford.

Barr, A. & Feigenbaum, E. A. (1981), *The Handbook of Artificial Intelligence*, William Kaufmann Inc.

Bird, R. & Wadler, P. (1988), *Introduction to Functional Programming*, Prentice Hall International.

Bittner, S. (1998), Die Modellierung eines Grundbuchsystems im Situationskalkül, Diploma thesis, University of Leipzig.

Bittner, S. & Frank, A. U. (to appear), 'A formal model of correctness in a cadastre', *International Journal on Computers Environment and Urban Systems (CEUS), second special issue on Cadastral Systems*.

Bittner, S., Wolff, A. v. & Frank, A. U. (2000), The structure of reality in a cadaster, *in* B. Brogaard, ed., '23rd International Wittgenstein Symposium', Kirchberg am Wechsel, pp. 88–96.

Bond, A. & Gasser, L. (1988*a*), An analysis of problems and research in DAI, *in* A. Bond & L. Gasser, eds, 'Readings in Distributed Artificial Intelligence', Morgan Kaufmann.

Bond, A. & Gasser, L. (1988*b*), *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann.

Breu, R. (1991), *Algebraic Specification Techniques in Object Oriented Programming Environments*, Vol. 562 of *Lecture Notes in Computer Science*, Springer-Verlag.

Britannica.com (2001), 'The encyclopædia britannica online'. http://www.britannica.com.

Brooks, R. (1986), 'A robust layered control system for a mobile robot', *IEEE Journal of Robotics and Automation* **2**(1), 14–23.

Bydlinski, F. (1996), *System und Prinzipien des Privatrechtes*, Springer.

Chandrasekaran, B., Josephson, J. & Benjamins, V. R. (1999), 'What are ontologies, and why do we need them?', *IEEE Intelligent Systems* **14**(1), 20–26.

Clocksin, W. & Mellish, C. (1981), *Programming in Prolog*, Springer.

Dale, P. F. & McLaughlin, J. D. (1989), *Land Information Management, An introduction with special reference to cadastral systems in third world countries*, Oxford University Press, Oxford.

Decker, K., Pannu, A., Sycara, K. & Williamson, M. (1997), Designing behaviors for information agents, *in* L. Johnson & B. Hayes-Roth, eds, 'Proceedings of the First International Conference on Autonomous Agents (Agents'97)', ACM Press, New York, pp. 404–412.

Decker, K. S. (1996), Distributed artificial intelligence testbeds, *in* G. O'Hare & N. Jennnings, eds, 'Foundations of Distributed Artificial Intelligence', John Wiley and Sons, Inc.

Dreyfus, H. L. (1997), *What computer still can't do: a critique of artificial reason*, The MIT Press.

Durfee, E., Lesser, V. & Corkill, D. (1992), Distributed problem solving, *in* C. Shapiro, ed., 'Encyclopedia of Artificial Intelligence', 2 edn, John Wiley, pp. 379–388.

Ehrlich, H.-D., Gogolla, M. & Lipeck, U. W. (1989), *Algebraische Spezifikation abstrakter Datentypen*, B.G.Teubner, Stuttgart.

Fellbaum, C., ed. (1998), *WordNet: An Electronic Lexical Database*, Language, Speech, and Communication, The MIT Press, Cambridge, Mass.

Ferber, J. (1999), *Multi-Agent Systems. An introduction to Distributed Artificial Intelligence*, Addison-Wesley.

Finin, T., Labrou, Y. & Mayfield, J. (1997), KQML as an agent communication language, *in* J. Bradshaw, ed., 'Software Agents', MIT Press.

Frank, A. (1996), An object-oriented, formal approach to the design of cadastral systems, *in* M. Kraak & M. Molenaar, eds, 'Spatial Data Handling', Taylor & Francis, Delft, The Netherlands.

Frank, A. (2000), Communication with maps: A formalized model, *in* C. Freksa, W. Brauer, C. Habel & K. F. Wender, eds, 'Spatial Cognition II (International Workshop on Maps and Diagrammatical Representations of the Environment, Hamburg, August 1999)', Vol. 1849 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin Heidelberg, pp. 80–99.

Frank, A. U. (1997*a*), How to write specifications in class based functional programming, *in* 'Gopher as used at Geoinfo-TU Vienna', Vol. 12, Department of Geoinformation, Vienna, pp. 199–212.

Frank, A. U. (1997*b*), Spatial ontology: A geographical information point of view, *in* O. Stock, ed., 'Spatial and Temporal Reasoning', Kluwer Academic Publishers, Dordrecht, pp. 135–153.

Frank, A. U. (2001), 'Personal communication'.

Frank, A. U. (to appear), Ontology for spatio-temporal databases, *in* T. Sellis, ed., 'Spatiotemporal Databases: The Chorochronos Approach'.

Frank, A. U. & Kuhn, W. (1995), Specifying Open GIS with functional languages, *in* M. Egenhofer & J. Hering, eds, 'Advances in Spatial Databases', Vol. 951 of *Lecture Notes in Computer Science*, Springer, pp. 184–195.

Frank, A. U., Bittner, S. & Raubal, M. (to appear), Spatial and cognitive simulation with multi-agent systems, *in* 'Conference on Spatial Information Theory (COSIT'01)', Morro Bay.

Garner, B. A. (1996), *BLACK'S Law Dictionary*, West Publishing.

Gat, E. (1997), On three-layer architectures, *in* D. Kortenkamp, R. P. Bonnasso & R. Murphy, eds, 'Artificial Intelligence and Mobile Robots', AAAI Press.

GBG (1955), *Grundbuchsgesetz*, BGBI 1955/39.

Gibson, J. (1979), *The ecological approach to visual perception*, Erlbaum, Hillsdale, NJ.

Gilbert, D., Aparicio, M., Atkinson, B., Brady, S., Ciccarino, J., Grosof, B., O'Connor, P., Osisek, D., Pritko, S., Spagna, R. & Wilson, L. (1995), The role of intelligent agents in the information infrastructure, IBM report.

Gilbert, N. (1993), 'Computer simulation of social processes', *Social research update*.

Gilbert, N. (1995), 'Simulation: an emergent perspective'. http://www.soc.surrey.ac.uk/research/simsoc/tutorial.html.

Gilbert, N. & Doran, J. (1994), *The computer simulation of social phenomena*, UCL Press.

Gilbert, N. & Troitzsch, K. G. (1999), *Simulation for the social scientist*, Open University Press.

Guarino, N. (1998), Formal ontology and information systems, *in* N. Guarino, ed., 'Formal Ontology in Information Systems (Proceedings of FOIS'98, Trento, Italy, 6-8 June, 1998)', IOS Press, Amsterdam, pp. 3–15.

Guttag, J., Horowitz, E. & Musser, D. (1978), The design of data type specifications, *in* R. Yeh, ed., 'Current Trends in Programming Methodology', Vol. vol. 4: Data Structuring, Prentice Hall, pp. 60–79.

Horebeek, I. V. & Levi, J. (1989), *Algebraic Specification in Software Engineering*, Springer-Verlag.

Hudak, P., Peterson, J. & Fasel, J. H. (1997), 'A gentle introduction to Haskell'.

Jones, M. P. & Peterson, J. C. (1999), 'Hugs 98: A functional programming system based on haskell 98, users manual'.

Kanger, H. (1981), Human rights and their realization, Technical Report 1981:1, Department of Philosphy Uppsala.

Kanger, S. & Kanger, H. (1966), 'Rights and parliamentarism', *Theoria* **32**, 85–115.

Krejci, H. (1995), *Privatrecht*, 3. edn, Manzsche Verlags- und Universiätsbuchhandlung, Vienna.

Levesque, H., Reiter, R., Lesperance, Y., Lin, F. & Scherl, R. (1996), 'GOLOG: A logic programming language for dynamic domains', *Journal of Logic Programming, Special Issue on Reasoning about Action and Change.*

Liskov, B. & Guttag, J. (1986), *Abstraction and Specification in Program Development*, The MIT Electrical Engineering and Computer Science Series, The MIT Press, Cambridge, Mass.

Liskov, B. & Zilles, S. (1978), An introduction to formal specifications of data abstractions, *in* R. Yeh, ed., 'Current Trends in Programming Methodology', Vol. I, Prentice-Hall, Englewood Cliffs, N.J., pp. 1 – 33.

Loeckx, J., Ehrlich, H.-D. & Wolf, M. (1996), *Specification of Abstract Data Types*, John Wiley and Sons, B.G. Teubner, Chichester, New York, Brisbane, Toronto, Singapore, Stuttgart, Leipzig.

Maes, P. (1994), 'Agents that reduce work and information overload', *Communications of the ACM* **37**(7), 30–40.

Marent, K.-H. & Preisl, G. (1994), *Grundbuchsrecht*, Linde Verlag, Wien.

McCarthy, J. & Hayes, P. J. (1969), Some philosophical problems from the standpoint of artificial intelligence, *in* B. Melzer & D. Michie, eds, 'Machine Intelligence 4', Edinburg University Press, Edinburgh, pp. 463–502.

Meyer, B. (1988), *Object-oriented Software Construction*, Prentice Hall International.

Minsky, M. (1985), *The Society of Mind*, Simon & Schuster, New York.

Moulin, B. & Chaib-Draa, B. (1996), An overview of distributed artificial intelligence, *in* G. O'Hare & N. Jennings, eds, 'Foundations of Distributed Artificial Intelligence', John Wiley and Sons.

Navratil, G. (2001), Die Formalisierung von Gesetzen, Ph.D thesis, Technical University Vienna.

North, D. C. (1997), *Institutions, Institutional Change and Economic Performance*, The Political Economy of Institutions and Decisions, Cambridge University Press, Cambridge.

Nwana, H. & Ndumu, D. (1996), 'An introduction to agent technology', *BT Technology Journal.*

Nwana, H. S. & Ndumu, D. T. (1999), 'A perspective on software agents research', *The Knowledge Engineering Review.*

O'Hare, G. & Jennnings, N., eds (1996), *Foundations of Distributed Artificial Intelligence*, Sixth-Generation Computer Technology Series, John Wiley and Sons, Inc.

Petta, P. (2000), 'Software agents'. http://www.ai.univie.ac.at/ paolo/lva/vu-sa2000/.

Peuquet, D., Smith, B. & Brogaard, B. (1999), The ontology of fields: Report of the specialist meeting held under the auspices of the Varenius project, Technical report, NCGIA.

Puppe, F. (1991), *Einführung in Expertensysteme*, Springer-Verlag.

Raubal, M. (2001), Agent-Based Simulation of Human Wayfinding in Buildings, Ph.D. thesis, Technical University Vienna.

Reiter, R. (1991), The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression, *in* V. Lifschitz, ed., 'Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy', Academic Press, San Diego, pp. 359–380.

Russell, S. & Norvig, P. (1995), *Artificial Intelligence- A modern Approach*, Prentice Hall International, Inc.

Sannella, D. (1997), 'Essential concepts of algebraic specification and program development', *Formal Aspects of Computing* **9**, 229–269.

Schröder, J. (1993), *Verteilte künstliche Intelligenz, Methoden und Anwendungen*, BI Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zrich.

Searle, J. (1995), *The Construction of Social Reality*, The Free Press, New York.

Searle, J. R. (1969), *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge.

Shannon, C. & Weaver, W. (1948), *The mathematical theory of communication*, University of Illinois Press.

Shapiro, S., Lesperance, Y. & Levesque, H. (1997), Specifying communicative multi-agent systems with congolog, *in* 'Working Notes of the AAAI Fall 1997 Symposium on Communicative Action in Humans and Machines', AAAI Press, pp. 75–82.

Shoham, Y. (1993), 'Agent-oriented programming', *Journal of Artificial Intelligence* **60**(1), 51–92.

Smith, B. (1994), Fiat objects, *in* N. Guarino, L. Vieu & S. Pribbenow, eds, 'Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology, 11th European Conference on Artificial Intelligence', Amsterdam.

Smith, B. (1995), On drawing lines on a map, *in* A. Frank & W. Kuhn, eds, 'Spatial Information Theory-A Theoretical Basis for GIS', Vol. 988 of *Lecture Notes in Computer Science*, Springer, Berlin-Heidelberg-New York, pp. 475–484.

Smith, B. (to appear), Ontology: Philosophical and computational, *in* L. Floridi, ed., 'The Blackwell Guide to the Philosophy of Computing and Information', Blackwell, Oxford.

Smith, B. & Searle, J. (2001), 'The construction of social reality: An exchange', *American Journal of Economics and Sociology*.

Smith, B. & Zaibert, L. (to appear), 'The metaphysics of real estate', *Topoi*.

Stubkjaer, E. (2000), Information communities: A case study in the ontology on real estate, *in* B. Brogaard, ed., '23rd International Wittgenstein Symposium', Vol. 8, Austrian L. Wittgenstein Society, Kirchberg am Wechsel, pp. 159–166.

Stubkjaer, E. (2001), Spatial-socio-economic units and societal needs - danish experiences in a theoretical context, *in* A. U. Frank, J. Raper & J.-P. Cheylan, eds, 'Life and Motion of Socio-Economic Units', Taylor & Francis, London.

Thompson, S. (1996), *The Craft of Functional Programming*, Addison-Wesley.

Twaroch, C. (2000), *Organisation des Katasters: Ziele, Grundsätze und Praxis*, Vol. 14 of *GeoInfo Series*, Department of Geoinformation, Vienna.

Weiss, G. (1999), *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, The MIT Press.

Wooldridge, M. (1992), The logical modelling of computational multi-agent systems, Ph.D thesis, University of Manchester.

Wooldridge, M. (1997), 'Agent-based software engineering', *IEEE Proceedings on Software Engineering* **144**(1), 26–37.

Wooldridge, M. (1999), Intelligent agents, *in* G. Weiss, ed., 'Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence', The MIT Press.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligent agents: Theory and practice', *Knowledge Engineering Review*.

Zaibert, L. (1999), 'Real estate as institutional fact: A philosophy of everyday objects', *American Journal of Sociology and Economics* **58**, 273–284.

Appendix A

Listings of source code

A.1 Basic definitions

```
-------------------------------------------------------------
-- basics.hs
-- basic types and functions
-- Steffen Bittner / 28.09.00
-------------------------------------------------------------

module Basics where

type AgentID = String
type LandID = String
type MessageID = String
type WorldState = Int

type OpTime = Int

nodub :: Eq t ⇒ [t] → [t]
nodub [] = []
nodub (a:x) = a: nodub (filter (a/=) x)

--exchange costs, monitoring costs, measurement c, enforcement c.
type Cost = Int
data CostData = CostData Cost Cost Cost Cost deriving (Show,Eq)


fst3 :: ( a,b,c) → a
fst3 (a,b,c) = a
snd3 :: ( a,b,c) → b
snd3 (a,b,c) = b
thrd :: ( a,b,c) → c
thrd (a,b,c) = c
-------------------------------------------------------------
-- IDs.hs
-- operations for objects identifiers
--
-- Steffen Bittner / 14.02.01
-------------------------------------------------------------

module IDs where

import Worlddata
import Agentdata
import Landdata
import Statusdata
import Message
import Basics

class MyIDs myid obj where
    equalsID :: myid → obj → Bool
    nequalsID :: myid → obj → Bool
    existsObj :: myid → [obj] → Bool
    getObjbyID :: [ obj] → myid → obj
    getObjID :: obj → myid

    nequalsID i o = not ( equalsID i o )
    existsObj i ilist = not ( null ( filter (equalsID i) ilist ))

    getObjbyID ol i = ob where
         olist = filter (equalsID i) ol
        ob = if (length olist > 0) then (head olist) else error "Obj not found in getObjbyID"

instance MyIDs AgentID Agent where
    equalsID aid (Agent aid' p i a t) = aid == aid'
    getObjID (Agent aid p i a t) = aid

instance MyIDs LandID Land where
    equalsID lid (Land lid' user) = lid == lid'
    getObjID (Land lid user) = lid
```

```
instance MyIDs AgentID PhyPercept where
    equalsID aid (LandUse aid1 aid2 lid) = aid == aid1
```

A.2 The world

```
----------------------------------------------------------------
-- worlddata.hs
-- the world data
-- Steffen Bittner / 19.10.00

----------------------------------------------------------------

module Worlddata where

import Basics
import Agentdata
import Landdata
import Message

data World = World [Agent] [Land] WorldState deriving (Show)
----------------------------------------------------------------
-- world.hs
-- operations on world data
-- Steffen Bittner / 28.09.00
----------------------------------------------------------------

module World where

import Basics
import Worlddata
import Worldtech
import Agent
import Agentdata
import Agenttech
import Land
import Message
import Extcosts
import Intcosts

----------------------------------------------------------------

class Worlds w where
    incWS :: w → w
    performActs :: w → w
    sendEnvReacts :: w → w
    sendMsgs :: w → w
    doAgts :: w → w
    runEnv :: w → w
    doWorld :: [w] → Int → [w]



    doWorld whist count = if (count > 0) then
        doWorld (whist++[w']) (count−1) else whist where
        w'= runEnv (last whist)

    runEnv = incWS. doAgts. performActs. sendEnvReacts. sendMsgs


instance Worlds World where

    sendMsgs (World agents lands ws) = (World agents' lands ws) where
        agents' = fillInboxes agents msgs
        msgs = concat msgs' where
          msgs' = map getActionMsgs agents

    performActs (World agents lands ws ) = (World agents' lands' ws ) where


        agents' = emptyActions agents
        lands' = updateLandsByActions alist lands
         alist = concat alistlist
          alistlist = map getActionActs agents

    sendEnvReacts (World agents lands ws ) = (World agents' lands ws ) where
        agents' = sendReactions agents reactions
        reactions = genReactions lands alist
         alist = concat alistlist
          alistlist = map getActionActs agents



    doAgts (World agents lands ws ) = (World agents' lands ws ) where
        agents' = map genExtCosts agents2
        agents2 = map ( doAgtL ws ) agents1
        agents1 = map (initAgentCost) agents



    incWS (World agents lands ws ) = (World agents lands (ws+1))
----------------------------------------------------------------
-- worldtech.hs
-- technical operations on worlddata
-- Steffen Bittner / 19.10.00
-- history:
--
----------------------------------------------------------------
```

```
module Worldtech where

import Basics
import Worlddata
import Agent
import Agentdata
import Agenttech
import Landdata
import Message
import IDs
------------------------------------------------------------


class Worldtech w ag l | w → ag, w → l where

    startWorld :: w
    getAgents :: w → [ag]
    addAgent :: w → ag → w
    addLand :: w → l → w


instance Worldtech World Agent Land where

    startWorld = World [] [] 0

    getAgents ( World a l ws ) = a

    addAgent (World agents l ws ) (Agent aid percepts intstate actions t)
        | existsObj aid agents = error "Agent already exists"
        | otherwise = (World agents' l ws )
        where
            agents' = (Agent aid percepts intstate actions t ): agents

    addLand (World a lands ws ) (Land lid aid)
        | existsObj lid lands = error "Land already exists"
        | otherwise = (World a lands' ws )
        where
            lands' = (Land lid aid ): lands
```

A.3 The agent

```
------------------------------------------------------------
-- agentdata.hs
-- the agent data
-- Steffen Bittner / 19.10.00
------------------------------------------------------------

module Agentdata where

import Basics
import Statusdata
import Message
import Landdata
------------------------------------------------------------



data Agent = Agent AgentID [Percept] IntState [Act] CostData
             deriving (Eq,Show)


data PhyPercept = LandUse AgentID AgentID LandID | NoPhyPercept
                  deriving (Eq,Show)

type EnvReaction = PhyPercept -- physical reactions of the environment are provided as percepts to the agents


data IntState = IntState [Status] [Duty] [Objective] deriving (Eq,Show)


-- percepts can be physical percepts or communication percepts (messages)
data Percept = PPercept PhyPercept | CPercept Message deriving (Eq,Show)

-- actions can be physical or communication actions
data Act = PAct Action | CAct Message deriving (Eq,Show)




------------------------------------------------------------
------------------------------------------------------------
-- agent.hs
-- operations on agent data
-- Steffen Bittner / 28.09.00
------------------------------------------------------------

module Agent where

import Basics
import Agentdata
import Actpercepts

import Statusdata
import Statustech
import Construles
import Actduties
import Actobjectives
import Message
import Landdata
import Phypercepts



class UpdStateInMsgs aid ws p i where
    updRegMsgs :: aid → ws → [p] → i → i
    updEvtInMsgs :: aid → ws → [p] → i → i
    updSubjObjInMsgs :: aid → ws → [p] → i → i




instance UpdStateInMsgs AgentID WorldState Percept IntState where

    updRegMsgs aid ws percepts ( IntState  slist  d o) =
        if (regAgName == aid) then (IntState slist d o) else ( IntState  slist ' d o) where
                                    −− the registry agents does not evalutate info from itself
        regAgName = getRegAgName slist
                                    −− collect all messages from the registry agent
        slist ' = nodub status1
        status1 = slist ++ sl'
        sl ' = filter (NoStatus /=) sl1
        sl1 = map (createStatusMsg ws) inbox'

        inbox' = fst  inboxlistlist
         inboxlistlist  = span (isRegAgInfo regAgName) inbox
        inbox = getMessagesfromPercepts percepts
        isRegAgInfo rname (Message startid z c mtype rmessage) =
            (rname==startid && mtype==Answer_Query)

    updEvtInMsgs aid ws percepts ( IntState  slist  d o) = (IntState  slist ' d o) where

         slist ' =  filterdubBeliefs   slist3
         slist3  =  slist1  ++ slist
         slist1  = filter  (NoStatus /=) slist2
         slist2  = map (updStateEvtMsg ws slist) inbox'

        inbox' = snd  inboxlistlist
         inboxlistlist  = span (isRegAgInfo regAgName) inbox
        inbox = getMessagesfromPercepts percepts
        isRegAgInfo rname (Message startid z c mtype rmessage) =
            (rname==startid && mtype==Answer_Query)
        regAgName = getRegAgName slist

    updSubjObjInMsgs aid ws percepts ( IntState  slist  duty  objectives ) = (IntState  slist ' duty'  objectives ') where

         slist ' =  filterdubBeliefs   slist4
         slist4  =  slist1  ++ slist
        duty' = duty ++ duty2
         slist1  = filter  (NoStatus /=) slist2
         slist2  = concat (map fst3 slist3)
         objectives ' = objectives ++ objective2
         objective2  = filter  (NoObjective /=) objective1
         objective1  = concat (map thrd slist3)

        duty2 = filter  (NoDuty /=) duty1
        duty1 = concat (map snd3 slist3)
         slist3  = map (updSubjObjMsg ws slist True) inbox'
        inbox' = snd  inboxlistlist
         inboxlistlist  = span (isRegAgInfo regAgName) inbox
        inbox = getMessagesfromPercepts percepts
        isRegAgInfo rname (Message startid z c mtype rmessage) =
            (rname==startid && mtype==Answer_Query)
        regAgName = getRegAgName slist



class UpdStateOutMsgs ws a i where
    updEvtOutMsgs :: ws → [a] → i → i
    updSubjObjOutMsgs :: ws → [a] → i → i

instance UpdStateOutMsgs WorldState Act IntState where

    updEvtOutMsgs ws actions (IntState  slist  d o) = (IntState  slist ' d o) where

         slist ' =  filterdubBeliefs   slist3
         slist3  =  slist1  ++ slist
         slist1  = filter  (NoStatus /=) slist2
         slist2  = map (updStateEvtMsg ws slist) outbox

        outbox = getMessagesfromActions actions

    updSubjObjOutMsgs ws actions (IntState  slist  d o) = (IntState  slist ' d o) where

         slist ' =  filterdubBeliefs   slist4
         slist4  =  slist1  ++ slist
         slist1  = filter  (NoStatus /=) slist2
         slist2  = concat (map fst3 slist3)

         slist3  = map (updSubjObjMsg ws slist False) outbox
        outbox = getMessagesfromActions actions



class ( UpdStateInMsgs aid ws p i, UpdStateOutMsgs ws a i) ⇒ MyAgentInternals aid ws p i a | i→ aid, i→ws, i → p, i → a where
```

```
    filterState  :: i → ws → i
    actDuties :: aid → ws → i → [a]
    actObjectives :: aid → ws → i → [a]

    updStatePhy :: aid → ws → [p] → i → i
    updStateInMessages :: aid → ws → [p] → i → i
    updStateOutMessages :: ws → [a] → i → i

    updStateInMessages aid ws p = updSubjObjInMsgs aid ws p. updEvtInMsgs aid ws p. updRegMsgs aid ws p

    updStateOutMessages ws a = updSubjObjOutMsgs ws a . updEvtOutMsgs ws a


instance MyAgentInternals AgentID WorldState Percept IntState Act where

    filterState (IntState slist duties objectives) ws = (IntState slist duties' objectives') where
       duties' = filter (isCurrentDuty ws) d1
       d1 = filter (NoDuty /=) duties
       objectives' = filter (isCurrentObjective ws) objectives1
       objectives1 = filter (NoObjective /=) objectives


    actDuties aid ws (IntState slist duties objectives) = erglist where
       erglist = concat (map (doCurrentDuty slist aid ws) duties)

    actObjectives aid ws (IntState slist duties objectives) = erglist where
       erglist = concat (map (doObjective slist aid ws) objectives)

    updStatePhy aid ws percepts (IntState slist duties objectives) = (IntState slist duties objectives') where
         objectives' = objectives ++ updStatePhyPercepts slist aid ws percepts'
         percepts' = map getPhyPercept (filter isPhyPercept percepts)



class MyAgentInternals aid ws p i a ⇒ AgentInternals aid ws p i a where


    updStatePercepts :: aid → ws → [p] → i → i

    selActs :: aid → ws → i → [a]

    updStateActions :: ws → [a] → i → i

    decision :: aid → ws → [p] → i → ([a], i)

    decision aid ws p i =
      ( selActs aid ws ( updStatePercepts aid ws p i),
        updStateActions ws (selActs aid ws ( updStatePercepts aid ws p i)) (updStatePercepts aid ws p i ) )



    updStatePercepts aid ws p = updStateInMessages aid ws p. updStatePhy aid ws p

    selActs aid ws i = (actDuties aid ws i) ++ (actObjectives aid ws i )
    updStateActions = updStateOutMessages

instance AgentInternals AgentID WorldState Percept IntState Act


class AbsAgent a aid ws | a → ws, a → aid where
    newAgt :: aid → a
    doAgt :: ws → a → a


instance AbsAgent Agent AgentID WorldState where
    newAgt aid = Agent aid [] ( IntState [] [] []) [] ( CostData 0 0 0 0)

    doAgt ws (Agent aid percepts intstate actions cost) = (Agent aid percepts intstate' actions' cost) where
       (actions', intstate') = decision aid ws percepts intstate2
       intstate2 = filterState intstate ws
-----------------------------------------------------------------
-- agenttech.hs
-- technical operations on the agent data
-- Steffen Bittner / 19.10.00
-----------------------------------------------------------------

module Agenttech where

import Basics
import Agentdata
import Actpercepts
import Statusdata
import Message
import Landdata
import IDs
-----------------------------------------------------------------



class Agenttech a where
    setInbox :: a → [Message] → a

    setActions :: a → [Act] → a
    getActionMsgs :: a → [Message]
    getActionActs :: a → [Action]
    fillInboxes   :: [ a] → [Message] → [a]
    emptyActions :: [ a] → [a]

    addMessageActions :: a → Message → a
    addBelief  ::  a → Status → a
    addObjective  ::  a →  Objective  → a
    initAgentCost  ::  a → a
    sendReactions  :: [ a] → [EnvReaction] → [a]
    sendReaction  :: [ EnvReaction] → a → a

instance Agenttech Agent where

    setInbox  ( Agent aid  percepts   intstate   actions  t )  msgs =
        ( Agent aid  percepts '   intstate   actions  t )  where
              percepts ' = genCPercepts mymsgs
              mymsgs = getMessagesByReceiverID aid msgs

    getActionMsgs ( Agent aid  percepts   intstate   actions  t )  = actmsgs where
        actmsgs = getMessagesfromActions actions

    getActionActs  ( Agent aid  percepts   intstate   actions  t )  = actacts  where
        actacts  = getActionsfromActs actions


    setActions  ( Agent aid  percepts   intstate   actions  t )  actions ' =
        ( Agent aid  percepts   intstate   actions '  t )


    fillInboxes   agents  msgs = map (flip setInbox msgs)  agents

    emptyActions agents = map (flip setActions  [])  agents


    addMessageActions ( Agent aid percepts   intstate   actions  t )  m =
        (  Agent aid  percepts   intstate   ( actions++[(CAct m)])  t )


    addBelief  ( Agent aid  percepts  ( IntState  st  duty  objectives )  actions  t )  bel  =
        ( Agent aid  percepts  ( IntState  ( bel : st )  duty  objectives )  actions  t )

    addObjective  ( Agent aid  percepts  ( IntState  st  duty  objectives )  actions  t )  objective  =
        ( Agent aid  percepts  ( IntState  st  duty  ( objective : objectives ))  actions  t )


    initAgentCost  ( Agent aid  percepts   intstate   actions  t )  =
        ( Agent aid  percepts   intstate   actions  ( CostData 0 0 0 0))

    sendReactions agents  reactions  = map (sendReaction reactions)  agents

    sendReaction  reactions  ( Agent aid  percepts   intstate   actions  t )  =
        ( Agent aid  percepts '   intstate   actions  t )  where
            percepts ' = percepts ++ genPPercepts (filter (equalsID  aid)  reactions )
```

A.4 The internal state of the agent

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
−− statusdata.hs
−− datatypes for status , goals , duties
−−
−− Steffen Bittner / 18.10.00
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

module Statusdata where

import Basics
import Message

data Duty = AskRegAgent Message WorldState
                      |  AnswerQuery Message WorldState
                      |  AnswerApplication Message WorldState
                      |  DoApplication Message WorldState
                      |  DoJudgement Message WorldState
                      |  DoServeComplaint Message WorldState
                      |  DoAnswerComplaint Message WorldState
                      |  DoAbandonLand Message WorldState
                      |  DoExecution Message WorldState
                      |  NoDuty deriving (Show,Eq)

data Objective =  SellParcel  LandID AgentID WorldState
                      |  AnswerOffer Message WorldState
                      |  UseLand LandID WorldState
                      |  DoComplaint LandID AgentID WorldState
                      |  ApplyExecution LandID AgentID WorldState
                      |  NoObjective deriving (Eq,Show)

data Status = Owner AgentID LandID Message WorldState
                        |  Legal_person  AgentID WorldState
                        |  Parcel  LandID WorldState
                        |  RegAg AgentID WorldState
                        |  CourtAg AgentID WorldState
                        |  SheriffAg  AgentID WorldState
                        |  Offeror  AgentID Message WorldState
                        |  Transferee  AgentID Message WorldState
                        |  Buyer AgentID Message WorldState
                        |  Seller  AgentID Message WorldState
                        |  Applicant  AgentID Message WorldState
                        |   Application_receiver   AgentID Message WorldState
                        |  Query_sender AgentID Message WorldState
```

```
                    | Query_receiver AgentID Message WorldState
                    | Plaintiff AgentID Message WorldState
                    | Defendant AgentID Message WorldState
                    | Judge AgentID Message WorldState
                    | Loser_Complaint AgentID Message WorldState
                    | JudgementExecutor AgentID Message WorldState
                    | Applicant_Execution AgentID Message WorldState
                    | Querying Message WorldState
                    | Offering Message WorldState
                    | Contracting Message WorldState
                    | Applying Message WorldState
                    | Transferring Message WorldState
                    | Suing Message WorldState
                    | ServingComplaint Message WorldState
                    | PronouncingJudgement Message WorldState
                    | ExecutingJudgement Message WorldState
                    | Execution_Title AgentID AgentID LandID Message WorldState
                    | NoStatus deriving (Show, Eq)
-----------------------------------------------------------------
-- construles.hs
-- implementation of constitutive rules
--
-- Steffen Bittner / 01.03.01
-----------------------------------------------------------------

module Construles where

import Basics
import Statusdata
import Statustech
import Message
import Landdata
import IDs


class Construles ws m s d o | ws → m, ws → s, ws → d, ws → o where
   createStatusMsg :: ws → m → s
   updStateEvtMsg :: ws → [s] → m → s
   updSubjObjMsg :: ws → [s] → Bool → m → ([s],[d],[o])



instance Construles WorldState Message Status Duty Objective where

   createStatusMsg ws (Message s z (Query_Owner aID lID) Answer_Query rmsg) =
      (Owner aID lID NoMessage ws)
   createStatusMsg ws _ = NoStatus

   -- constitutive rules for events
   updStateEvtMsg ws sl (Message s z (Sell seller buyer land) Offer rmsg) =
      if ((isLegalPerson sl seller) && (isLegalPerson sl buyer) && (isParcel sl land))
         then (Offering (Message s z (Sell seller buyer land) Offer rmsg) ws) else NoStatus

   updStateEvtMsg ws sl (Message s z (Query_Owner aid land) Query rmsg) =
      if ((isRegAgent sl z) && (isLegalPerson sl s)) then
          (Querying (Message s z (Query_Owner aid land) Query rmsg) ws) else NoStatus

   updStateEvtMsg ws sl (Message s z (Query_Owner aid land) Answer_Query rmsg) =
      NoStatus

   updStateEvtMsg ws sl (Message s z (Sell seller buyer land) Accept_Offer rmsg) =
      if (( isOfferor sl z) && (isTransferee sl s)) then
          (Contracting (Message s z (Sell seller buyer land) Accept_Offer rmsg) ws)
         else NoStatus

   updStateEvtMsg ws sl (Message s z (Sell seller buyer land) Reject_Offer rmsg) =
      NoStatus

   updStateEvtMsg ws sl (Message s z (Sell seller buyer land) Application rmsg) =
      if ((isRegAgent sl z) && (isLegalPerson sl s)) then
            (Applying (Message s z (Sell seller buyer land) Application rmsg) ws)
         else NoStatus

   updStateEvtMsg ws sl
      (Message s z (Sell seller buyer land) Accept_Application rmsg) =
         if ((isRegAgent sl s) && (isLegalPerson sl z)) then
            (Transferring (Message s z (Sell seller buyer land) Accept_Application rmsg) ws)
            else NoStatus

   updStateEvtMsg ws sl
      (Message s z (Sell seller buyer land) Reject_Application rmsg) = NoStatus

   updStateEvtMsg ws sl
      (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) =
         if ((isCourtAgent sl z) && (isLegalPerson sl s) && (s==plaintiff)) then
            (Suing (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) ws)
            else NoStatus

   updStateEvtMsg ws sl
      (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) =
         if ((isCourtAgent sl s) && (isLegalPerson sl z) && (z==defendant)) then
            (ServingComplaint (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) ws)
            else  NoStatus

   updStateEvtMsg ws sl
      (Message s z (Complaint plaintiff defendant land) Answer_Complaint proofmsg) =
         NoStatus
```

```
updStateEvtMsg ws sl
    (Message s z (Complaint plaintiff defendant land) Judgement proofmsg) =
        if ((isCourtAgent sl s) && (isLegalPerson sl z) &&
            (( isPlaintiff sl z ) || ( isDefendant sl z))) then
               (PronouncingJudgement (Message s z (Complaint plaintiff defendant land) Judgement proofmsg) ws)
            else NoStatus

updStateEvtMsg ws sl
    (Message s z (Complaint plaintiff defendant land) Reject_Complaint proofmsg) =
        NoStatus

updStateEvtMsg ws sl
    (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) =
        if (( isSheriffAgent sl z) && (isLegalPerson sl s) && (s==plaintiff)) then
            (ExecutingJudgement (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) ws)
            else NoStatus

updStateEvtMsg ws sl
    (Message s z (Complaint plaintiff defendant land) Accept_Execution proofmsg) =
        NoStatus

updStateEvtMsg ws sl
    (Message s z (Complaint plaintiff defendant land) Reject_Execution proofmsg) =
        NoStatus

updStateEvtMsg _ _ _ = error "No constitutive rule found in updStateEvtMsg"

-- constitutive rules for subjects / objects
updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Offer rmsg) =
    if ( isOffering sl (Message s z ( Sell seller buyer land) Offer rmsg)) then sdlist
       else ([NoStatus],[NoDuty],[NoObjective])
    where
        sdlist = ([( Offeror seller (Message s z ( Sell seller buyer land) Offer rmsg) ws)] ++
            [( Transferee buyer (Message s z ( Sell seller buyer land) Offer rmsg) ws)], duty, objective )
       duty = if ( inflag==True) then [(AskRegAgent (Message s z (Sell seller buyer land) Offer rmsg) ws)] else
            [NoDuty]
        objective = if ( inflag==True) then [(AnswerOffer (Message s z (Sell seller buyer land) Offer rmsg) (ws+2))] else
            [NoObjective]

updSubjObjMsg ws sl inflag
    (Message s z ( Query_Owner aid land) Query rmsg) =
    if (isQuerying sl (Message s z ( Query_Owner aid land) Query rmsg)) then sdlist
       else ([NoStatus],[NoDuty],[NoObjective])
    where
        sdlist = ( [( Query_sender s (Message s z ( Query_Owner aid land) Query rmsg) ws)] ++
                   [( Query_receiver z (Message s z ( Query_Owner aid land) Query rmsg) ws)],
                     duty , [ NoObjective])
       duty = if ( inflag==True)
           then [AnswerQuery (Message s z (Query_Owner aid land) Query rmsg) ws] else [NoDuty]

updSubjObjMsg ws sl inflag
    (Message s z ( Query_Owner aid land) Answer_Query rmsg) =
        ([NoStatus],[NoDuty],[NoObjective])

updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Accept_Offer rmsg) =
        if ( isContracting sl (Message s z ( Sell seller buyer land) Accept_Offer rmsg)) then sdlist
           else ([NoStatus],[NoDuty],[NoObjective])
        where sdlist = ([( Seller seller (Message s z ( Sell seller buyer land) Accept_Offer rmsg) ws),
                         (Buyer buyer (Message s z ( Sell seller buyer land) Accept_Offer rmsg) ws)],duty , [NoObjective])
               duty = if ( inflag==True) then
                   [DoApplication (Message s z ( Sell seller buyer land) Accept_Offer rmsg) ws] else [NoDuty]

updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Reject_Offer rmsg) =
        ([NoStatus],[NoDuty], [NoObjective])

updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Application rmsg) =
        if ( isApplying sl (Message s z ( Sell seller buyer land) Application rmsg)) then sdlist
           else([NoStatus],[NoDuty], [NoObjective])
        where sdlist = ([( Applicant s (Message s z ( Sell seller buyer land) Accept_Offer rmsg) ws),
                          ( Application_receiver z (Message s z ( Sell seller buyer land) Accept_Offer rmsg) ws)],duty , [NoObjective])
               duty = if ( inflag==True) then
                    [AnswerApplication (Message s z ( Sell seller buyer land) Application rmsg) ws] else [NoDuty]

updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Accept_Application rmsg) =
        if ( isTransferring sl (Message s z ( Sell seller buyer land) Accept_Application rmsg)) then sdlist
           else ([NoStatus],[NoDuty], [NoObjective])
        where sdlist = ([(Owner buyer land (Message s z ( Sell seller buyer land) Accept_Application rmsg) ws)],
                          [NoDuty], [NoObjective])

updSubjObjMsg ws sl inflag
    (Message s z ( Sell seller buyer land) Reject_Application rmsg) =
        ([NoStatus],[NoDuty], [NoObjective])

updSubjObjMsg ws sl inflag
    (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) =
        if (isSuing sl (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg)) then sdlist
           else ([NoStatus],[NoDuty], [NoObjective])
        where sdlist = ([(Defendant defendant (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) ws),
                          ( Plaintiff plaintiff (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) ws),
                    (Judge z (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) ws)], duty , [NoObjective])
               duty = if ( inflag==True) then
                   [( DoServeComplaint (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) ws),
                    (DoJudgement (Message s z (Complaint plaintiff defendant land) Legal_Action proofmsg) (ws+2))] else [NoDuty]

updSubjObjMsg ws sl inflag
```

```
      (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) =
          if (isServingComplaint sl (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg)) then sdlist
             else ([NoStatus],[NoDuty],[NoObjective])
          where sdlist = if (inflag==True) then
                        ([(Defendant defendant (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) ws),
                          (Plaintiff  plaintiff  (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) ws),
                        (Judge s (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) ws)],
                        [(DoAnswerComplaint (Message s z (Complaint plaintiff defendant land) Serve_Complaint proofmsg) ws)],[NoObjective])
                        else
                             ([NoStatus],[NoDuty],[NoObjective])

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Answer_Complaint proofmsg) =
      ([NoStatus],[NoDuty],[NoObjective])

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Judgement proofmsg) =
          if (isPronouncingJudgement sl (Message s z (Complaint plaintiff defendant land) Judgement proofmsg)) then sdlist
             else  ([NoStatus],[NoDuty],[NoObjective])
          where sdlist = ([(Loser_Complaint defendant (Message s z (Complaint plaintiff defendant land) Judgement proofmsg) ws),
                           (Execution_Title  plaintiff  defendant land (Message s z (Complaint plaintiff defendant land) Judgement proofmsg)
                            ws)],  duty,[NoObjective])
                 duty = if ((isDefendant sl z) && (inflag==True)) then
                        [(DoAbandonLand (Message s z (Complaint plaintiff defendant land) Judgement proofmsg) ws)] else [NoDuty]

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Reject_Complaint proofmsg) =
          ([NoStatus],[NoDuty],[NoObjective])

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) =
          if (isExecutingJudgement sl (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg)) then sdlist
             else ([NoStatus],[NoDuty],[NoObjective])
          where sdlist = ([(Applicant_Execution s (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) ws),
                             (JudgementExecutor z (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) ws)],
                          duty,[NoObjective])
                 duty = if (inflag==True) then [DoExecution (Message s z (Complaint plaintiff defendant land) Apply_Execution proofmsg) ws]
                           else [NoDuty]

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Accept_Execution proofmsg) =
          ([NoStatus],[NoDuty],[NoObjective])

   updSubjObjMsg ws sl inflag
      (Message s z (Complaint plaintiff defendant land) Reject_Execution proofmsg) =
            ([NoStatus],[NoDuty],[NoObjective])

   updSubjObjMsg _ _ _ _ =
      error "No constitutive rule found in updSubjObjMsg"
------------------------------------------------------------
-- actduties.hs
-- activities of the agent according to its duties
--
-- Steffen Bittner / 01.03.01
------------------------------------------------------------

module Actduties where

import Basics
import Statusdata
import Statustech
import Message
import Landdata
import IDs
import Agentdata

class Actduties s d aid ws m a | a → d, a → m, a → s, a → aid where


   answerQuery :: [s] → aid → ws → m → [a]
   askRegAgent :: [s] → aid → ws → m → [a]
   doApplication   :: [s] → aid → ws → m → [a]
   answerApplication  :: [s] → aid → ws → m → [a]
   serveComplaint :: [s] → aid → ws → m → [a]
   doJudgement :: [s] → aid → ws → m → [a]
   answerComplaint :: [s] → aid → ws → m → [a]
   abandonLand :: ws → m → [a]
   doExecution :: [s] → aid → ws → m → [a]

   doCurrentDuty :: [s] → aid → ws → d → [a]
   doDuty :: [s] → aid → ws → d → [a]


instance Actduties Status Duty AgentID WorldState Message Act where

   answerQuery slist aid ws msg = [(CAct answer)] where
        answer = (Message aid goal (Query_Owner ownerid land ) Answer_Query NoMessage)
        goal = getSender msg
        ownerid = getLastOwner slist land
        land = getLandIDfromContent content
        content = getContent msg

   askRegAgent slist aid ws msg = [(CAct answer)] where
      answer = (Message aid regag (Query_Owner "" land ) Query NoMessage)
      land = getLandIDfromContent c
      regag = getRegAgName slist
      (Message seller g c t rmsg) = msg
```

```
    doApplication  slist  aid  ws msg = [(CAct answer)] where
        answer = (Message aid regag content  Application  NoMessage)
        content  = getContent msg
        regag  = getRegAgName slist

    answerApplication  slist  aid  ws msg = [(CAct answer1),(CAct answer2)] where
        answer1 = (Message aid seller  content  msgtype NoMessage)
        answer2 = (Message aid buyer content  msgtype NoMessage)
        msgtype = if ( ownerid == seller) then Accept_Application else  Reject_Application
        ownerid = getLastOwner slist  land
        land = getLandIDfromContent content
        buyer = getBuyerIDfromContent content
         seller  = getSellerIDfromContent content
        content  = getContent msg

    serveComplaint  slist  aid  ws msg = [(CAct erg)] where
        erg  = (Message aid defendant content  Serve_Complaint NoMessage)
        defendant = getDefendantIDfromContent content
        (Message  plaintiff   courtag  content  msgtype rmsg) = msg

    doJudgement slist  aid  ws msg = [(CAct erg1),(CAct erg2)] where
        erg1 = if  ( isOwnershipTransfer proofmessage  plaintiff  land)  then
            (Message aid defendant content  Judgement NoMessage) else
                (Message aid defendant content  Reject_Complaint NoMessage)
        erg2  = if  ( isOwnershipTransfer proofmessage  plaintiff  land)  then
            (Message aid  plaintiff  content  Judgement NoMessage) else
                (Message aid  plaintiff  content  Reject_Complaint NoMessage)
        defendant = getDefendantIDfromContent content
        land  = getLandIDfromContent content

        (Message  plaintiff   courtag  content  msgtype proofmessage) = msg

    answerComplaint slist  aid  ws msg = [(CAct erg)] where
        erg  = (Message aid courtag content  Answer_Complaint NoMessage)
        courtag  = getCourtAgName slist
        content  = getContent msg

    abandonLand ws msg = [(PAct erg)] where
        erg  = (ActAbandonLand land)
        land  = getLandIDfromContent content
        content  = getContent msg

    doExecution  slist  aid  ws msg = [(CAct msg'),(PAct act)] where
        act  = if  ( isExecutionTitle  proofmsg plain  def  land)  then  (ActEvictLand land) else NoAction
        msg' = if  ( isExecutionTitle  proofmsg plain  def  land)  then msg1 else msg2
        msg1 = (Message aid applicant content  Accept_Execution  NoMessage)
        msg2 = (Message aid applicant content  Reject_Execution  NoMessage)
        def  = getDefendantIDfromContent content
        plain  = getPlaintiffIDfromContent content
        land  = getLandIDfromContent content
        (Message applicant  sheriff  content  msgtype proofmsg) = msg

    doCurrentDuty sl aid  ws duty =
        if  (  ws == (getDutyWS duty)) then doDuty sl aid ws duty else []

    doDuty sl  aid  ws (AnswerQuery mid ws') = (answerQuery sl aid ws mid)
--   doDuty sl  hist  aid  ws (AnswerOffer mid ws' ) = (answerOffer sl  hist  aid  ws mid)
    doDuty sl  aid  ws (AskRegAgent mid ws' ) = (askRegAgent sl aid ws mid)
    doDuty sl  aid  ws (DoApplication mid ws' ) = (doApplication  sl  aid  ws mid)
    doDuty sl  aid  ws (AnswerApplication mid ws' ) = (answerApplication  sl  aid  ws mid)
    doDuty sl  aid  ws (DoServeComplaint mid ws' ) = (serveComplaint sl aid ws mid)
    doDuty sl  aid  ws (DoJudgement mid ws' ) = (doJudgement sl aid ws mid)
    doDuty sl  aid  ws (DoAnswerComplaint mid ws' ) = (answerComplaint sl aid ws mid)
    doDuty sl  aid  ws (DoAbandonLand mid ws' ) = (abandonLand ws mid)
    doDuty sl  aid  ws (DoExecution mid ws' ) = (doExecution sl aid  ws mid)
    doDuty _ _ _  NoDuty = error "NoDuty in doDuty"
    doDuty _ _ _ _   = error "No case found in doDuty"
-----------------------------------------------------------------
-- actobjectives.hs
-- activities  of  the  agent  according  to  its   objectives
--
-- Steffen Bittner / 01.03.01

module Actobjectives where

import Basics
import Statusdata
import Statustech
import Message
import Landdata
import Land
import IDs
import Agentdata


class  Actobjectives  s  o  aid  lid  ws m a | s → m, s → lid ,  s → o,  s → ws where

    doOffer  :: [ s]  → aid → aid →  lid  → ws → [a]
    answerOffer  :: [ s]  → aid → ws → m → [a]

    doComplaint :: [ s ]  → aid  → aid  → ws →  lid  → [a]
    doApplyExecution  :: [ s ]  → aid  → aid  → ws →  lid  → [a]
    doActUseLand :: [s ]  →  lid  → aid  → [ a]
    doObjective   :: [ s]  → aid → ws → o → [a]


instance Actobjectives  Status  Objective  AgentID LandID WorldState Message Act where

    doOffer  sl  aid  buyer land  ws   = [(CAct msg)] where
        msg = (Message aid buyer (Sell aid buyer land) Offer NoMessage)

    answerOffer slist aid ws msg = [(CAct answer)] where
        answer = if (ownerid == seller) then (Message aid seller c Accept_Offer NoMessage) else
              (Message aid seller c Reject_Offer NoMessage)
        ownerid = getOwnerNow slist land ws
        land = getLandIDfromContent c
        regag = getRegAgName slist
        (Message seller g c t rmsg) = msg


    doComplaint sl aid aid' ws land = [(CAct msg)] where
        msg = (Message aid court (Complaint aid aid' land) Legal_Action proof')
        court = getCourtAgName sl
        proof' = if (ai == aid) then proof else NoMessage --reference to the proofmessage of ownership
        (Owner ai li proof w) = getLastOwnerStatus sl land

    doApplyExecution sl aid aid' ws land = [(CAct msg)] where
        msg = (Message aid sheriff (Complaint aid aid' land) Apply_Execution proof)
        sheriff = getSheriffAgName sl
        proof = getJudgementID sl aid aid' land --reference to the proofmessage of the
                                                --execution title

    doActUseLand slist land aid = [(PAct (ActUseLand land aid))]

    doObjective sl aid ws ( SellParcel land aid' ws') = erg where
        erg = if (ws==ws') then (doOffer sl aid aid' land ws) else []

    doObjective sl aid ws (AnswerOffer mid ws' ) = erg where
        erg = if (ws==ws') then (answerOffer sl aid ws mid) else []

    doObjective sl aid ws (DoComplaint land aid' ws') = erg where
        erg = if (ws==ws') then (doComplaint sl aid aid' ws land) else []

    doObjective sl aid ws (ApplyExecution land aid' ws') = erg where
        erg = if (ws==ws') then (doApplyExecution sl aid aid' ws land) else []


    doObjective sl aid ws (UseLand land ws') = erg where
         erg = if (ws /= ws') then [] else doActUseLand sl land aid


    doObjective _ _ _ NoObjective = error "NoObjective in doObjective"
    doObjective _ _ _ _ = error "No case found in doObjective"
----------------------------------------------------------------
-- statustech.hs
-- contains "technical" operations on status , objectives , duties ,
--
-- Steffen Bittner / 18.10.00
----------------------------------------------------------------

module Statustech where

import Basics
import Statusdata
import IDs
import Message


class Statustech s where
    equStatus :: s → s → Bool
     filterdubBeliefs :: [ s] → [s]
    getRegAgName :: [s] → AgentID
    getCourtAgName :: [s] → AgentID
    getSheriffAgName :: [ s] → AgentID
    equAIDRegAg :: AgentID → s → Bool
    equAIDLegalPerson :: AgentID → s → Bool
    equLIDParcel :: LandID → s → Bool
    equMIDOffering :: Message → s → Bool
    equMIDQuerying :: Message → s → Bool
    equAIDOfferor :: AgentID → s → Bool
    equAIDTransferee :: AgentID → s → Bool
    equAIDPlaintiff :: AgentID → s → Bool
    equAIDDefendant :: AgentID → s → Bool
    equMIDContracting :: Message → s → Bool
    equMIDApplying :: Message → s → Bool
    equMIDTransferring :: Message → s → Bool
    equAIDCourtAg :: AgentID → s → Bool
    equMIDSuing :: Message → s → Bool
    equMIDServingComplaint :: Message → s → Bool
    equMIDPronouncingJudgement :: Message → s → Bool
    equAIDSheriffAg :: AgentID → s → Bool
    equMIDExecutingJudgement :: Message → s → Bool
    isRegAgent :: [ s] → AgentID → Bool
    isLegalPerson :: [ s] → AgentID → Bool
    isParcel :: [ s] → LandID → Bool
    isOffering :: [ s] → Message → Bool
    isQuerying :: [ s] → Message → Bool
    isOfferor :: [ s] → AgentID → Bool
    isTransferee :: [ s] → AgentID → Bool
     isPlaintiff :: [ s] → AgentID → Bool
    isDefendant :: [ s] → AgentID → Bool
    isContracting :: [ s] → Message → Bool
    isApplying :: [ s] → Message → Bool
    isTransferring :: [ s] → Message → Bool
    isCourtAgent :: [ s] → AgentID → Bool
    isSuing :: [ s] → Message → Bool
    isServingComplaint :: [ s] → Message → Bool
    isPronouncingJudgement :: [s] → Message → Bool
```

```
    isSheriffAgent  :: [ s ] → AgentID → Bool
    isExecutingJudgement :: [ s ] → Message → Bool
    getLastOwner :: [ s ] → LandID → AgentID
    getLastOwnerStatus :: [ s ] → LandID → s
    getOwnerNow :: [s] → LandID → WorldState → AgentID
    getJudgementID :: [ s ] → AgentID → AgentID → LandID → Message
    hasExecutionTitle  :: [ s ] → AgentID → AgentID → LandID → Bool


instance Statustech Status where

    equStatus ( Owner aid lid mid ws) (Owner aid' lid ' mid' ws') =
        (( aid == aid') && (lid == lid') && (ws == ws'))
    equStatus ( Loser_Complaint aid mid ws) (Loser_Complaint aid' mid' ws') =
        (( aid == aid') && (ws == ws'))
    equStatus ( Execution_Title aid aid2 lid mid ws)
        ( Execution_Title aid' aid2' lid ' mid' ws') =
            (( aid == aid') && (aid2 == aid2') && (lid == lid') && (ws == ws'))
    equStatus _ _ = False

     filterdubBeliefs  [] = []
     filterdubBeliefs  (a:x) = a: ( filterdubBeliefs  ( filter ( nequStatus a) x ))
        where nequStatus x y = not (equStatus x y)

    getRegAgName ((RegAg aid ws):rest) = aid
    getRegAgName [] = error "registry agent unknown to agent"
    getRegAgName (_:rest) = getRegAgName rest

    getCourtAgName ((CourtAg aid ws):rest) = aid
    getCourtAgName [] = error "court agent unknown to agent"
    getCourtAgName (_:rest) = getCourtAgName rest

    getSheriffAgName ((SheriffAg aid ws): rest ) = aid
    getSheriffAgName [] = error "sheriff agent unknown to agent"
    getSheriffAgName (_: rest ) = getSheriffAgName rest

    equAIDRegAg aid' (RegAg aid ws) = (aid == aid')
    equAIDRegAg _ _ = False

    equAIDLegalPerson aid' (Legal_person aid ws) = (aid == aid')
    equAIDLegalPerson _ _ = False

    equLIDParcel lid ' ( Parcel lid ws) = (lid == lid')
    equLIDParcel _ _ = False

    equMIDOffering mid' (Offering mid ws) = (mid == mid')
    equMIDOffering _ _ = False

    equMIDQuerying mid' (Querying mid ws) = (mid == mid')
    equMIDQuerying _ _ = False

    equAIDOfferor aid' ( Offeror aid mid ws) = (aid == aid')
    equAIDOfferor _ _ = False

    equAIDTransferee aid' ( Transferee aid mid ws) = (aid == aid')
    equAIDTransferee _ _ = False

    equAIDPlaintiff aid' ( Plaintiff aid mid ws) = (aid == aid')
    equAIDPlaintiff _ _ = False

    equAIDDefendant aid' (Defendant aid mid ws) = (aid == aid')
    equAIDDefendant _ _ = False

    equMIDContracting mid' (Contracting mid ws) = (mid == mid')
    equMIDContracting _ _ = False


    equMIDApplying mid' (Applying mid ws) = (mid == mid')
    equMIDApplying _ _ = False

    equMIDTransferring mid' (Transferring mid ws) = (mid == mid')
    equMIDTransferring _ _ = False

    equAIDCourtAg aid' (CourtAg aid ws) = (aid == aid')
    equAIDCourtAg _ _ = False

    equMIDSuing mid' (Suing mid ws) = (mid == mid')
    equMIDSuing _ _ = False

    equMIDServingComplaint mid' (ServingComplaint mid ws) = (mid == mid')
    equMIDServingComplaint _ _ = False

    equMIDPronouncingJudgement mid' (PronouncingJudgement mid ws) = (mid == mid')
    equMIDPronouncingJudgement _ _ = False

    equAIDSheriffAg aid' ( SheriffAg aid ws) = (aid == aid')
    equAIDSheriffAg _ _ = False

    equMIDExecutingJudgement mid' (ExecutingJudgement mid ws) = (mid == mid')
    equMIDExecutingJudgement _ _ = False

    isRegAgent sl aid ' = not (null (filter ( equAIDRegAg aid') sl))
    isLegalPerson sl aid ' = not (null (filter ( equAIDLegalPerson aid') sl))
     isParcel sl lid ' = not (null (filter ( equLIDParcel lid ') sl))
     isOffering sl mid' = not (null (filter ( equMIDOffering mid') sl))
    isQuerying sl mid' = not (null (filter ( equMIDQuerying mid') sl))
     isOfferor sl aid ' = not (null (filter ( equAIDOfferor aid') sl))
     isTransferee sl aid ' = not (null (filter ( equAIDTransferee aid') sl))
      isPlaintiff sl aid ' = not (null (filter ( equAIDPlaintiff aid') sl))
    isDefendant sl aid ' = not (null (filter ( equAIDDefendant aid') sl))
    isContracting  sl  mid' = not (null (filter ( equMIDContracting mid') sl))
    isApplying  sl  mid' = not (null (filter ( equMIDApplying mid') sl))
    isTransferring  sl  mid' = not (null (filter ( equMIDTransferring mid') sl))
    isCourtAgent sl  aid ' = not (null (filter ( equAIDCourtAg aid') sl))
    isSuing  sl  mid' = not (null (filter ( equMIDSuing mid') sl))
    isServingComplaint  sl  mid' = not (null (filter ( equMIDServingComplaint mid') sl))
    isPronouncingJudgement sl mid' = not (null (filter ( equMIDPronouncingJudgement mid') sl))
    isSheriffAgent  sl  aid ' = not (null (filter ( equAIDSheriffAg aid')  sl ))
    isExecutingJudgement sl  mid' = not (null (filter ( equMIDExecutingJudgement mid') sl))

    getLastOwnerStatus  slist  land = if (length  slist3  <1) then
        error "Owner unknown in getLastOwnerStatus"else erg where
        erg = last  slist3
        slist3  = qsortOwner slist2
        slist2  = filter  ( isOwner land)  slist
        isOwner l ( Owner aid lid rmsg ws) = lid == l
        isOwner _ _ = False
        −− quicksort for the  last  owner
        qsortOwner  [] = []
        qsortOwner [( Owner aid lid  rmsg ws)] = [( Owner aid lid  rmsg ws)]
        qsortOwner (( Owner aid lid   rmsg ws): list ) =
            ( qsortOwner  left ) ++ [( Owner aid lid rmsg ws)] ++ ( qsortOwner right) where
            left  = filter  ( gr  ws)  list
            right  = filter  (  klgl  ws)  list
            gr  ws ( Owner aid'  lid '  rmsg'  ws') = ws > ws'
            klgl  ws ( Owner aid1 lid2 rmsg2 ws3) = ws ≤ ws3
        qsortOwner _ = error "wrong status in qsortOwner"

    getLastOwner  slist  land = name where
        name = getO lowner
        lowner = getLastOwnerStatus  slist  land
        getO ( Owner aid lid  rmsg ws) = aid

    getOwnerNow (( Owner aid lid rmsg ws):rest) land ws' = if ( lid  == land && ws == ws') then aid
        else getOwnerNow rest land ws'
    getOwnerNow [] land ws' = ""
    getOwnerNow (_:rest) land ws' = getOwnerNow rest land ws'

    getJudgementID [] _ _ _ = NoMessage
    getJudgementID (( Execution_Title aid'  luser '  lid '  msgid ws): rest )  aid  luser  lid  =
        if  ( lid==lid' && aid==aid' && luser==luser') then msgid
            else ( getJudgementID rest aid  luser  lid )
    getJudgementID (_: rest )  aid  luser  lid  = getJudgementID rest aid luser  lid

    hasExecutionTitle  []  _ _ _ = False
    hasExecutionTitle  (( Execution_Title  aid'  luser '  lid '  msgid ws ): rest )  aid  luser  lid  =
        if  ( lid==lid' && aid==aid' && luser==luser') then True
            else ( hasExecutionTitle  rest  aid  luser  lid )
    hasExecutionTitle  ( _: rest )  aid  luser  lid  = hasExecutionTitle  rest  aid  luser  lid

class Dutytech d where
    getDutyWS :: d →  WorldState
    isCurrentDuty  ::  WorldState → d → Bool

instance Dutytech Duty where

    getDutyWS (AskRegAgent mid ws) = ws
    getDutyWS (AnswerQuery mid ws) = ws
    getDutyWS (AnswerApplication mid ws) = ws
    getDutyWS (DoApplication mid ws) = ws
    getDutyWS (DoServeComplaint mid ws) = ws
    getDutyWS (DoJudgement mid ws) = ws
    getDutyWS (DoAnswerComplaint mid ws) = ws
    getDutyWS (DoAbandonLand mid ws) = ws
    getDutyWS (DoExecution mid ws) = ws
    getDutyWS NoDuty = error "getDutyWS: No worldstate in NoDuty"
    getDutyWS _ = error "no case in getDutyWS"


    isCurrentDuty ws1 duty = (getDutyWS duty) ≥ws1

class  Objectivetech  g where
    getObjectiveWS ::  g → WorldState
    isCurrentObjective  ::  WorldState → g → Bool

instance Objectivetech  Objective  where
    getObjectiveWS (AnswerOffer mid ws) = ws
    getObjectiveWS ( SellParcel  l  ai  ws) = ws
    getObjectiveWS (UseLand l ws) = ws
    getObjectiveWS (ApplyExecution lid  aid  ws) = ws
    getObjectiveWS (DoComplaint lid aid ws) = ws

    getObjectiveWS NoObjective = error "getObjectiveWS: No worldstate in NoObjective"
    getObjectiveWS _ = error "no case in getObjectiveWS"


    isCurrentObjective  ws1  objective  = (getObjectiveWS  objective )  ≥ws1
```

A.5 Percepts, actions and messages

```
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
−− actpercepts.hs
−− operations on percepts and acts
−− Steffen Bittner / 19.10.00
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

module Actpercepts where
```

```
import Agentdata

--import Basics-
--import Statusdata
import Message
import Landdata
---------------------------------------------------------------


class MyPercepts p msg phy | p → msg, p → phy where

    isMessagePercept :: p → Bool
    getMessagePercept :: p → msg
    getMessagesfromPercepts :: [ p] → [msg]
    getPhyPerceptsfromPercepts :: [ p] → [phy]
    isPhyPercept :: p → Bool
    getPhyPerceptPercept :: p → phy
    getPhyPercept :: p → phy
    genCPercept :: msg → p
    genCPercepts :: [ msg] → [p]
    genPPercept :: phy → p
    genPPercepts :: [ phy] → [p]


instance MyPercepts Percept Message PhyPercept where

    isMessagePercept (CPercept _ ) = True
    isMessagePercept _ = False

    getMessagePercept (CPercept p ) = p

    getMessagesfromPercepts percepts = map (getMessagePercept) (filter isMessagePercept percepts)

    isPhyPercept (PPercept _ ) = True
    isPhyPercept _ = False

    getPhyPerceptPercept (PPercept p ) = p


    getPhyPerceptsfromPercepts percepts = map (getPhyPerceptPercept) (filter isPhyPercept percepts)

    getPhyPercept (PPercept p ) = p

    genCPercept m = CPercept m

    genCPercepts msgs = map genCPercept msgs

    genPPercept m = PPercept m

    genPPercepts perc = map genPPercept perc


class Acts a msg act | a → msg, a → act where
    isMessageAction :: a → Bool
    getMessageAction :: a → msg
    getMessagesfromActions :: [ a] → [msg]
    isActAction :: a → Bool
    getActAction :: a → act
    getActionsfromActs :: [ a] → [act]


instance Acts Act Message Action where
    isMessageAction (CAct _ ) = True
    isMessageAction _ = False

    getMessageAction (CAct p ) = p

    getMessagesfromActions actions = map (getMessageAction) (filter isMessageAction actions)

    isActAction (PAct _ ) = True
    isActAction _ = False

    getActAction (PAct p ) = p

    getActionsfromActs actions = map (getActAction) (filter isActAction actions)




---------------------------------------------------------------
---------------------------------------------------------------
-- update of the internal state based on the physical percepts
--
--
-- Steffen Bittner / 15.05.01
---------------------------------------------------------------

module Phypercepts where

import Basics
import Statusdata
import Agentdata
import Statustech
import Message


class Phypercepts s p where
```

```
    updStatePhyPercept :: [ s ] → AgentID → WorldState → p → Objective
    updStatePhyPercepts :: [ s ] → AgentID → WorldState → [p] → [ Objective ]

instance Phypercepts Status PhyPercept where

    updStatePhyPercept slist aid ws (LandUse aid' luser land ) =
         if ( hasExecutionTitle slist aid luser land) then
              (ApplyExecution land luser ws) else
                   (DoComplaint land luser ws)


    updStatePhyPercept _ _ _ NoPhyPercept = error "NoPhyAction in updStatePhyPercept"
    updStatePhyPercept _ _ _ _ = error "no case found in updStatePhyPercept"

    updStatePhyPercepts slist aid ws percepts = map (updStatePhyPercept slist aid ws) percepts
-------------------------------------------------------------
-- message.hs
-- message, content and message exchange
-- Steffen Bittner / 28.09.00
-------------------------------------------------------------

module Message where

import Basics

-------------------------------------------------------------

data MessageType = Offer
                              | Reject_Offer
                              | Accept_Offer
                              | Application
                              | Accept_Application
                              | Reject_Application
                              | Query
                              | Answer_Query
                              | Legal_Action
                              | Serve_Complaint
                              | Answer_Complaint
                              | Judgement
                              | Reject_Complaint
                              | Apply_Execution
                              | Accept_Execution
                              | Reject_Execution deriving (Eq,Show)

data Content = Sell AgentID AgentID LandID
                        | Query_Owner AgentID LandID
                        | Complaint AgentID AgentID LandID
                        | NoContent deriving (Show, Eq)

class Contents c where
    getLandIDfromContent :: c → LandID
    getSellerIDfromContent :: c → AgentID
    getBuyerIDfromContent :: c → AgentID
    getDefendantIDfromContent :: c → AgentID
    getPlaintiffIDfromContent :: c → AgentID

instance Contents Content where

    getLandIDfromContent (Query_Owner aid lid) = lid
    getLandIDfromContent (Sell sid bid lid ) = lid
    getLandIDfromContent (Complaint pid die lid) = lid
    getLandIDfromContent _ = error "wrong content in getLandIDfromContent"

    getSellerIDfromContent ( Sell seller buyer lid ) = seller
    getSellerIDfromContent _ = error "wrong content in getSellerIDfromContent"

    getBuyerIDfromContent (Sell seller buyer lid ) = buyer
    getBuyerIDfromContent _ = error "wrong content in getBuyerIDfromContent"

    getDefendantIDfromContent (Complaint plaintiff defendant landid) = defendant
    getDefendantIDfromContent _ = error "wrong content in getDefendantIDfromContent"

    getPlaintiffIDfromContent ( Complaint plaintiff defendant landid) = plaintiff
    getPlaintiffIDfromContent _ = error "wrong content in getPlaintiffIDfromContent"

data Message = Message AgentID AgentID Content MessageType Message
                        | NoMessage deriving (Eq,Show)

class Messages m where
    isReceiver :: AgentID → m → Bool
    getMessagesByReceiverID :: AgentID → [m] → [m]
    newMessage :: AgentID → AgentID → Content → MessageType → m
    getContent :: m → Content
    getReceiver :: m → AgentID
    getSender :: m → AgentID
    isMessagefrom :: AgentID → m → Bool
    isOwnershipTransfer :: m → AgentID → LandID → Bool
    isExecutionTitle :: m → AgentID → AgentID → LandID → Bool

instance Messages Message where

    isReceiver aid (Message start ziel c t rmsg) = ziel == aid

    getMessagesByReceiverID aid msgs = msgs' where
        msgs' = filter ( isReceiver aid) msgs

    newMessage startid goalid cont mtype = (Message startid goalid cont mtype NoMessage)

    getContent (Message s z c t rmsg) = c
```

```
 getReceiver (Message s z c t rmsg) = z
 getSender (Message s z c t rmsg) = s
 isMessagefrom from' (Message from to c t rmsg) = from == from'

 isOwnershipTransfer
     (Message from to (Sell seller buyer land) Accept_Application rmsg) aid lid =
         ((buyer == aid) && (lid == land))
 isOwnershipTransfer _ _ _ = False

 isExecutionTitle
     (Message from to (Complaint plaintiff defendant land) Judgement rmsg) pid did lid =
         (( plaintiff == pid) && (lid == land) && defendant==did)
 isExecutionTitle _ _ _ _= False
```

A.6 Land and physical activities

```
----------------------------------------------------------------
-- landdata.hs
-- data type land
-- Steffen Bittner / 14.02.01
----------------------------------------------------------------

module Landdata where

import Basics

----------------------------------------------------------------


data Land = Land LandID AgentID deriving (Eq,Show)

data Action = ActUseLand LandID AgentID
            | ActAbandonLand LandID
            | ActEvictLand LandID
            | NoAction deriving (Eq,Show)
----------------------------------------------------------------
-- land.hs
-- operations on land
-- Steffen Bittner / 28.09.00
----------------------------------------------------------------

module Land where

import Basics
import Landdata
import IDs
import Agentdata


class Lands l where
    getLID :: l → LandID
    getLUser :: l → AgentID
    newLand :: LandID → l
    existLand :: LandID → [l] → Bool
    replaceLand :: LandID → AgentID → [l] → [l]

    updateLandsByAction :: Action → [l] → [l]

    updateLandsByActions :: [Action] → [l] → [l]

    canUseLand :: LandID → AgentID → [l] → Bool
    getLandUser :: LandID → [l] → AgentID

    genReactions :: [l] → [Action] → [EnvReaction]
    genReaction ::  [l] → Action → EnvReaction


instance Lands Land where

    getLUser (Land lID aid) = aid

    newLand lid = (Land lid "")

    replaceLand lid aid llist = (myland: llist') where
        llist' = filter (nequalsID lid) llist
        myland = (Land lid aid)

    updateLandsByAction (ActUseLand lid aid) llist = if (canUseLand lid aid llist) then
        replaceLand lid aid llist else llist

    updateLandsByAction (ActAbandonLand lid) llist = replaceLand lid ("") llist

    updateLandsByAction (ActEvictLand lid) llist  = replaceLand lid ("") llist

    updateLandsByAction _ _ = error "no valid action in updateLandsByAction"

    updateLandsByActions alist lands = foldl (flip (updateLandsByAction)) lands alist

    canUseLand lid aid lands = erg where
        uname = getLandUser lid lands
        erg = ((uname == aid) || (uname == ""))

    getLandUser lid lands = uname where
        land = getObjbyID lands lid
        uname = getLUser land

    genReaction lands (ActUseLand lid aid) = if ( not( canUseLand lid aid lands)) then
```

```
        (LandUse aid (getLandUser lid lands)  lid ) else NoPhyPercept

    genReaction _ _ = NoPhyPercept --no reaction, phypercepts = env reactions

    genReactions lands  actions  = erg where
        erg = filter (NoPhyPercept /=) erg'
        erg' = map (genReaction lands) actions
```

A.7 The simulation interface

```
-----------------------------------------------------------------
-- sim.hs
-- simulation of behaviour of agents in  realtiy :
-- exapmples: transfer of ownership, complaints, execution
-- Steffen Bittner / 28.09.00
-----------------------------------------------------------------

module Sim where

import World
import Worlddata
import Worldtech
import Agent
import Agentdata
import Agenttech
import Actpercepts
import Land
import Landdata
import Statusdata
import Message
import Extcosts

import Statustech
import Basics
import Construles
import IDs

-- Simulation of ownership transfer

initWorldOT :: World → World
initWorldOT w = w' where
    w' = addAgent w1 regAg
    w1 = addAgent w2 seller
    w2 = addAgent w3 buyer
    w3 = addLand w mguertel
    mguertel = newLand "P"
     seller  = createSellerAgent
    buyer = createBuyerAgent
    regAg = createRegAgOT

createRegAgOT :: Agent
createRegAgOT = regAg' where
    regAg' = addBelief regAg1 (Owner "A""P" NoMessage 0)
    regAg1 = addBelief regAg2 (RegAg "RegAg" 0)
    regAg2 = addBelief regAg4 (Parcel "P" 0)
    regAg4 = addBelief regAg5 (Legal_person "B" 0)
    regAg5 = addBelief regAg (Legal_person "A" 0)
    regAg = newAgt "RegAg"::Agent

createBuyerAgent :: Agent
createBuyerAgent = buyer' where
    buyer' = addBelief buyer3 (Legal_person "B" 0)
    buyer3 = addBelief buyer2 (Legal_person "A" 0)
    buyer2 = addBelief buyer1 (RegAg "RegAg" 0)
    buyer1 = addBelief buyer (Parcel "P" 0)
    buyer = newAgt "B"::Agent

 createSellerAgent  ::  Agent
 createSellerAgent  = seller ' where
     seller ' = addBelief  seller5  (Legal_person "B" 0)
     seller5  = addBelief  seller4  (Legal_person "A" 0)
     seller4  = addBelief  seller3  (Owner "A""P" NoMessage 0)
     seller3  = addBelief  seller2  (RegAg "RegAg" 0)
     seller2  = addBelief  seller1  (Parcel "P" 0)
     seller1  = addObjective  seller   objective
     objective  = (SellParcel "P" "B" 0)
     seller  = newAgt "A"::Agent

simOT n = output where
    output = putStrLn ( concat (map showWorld wlist) ++ showWorldCosts wlist)
    wlist  = doWorld [w2] n
    w2 = initWorldOT w3
    w3 = startWorld

-- Simulation of conflicts regarding land use:

initWorldU :: World → World
initWorldU w = w' where
    w' = addAgent w1 courtAg
    w1 = addAgent w2 auser
    w2 = addAgent w3 uuser
    w3 = addAgent w4 sheriff
    w4 = addLand w5 mguertel
    w5 = addAgent w regAg
    mguertel = newLand "P"
    courtAg = createCourtAgent
    uuser = createUnauthorizedUser
    auser = createAuthorizedUser
    sheriff  = createSheriffAg
    regAg = createRegAgU

createSheriffAg  ::  Agent
createSheriffAg  = sAg' where
    sAg' = addBelief sAg1 (CourtAg "CourtAg"0 )
    sAg1 = addBelief sAg2 (SheriffAg  "SheriffAg"  0)
    sAg2 = addBelief sAg3 (RegAg "RegAg"0)
    sAg3 = addBelief sAg4 (Parcel  "P" 0)
    sAg4 = addBelief sAg5 (Legal_person  "A" 0)
    sAg5 = addBelief sAg (Legal_person  "B" 0)
    sAg = newAgt "SheriffAg"::Agent

createRegAgU :: Agent
createRegAgU = regAg' where
    regAg' = addBelief regAg0 (CourtAg "CourtAg"0 )
    regAg0 = addBelief regAg1 (Owner "A""P" NoMessage 0)
    regAg1 = addBelief regAg2 (SheriffAg  "SheriffAg"  0)
    regAg2 = addBelief regAg3 (RegAg "RegAg"0)
    regAg3 = addBelief regAg4 (Parcel  "P" 0)
    regAg4 = addBelief regAg5 (Legal_person  "A" 0)
    regAg5 = addBelief regAg6 (Legal_person  "B" 0)
    regAg6 = addMessageActions regAg (Message "RegAg""A" (Sell "X""A""P")  Accept_Application  NoMessage)
    regAg = newAgt "RegAg"::Agent

createCourtAgent  ::  Agent
createCourtAgent = courtAg' where

    courtAg' = addBelief courtAg1 (CourtAg "CourtAg"0 )
    courtAg1 = addBelief courtAg2 (SheriffAg  "SeriffAg"  0)
    courtAg2 = addBelief courtAg3 (RegAg "RegAg"0)
    courtAg3 = addBelief courtAg4 (Parcel  "P" 0)
    courtAg4 = addBelief courtAg5 (Legal_person  "A" 0)
    courtAg5 = addBelief courtAg (Legal_person  "B" 0)
    courtAg = newAgt "CourtAg"::Agent

createUnauthorizedUser  ::  Agent
createUnauthorizedUser = uuser' where
    uuser' = addBelief uuser7 (CourtAg "CourtAg"0 )
    uuser7 = addBelief uuser6 (RegAg "RegAg"0)
    uuser6 = addBelief uuser5 (Legal_person  "A" 0)
    uuser5 = addBelief uuser3 (Legal_person  "B" 0)
    uuser3 = addBelief uuser2 (SheriffAg  "SheriffAg"  0)
    uuser2 = addBelief uuser1 (Parcel  "P" 0)
    uuser1 = addObjective uuser0  objective2
    uuser0 = addObjective uuser  objective1
     objective1  = (UseLand "P"0 )
     objective2  = (UseLand "P"6 )
    uuser = newAgt "B"::Agent

createAuthorizedUser  ::  Agent
createAuthorizedUser  = auser' where
    auser' = addBelief auser7 (CourtAg "CourtAg"0 )
    auser7 = addBelief auser6 (RegAg "RegAg"0)
    auser6 = addBelief auser5 (Legal_person  "A" 0)
    auser5 = addBelief auser3 (Legal_person  "B" 0)

    auser3 = addBelief auser2 (SheriffAg  "SheriffAg"  0)
    auser2 = addBelief auser1 (Parcel  "P" 0)
    auser1 = addObjective auser0  objective2
    auser0 = addObjective auser  objective1
     objective2  = (UseLand "P"7 )
     objective1  = (UseLand "P"1 )
    auser = newAgt "A"::Agent

simU :: Int → IO()
simU n = output where
    output = putStrLn ( concat (map showWorld wlist) ++ showWorldCosts wlist)

     wlist  = doWorld [w2] n
    w2 = initWorldU w3
    w3 = startWorld

showAgent :: Agent → String
showAgent (Agent aid percepts  ( IntState  status  duties  objectives )  actions   cost) = "\nAgent( AID="++ aid
    ++ "\n INBOX="++ show (getMessagesfromPercepts percepts) ++ "\n PHYPERCEPS="
    ++ show (getPhyPerceptsfromPercepts percepts)
    ++ "\n STATUS="++ (showStatusList status) ++ "\n DUTIES="++ show duties ++ "\n OBJECTIVES="++ show objectives
    ++ "\n ACTIONS="++ show (getActionsfromActs actions) ++ "\n OUTBOX="++ show (getMessagesfromActions actions)
 -- ++ "\n COSTS= " ++ show cost
    ++ " )"

showAgents  :: [ Agent] → String
showAgents (a: alist ) = showAgent a ++ "\n"++ showAgents alist
showAgents [] = []

showLands :: [Land] → String
showLands (a: alist ) = show a ++ "\n"++ showLands alist
showLands [] = []


showWorld (World agents lands ws) = "\n ------------- WorldState: "
    ++ myws ++ "-------------\n"
    ++ showAgents agents ++ "\n"++ showLands lands
    where myws = if (ws == 0) then "initial worldstate"else show (ws -1)



showStatusList  :: [ Status] → String
showStatusList  sl  = concat (map showStatus sl)
```

```
showStatus :: Status → String
showStatus (Owner aid lid msg ws) = "(Owner "++ show aid ++ " "++ show lid ++ " <m> "++show ws ++") "
showStatus (Legal_person aid ws) = "(Legal_person "++ show aid ++ " "++show ws ++") "
showStatus (Parcel lid ws) = "(Parcel "++ show lid ++ " "++show ws ++") "
showStatus (RegAg aid ws) = "(RegAg "++ show aid ++ " "++show ws ++") "
showStatus (CourtAg aid ws) = "(CourtAg "++ show aid ++ " "++show ws ++") "
showStatus (SheriffAg aid ws) = "(SheriffAg "++ show aid ++ " "++show ws ++") "
showStatus (Offeror aid msg ws) = "(Offeror "++ show aid ++ " <m> "++show ws ++") "
showStatus (Transferee aid msg ws) = "(Transferee "++ show aid ++ " <m> "++show ws ++") "
showStatus (Buyer aid msg ws) = "(Buyer "++ show aid ++ " <m> "++show ws ++") "
showStatus (Seller aid msg ws) = "(Seller "++ show aid ++ " <m> "++show ws ++") "
showStatus (Applicant aid msg ws) = "(Applicant "++ show aid ++ " <m> "++show ws ++") "
showStatus ( Application_receiver aid msg ws) = "( Application_receiver "++ show aid ++ " <m> "++show ws ++") "
showStatus (Query_sender aid msg ws) = "(Query_sender "++ show aid ++ " <m> "++show ws ++") "
showStatus (Query_receiver aid msg ws) = "(Query_receiver "++ show aid ++ " <m> "++show ws ++") "
showStatus ( Plaintiff aid msg ws) = "( Plaintiff "++ show aid ++ " <m> "++show ws ++") "
showStatus (Defendant aid msg ws) = "(Defendant "++ show aid ++ " <m> "++show ws ++") "
showStatus (Judge aid msg ws) = "(Judge "++ show aid ++ " <m> "++show ws ++") "
showStatus (Loser_Complaint aid msg ws) = "(Loser_Complaint "++ show aid ++ " <m> "++show ws ++") "
showStatus (JudgementExecutor aid msg ws) = "(JudgementExecutor "++ show aid ++ " <m> "++show ws ++") "
showStatus (Applicant_Execution aid msg ws) = "(Applicant_Execution "++ show aid ++ " <m> "++show ws ++") "
showStatus (Querying msg ws) = "(Querying"++ " <m> "++show ws ++") "
showStatus (Offering msg ws) = "(Offering"++ " <m> "++show ws ++") "
showStatus (Contracting msg ws) = "(Contracting"++ " <m> "++show ws ++") "
showStatus (Applying msg ws) = "(Applying"++ " <m> "++show ws ++") "
showStatus (Transferring msg ws) = "(Transferring"++ " <m> "++show ws ++") "
showStatus (Suing msg ws) = "(Suing"++ " <m> "++show ws ++") "
showStatus (ServingComplaint msg ws) = "(ServingComplaint"++ " <m> "++show ws ++") "
showStatus (PronouncingJudgement msg ws) = "(PronouncingJudgement"++ " <m> "++show ws ++") "
showStatus (ExecutingJudgement msg ws) = "(ExecutingJudgement"++ " <m> "++show ws ++") "
showStatus (Execution_Title aid aid' lid msg ws) = "(Execution_Title "++ show aid ++ " "++ show aid' ++ " "
              ++ show lid ++ " <m> "++show ws ++") "
showStatus (NoStatus) = "(NoStatus) "
showStatus _ = error "no case found in showStatus"
```

A.8 Cost assessment

```
-----------------------------------------------------------------
-- extcosts.hs
-- external costs in the simulation
-- Steffen Bittner / 28.09.00
--
-----------------------------------------------------------------

-- counting of costs based on the actions and percepts of the agents


module Extcosts where

import Landdata
import Message
import Agentdata
import Worlddata
import Basics
import IDs



class Num c ⇒ ExtCosts e c | e → c where
    cost :: e → c
    gesCost :: [ e] → c

    isExchangeCost :: e → Bool
    isMonitoringCost :: e → Bool
    isMeasurementCost :: e → Bool
    isEnforcementCost :: e → Bool

    genExchangeCosts :: [e] → c
    genMonitoringCosts :: [e] → c
    genMeasurementCosts :: [e] → c
    genEnforcementCosts :: [e] → c


    gesCost [] = 0
    gesCost e = foldl (+) 0 (map cost e)

    genExchangeCosts = gesCost. (filter isExchangeCost)
    genMeasurementCosts = gesCost. (filter isMeasurementCost)
    genMonitoringCosts = gesCost. (filter isMonitoringCost)
    genEnforcementCosts = gesCost. (filter isEnforcementCost)



instance ExtCosts Act Cost where

    cost (PAct (ActUseLand _ _ )) = 5
    cost (PAct (ActAbandonLand _ )) = 5
    cost (PAct (ActEvictLand _ )) = 5

    cost (CAct (Message aid aid' content Offer rid)) = 2
    cost (CAct (Message aid aid' content Reject_Offer rid)) = 2
    cost (CAct (Message aid aid' content Accept_Offer rid)) = 2
    cost (CAct (Message aid aid' content Application rid)) = 10 -- application costs
    cost (CAct (Message aid aid' content Accept_Application rid)) = 2
    cost (CAct (Message aid aid' content Reject_Application rid)) = 2
    cost (CAct (Message aid aid' content Query rid)) = 4 --query costs
    cost (CAct (Message aid aid' content Answer_Query rid)) = 2
    cost (CAct (Message aid aid' content Legal_Action rid)) = 2
    cost (CAct (Message aid aid' content Serve_Complaint rid)) = 2
    cost (CAct (Message aid aid' content Answer_Complaint rid)) = 2
    cost (CAct (Message aid aid' content Judgement rid)) = 2
    cost (CAct (Message aid aid' content Apply_Execution rid)) = 2
    cost (CAct (Message aid aid' content Accept_Execution rid)) = 2
    cost (CAct (Message aid aid' content Reject_Execution rid)) = 2
    cost _ = error "Cost not defined in ExtCosts CAct ExtCost:cost"

    isExchangeCost (PAct (ActUseLand _ _)) = True


    isExchangeCost (CAct (Message aid aid' content Offer rid)) = True
    isExchangeCost (CAct (Message aid aid' content Reject_Offer rid)) = True
    isExchangeCost (CAct (Message aid aid' content Accept_Offer rid)) = True
    isExchangeCost _ = False


    isMonitoringCost (CAct (Message aid aid' content Application rid)) = True
    isMonitoringCost (CAct (Message aid aid' content Accept_Application rid)) = True
    isMonitoringCost (CAct (Message aid aid' content Reject_Application rid)) = True
    isMonitoringCost _ = False

    isMeasurementCost (CAct (Message aid aid' content Query rid)) = True
    isMeasurementCost (CAct (Message aid aid' content Answer_Query rid)) = True
    isMeasurementCost _ = False

    isEnforcementCost (PAct (ActAbandonLand _)) = True
    isEnforcementCost (PAct (ActEvictLand _)) = True

    isEnforcementCost (CAct (Message aid aid' content Legal_Action rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Serve_Complaint rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Answer_Complaint rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Judgement rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Apply_Execution rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Accept_Execution rid)) = True
    isEnforcementCost (CAct (Message aid aid' content Reject_Execution rid)) = True
    isEnforcementCost _ = False



instance ExtCosts Percept Cost where

    cost (PPercept (LandUse _ _ _)) = 0

    cost (CPercept (Message aid aid' content Offer rid)) = 0
    cost (CPercept (Message aid aid' content Reject_Offer rid)) = 0
    cost (CPercept (Message aid aid' content Accept_Offer rid)) = 0
    cost (CPercept (Message aid aid' content Application rid)) = -10 -- receiving an application brings money
    cost (CPercept (Message aid aid' content Accept_Application rid)) = 0
    cost (CPercept (Message aid aid' content Reject_Application rid)) = 0
    cost (CPercept (Message aid aid' content Query rid)) = -4 --receiving a query brings money
    cost (CPercept (Message aid aid' content Answer_Query rid)) = 0

    cost (CPercept (Message aid aid' content Legal_Action rid)) = 0
    cost (CPercept (Message aid aid' content Serve_Complaint rid)) = 0
    cost (CPercept (Message aid aid' content Answer_Complaint rid)) = 0
    cost (CPercept (Message aid aid' content Judgement rid)) = 0
    cost (CPercept (Message aid aid' content Apply_Execution rid)) = 0
    cost (CPercept (Message aid aid' content Accept_Execution rid)) = 0
    cost (CPercept (Message aid aid' content Reject_Execution rid)) = 0
    cost _ = error "Cost not defined in ExtCosts CPerept ExtCost:cost"

    isExchangeCost _ = False

    isMonitoringCost (CPercept (Message aid aid' content Application rid)) = True
    isMonitoringCost _ = False

    isMeasurementCost (CPercept (Message aid aid' content Query rid)) = True
    isMeasurementCost _ = False

    isEnforcementCost _ = False



genExtCosts :: Agent → Agent
genExtCosts (Agent aid percepts intstate actions (CostData ex mon mea enf)) =
    (Agent aid percepts intstate actions (CostData ex' mon' mea' enf')) where
        ex' = ex + (genExchangeCosts percepts) + (genExchangeCosts actions)
        mon' = mon + (genMonitoringCosts percepts) + (genMonitoringCosts actions)
        mea' = mea + (genMeasurementCosts percepts) + (genMeasurementCosts actions)
        enf' = enf + (genEnforcementCosts percepts) + (genEnforcementCosts actions)


sumCosts :: CostData → CostData → CostData
sumCosts (CostData ex mo me en) (CostData ex' mo' me' en') = (CostData (ex+ex') (mo+mo') (me+me') (en+en'))

getAgentCost :: Agent → (AgentID,CostData)
getAgentCost (Agent aid p i a c) = (aid,c)


getAgentCosts :: World → [(AgentID,CostData)]
getAgentCosts (World a l ws) = (map getAgentCost a)


getCostbyID :: ( AgentID,CostData) → [(AgentID,CostData)] → CostData
getCostbyID a blist = snd (head(filter (equAID a) blist))
```

```
equAID :: (AgentID,CostData) → (AgentID,CostData) → Bool
equAID a b = fst a == fst b

addCost :: ( AgentID,CostData) → [(AgentID,CostData)] → (AgentID,CostData)
addCost (a,c)  blist  = (a, (sumCosts c c'))  where
           c' = getCostbyID (a,c)  blist

addCosts  :: [( AgentID,CostData)] → [(AgentID,CostData)] → [(AgentID,CostData)]
addCosts  alist   blist  = map ( flip addCost alist )  blist

getAbsAgentCosts  :: [ World] → [( AgentID, CostData)]
getAbsAgentCosts worlds =  costlist  where
     costlist  = foldl addCosts start  rest
    ( start : rest ) = map getAgentCosts worlds

showAbsAgentCost :: (AgentID, CostData) → String
showAbsAgentCost (aid, (CostData ex mo me en) ) = "("++ aid ++ ",Exchange="++ show ex ++",Monitoring=" ++
                              show mo ++",Measurement=" ++ show me ++",Enforcement=" ++ show en ++ ") "

showAbsAgentCosts :: [World] → String
showAbsAgentCosts wl = show (map showAbsAgentCost (getAbsAgentCosts wl) )

showWorldCosts :: [World] → String
showWorldCosts worlds = "\n********Costs*******\n"++ showAbsAgentCosts worlds
------------------------------------------------------------
-- Intcosts.hs
-- internal costs  in  the  simulation
-- Steffen Bittner / 16.02.01
------------------------------------------------------------

--counting of costs based  on  lifted   operations

module Intcosts where

import Statusdata
import Basics
import Message
import Landdata
import Actduties
import Actobjectives
import Statustech
import Agentdata
import Land
import Agent
import Extcosts

data OpCosts f = OpC f CostData deriving Show

getOpCosts  :: ( OpCosts f)  →  CostData
getOpCosts (OpC f t) = t

getOp  :: ( OpCosts f)  →  f
getOp (OpC f t) = f


setOpCosts  :: ( OpCosts f)  →  CostData → (OpCosts f)
setOpCosts (OpC f c)  cost  = (OpC f cost)



 liftCost   ::  f  →  CostData  →  OpCosts f
 liftCost  f  cost  = (  OpC f cost)


class ActdutiesL  s  d  aid  ws a  |  a → d,  a → s,  a → aid where
    doDutyL :: [s]  → aid  → ws → d → (OpCosts [a])
    doCurrentDutyL :: [s]  →  aid  →  ws →  d →  (OpCosts [a])


instance ActdutiesL Status Duty AgentID WorldState Act where
    doDutyL sl aid  ws (AnswerQuery mid ws') = liftCost (answerQuery sl aid ws mid) (CostData 0 0 2 0)
    doDutyL sl aid  ws (AskRegAgent mid ws' ) = liftCost ( askRegAgent sl  aid  ws mid) (CostData 0 0 1 0)
    doDutyL sl aid  ws (DoApplication mid ws' ) = liftCost  ( doApplication  sl  aid ws mid) (CostData 0 1 0 0)
    doDutyL sl aid  ws (AnswerApplication mid ws' ) = liftCost  ( answerApplication  sl  aid ws mid) (CostData 0 6 0 0)
    doDutyL sl aid  ws (DoServeComplaint mid ws' ) = liftCost (serveComplaint  sl  aid  ws mid) (CostData 0 0 0 1)
    doDutyL sl aid  ws (DoJudgement mid ws' ) =  liftCost  ( doJudgement sl aid ws mid) (CostData 0 0 0 1)
    doDutyL sl aid  ws (DoAnswerComplaint mid ws' ) = liftCost (answerComplaint sl aid ws mid) (CostData 0 0 0 1)
    doDutyL sl aid  ws (DoAbandonLand mid ws' ) = liftCost (abandonLand ws mid) (CostData 0 0 0 1)
    doDutyL sl aid  ws (DoExecution mid ws' ) = liftCost  ( doExecution  sl  aid  ws mid) (CostData 0 0 0 1)
    doDutyL _ _ _ NoDuty = error "NoDuty in doDutyL"
    doDutyL _ _ _ _  = error "No case found in doDutyL"

    doCurrentDutyL sl aid ws duty =
        if (  ws == (getDutyWS duty)) then doDutyL sl aid ws duty else (OpC [] (CostData 0 0 0 0))

class  ActobjectivesL  s  o  aid  ws a   where

    doObjectiveL  :: [ s]  → aid  → ws → o → (OpCosts [a])


instance ActobjectivesL Status  Objective  AgentID WorldState Act where
    doObjectiveL  sl  aid  ws ( SellParcel  land  aid'  ws') = erg where
        erg = if (ws==ws') then liftCost (doOffer sl  aid  aid'  land ws) (CostData 1 0 0 0) else ( OpC [] (CostData 0 0 0 0))

    doObjectiveL  sl  aid  ws (AnswerOffer mid ws' ) = erg where
        erg = if (ws==ws') then liftCost (answerOffer sl  aid ws mid) (CostData 1 0 0 0) else ( OpC [] (CostData 0 0 0 0))

    doObjectiveL  sl  aid  ws (DoComplaint land aid' ws') = erg where
        erg = if (ws==ws') then liftCost (doComplaint sl aid aid'  ws land) ( CostData 0 0 0 1) else ( OpC [] (CostData 0 0 0 0))
    doObjectiveL sl aid ws (ApplyExecution land aid' ws') = erg where
       erg = if (ws==ws') then liftCost (doApplyExecution sl aid aid' ws land) (CostData 0 0 0 1) else (OpC [] (CostData 0 0 0 0))

    doObjectiveL sl aid ws (UseLand land ws') = erg where
         erg = if (ws /= ws') then (OpC [] (CostData 0 0 0 0)) else liftCost (doActUseLand sl land aid) (CostData 1 0 0 0)

    doObjectiveL _ _ _ NoObjective = error "NoObjective in doObjectiveL"
    doObjectiveL _ _ _ _ = error "No case found in doObjectiveL"


class (UpdStateInMsgs aid ws p i, UpdStateOutMsgs ws a i) => MyAgentInternalsL aid ws p i a | i-> aid, i->ws, i -> p, i -> a where

    actDutiesL :: aid -> ws -> i -> [(OpCosts [a])]
    actObjectivesL :: aid -> ws -> i -> [(OpCosts [a])]

instance MyAgentInternalsL AgentID WorldState Percept IntState Act where

    actDutiesL aid ws (IntState slist duties objectives) = erglist where
        erglist = (map (doCurrentDutyL slist aid ws) duties)

    actObjectivesL aid ws (IntState slist duties objectives) = erglist where
        erglist = (map (doObjectiveL slist aid ws) objectives)


class (MyAgentInternalsL aid ws p i a, AgentInternals aid ws p i a) => AgentInternalsL aid ws p i a where


    selActsL :: aid -> ws -> i -> [(OpCosts [a])]
    decisionL :: aid -> ws -> [p] -> i -> ([(OpCosts [a])], i)


    selActsL aid ws i = (actDutiesL aid ws i) ++ (actObjectivesL aid ws i)


    decisionL aid ws p i =
      ( selActsL aid ws ( updStatePercepts aid ws p i),

         updStateActions ws (concat( map getOp (selActsL aid ws ( updStatePercepts aid ws p i)) )) ( updStatePercepts aid ws p i ))

instance AgentInternalsL AgentID WorldState Percept IntState Act

class (AgentInternalsL aid ws p i act ) => AbsAgentL a ws i p aid act | ws -> i, ws -> p, ws -> aid, ws -> act where

    doAgtL :: ws -> a -> a



instance AbsAgentL Agent WorldState IntState Percept AgentID Act where

    doAgtL ws (Agent aid percepts intstate actions cost) = (Agent aid percepts intstate' actions' cost') where

        cost' = foldl sumCosts cost costlist
         costlist = map getOpCosts actions1
        actions' = concat (map getOp actions1)
        (actions1, intstate') = decisionL aid ws percepts intstate2
        intstate2 = filterState intstate ws
```

Appendix B

Program outputs

B.1 Output of the ownership transfer simulation

```
Sim> simOT 7

 ------------- WorldState: initial worldstate-------------

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Legal_person "B" 0) (Legal_person "A" 0) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[SellParcel "P" "B" 0]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 0-------------

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A" 0) (Owner "A"
 "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[SellParcel "P" "B" 0]
 ACTIONS=[]
 OUTBOX=[Message "A" "B" (Sell "A" "B" "P") Offer NoMessage] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 1-------------

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A" 0) (Owner "A"
 "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "A" "B" (Sell "A" "B" "P") Offer NoMessage]
 PHYPERCEPS=[]
 STATUS=(Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1) (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m>
 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[AskRegAgent (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 1]
 OBJECTIVES=[AnswerOffer (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 3]
 ACTIONS=[]
 OUTBOX=[Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage] )

Land "P" ""

 ------------- WorldState: 2-------------

Agent( AID=RegAg
 INBOX=[Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage]
 PHYPERCEPS=[]
 STATUS=(Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0)
 (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[AnswerQuery (Message "B" "RegAg" (Query_Owner "" "P") Query NoMessage) 2]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "RegAg" "B" (Query_Owner "A" "P") Answer_Query NoMessage] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A" 0) (Owner "A"
 "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1) (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m>
 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[AnswerOffer (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 3]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 3-------------

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0)
 (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A" 0) (Owner "A"
 "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "RegAg" "B" (Query_Owner "A" "P") Answer_Query NoMessage]
 PHYPERCEPS=[]
 STATUS=(Seller "A" <m> 3) (Buyer "B" <m> 3) (Contracting <m> 3) (Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1)
 (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m> 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0)
(RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
 DUTIES=[]
 OBJECTIVES=[AnswerOffer (Message "A" "B" (Sell "A" "B" "P") Offer NoMessage) 3]
 ACTIONS=[]
 OUTBOX=[Message "B" "A" (Sell "A" "B" "P") Accept_Offer NoMessage] )

Land "P" ""

 ------------- WorldState: 4-------------
```

```
Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0)
 (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[Message "B" "A" (Sell "A" "B" "P") Accept_Offer NoMessage]
 PHYPERCEPS=[]
 STATUS=(Applicant "A" <m> 4) (Application_receiver "RegAg" <m> 4) (Applying <m> 4) (Seller "A" <m> 4) (Buyer "B" <m> 4)
 (Contracting <m> 4) (Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A"
0) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[DoApplication (Message "B" "A" (Sell "A" "B" "P") Accept_Offer NoMessage) 4]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "A" "RegAg" (Sell "A" "B" "P") Application NoMessage] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Seller "A" <m> 3) (Buyer "B" <m> 3) (Contracting <m> 3) (Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1)
 (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m> 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0)
(RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 5-------------

Agent( AID=RegAg
 INBOX=[Message "A" "RegAg" (Sell "A" "B" "P") Application NoMessage]
 PHYPERCEPS=[]
 STATUS=(Owner "B" "P" <m> 5) (Transferring <m> 5) (Transferring <m> 5) (Applicant "A" <m> 5) (Application_receiver "Reg
Ag" <m> 5) (Applying <m> 5) (Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m>
0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[AnswerApplication (Message "A" "RegAg" (Sell "A" "B" "P") Application NoMessage) 5]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "RegAg" "A" (Sell "A" "B" "P") Accept_Application NoMessage,Message "RegAg" "B" (Sell "A" "B" "P") Acce
pt_Application NoMessage] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Applicant "A" <m> 4) (Application_receiver "RegAg" <m> 4) (Applying <m> 4) (Seller "A" <m> 4) (Buyer "B" <m> 4)
 (Contracting <m> 4) (Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m> 0) (Legal_person "B" 0) (Legal_person "A"
0) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Seller "A" <m> 3) (Buyer "B" <m> 3) (Contracting <m> 3) (Query_sender "B" <m> 1) (Query_receiver "RegAg" <m> 1)
 (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m> 1) (Offering <m> 1) (Legal_person "B" 0) (Legal_person "A" 0)
(RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 6-------------

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "B" "P" <m> 5) (Transferring <m> 5) (Transferring <m> 5) (Applicant "A" <m> 5) (Application_receiver "Reg
Ag" <m> 5) (Applying <m> 5) (Query_sender "B" <m> 2) (Query_receiver "RegAg" <m> 2) (Querying <m> 2) (Owner "A" "P" <m>
0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "B" 0) (Legal_person "A" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[Message "RegAg" "A" (Sell "A" "B" "P") Accept_Application NoMessage]
 PHYPERCEPS=[]
 STATUS=(Owner "B" "P" <m> 6) (Transferring <m> 6) (Applicant "A" <m> 4) (Application_receiver "RegAg" <m> 4) (Applying
<m> 4) (Seller "A" <m> 4) (Buyer "B" <m> 4) (Contracting <m> 4) (Offeror "A" <m> 0) (Transferee "B" <m> 0) (Offering <m>
 0) (Legal_person "B" 0) (Legal_person "A" 0) (Owner "A" "P" <m> 0) (RegAg "RegAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "RegAg" "B" (Sell "A" "B" "P") Accept_Application NoMessage]
 PHYPERCEPS=[]
 STATUS=(Owner "B" "P" <m> 6) (Transferring <m> 6) (Seller "A" <m> 3) (Buyer "B" <m> 3) (Contracting <m> 3) (Query_sende
r "B" <m> 1) (Query_receiver "RegAg" <m> 1) (Querying <m> 1) (Offeror "A" <m> 1) (Transferee "B" <m> 1) (Offering <m> 1)
 (Legal_person "B" 0) (Legal_person "A" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Owner "A" "P" <m> 3)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

********Costs*******
["(RegAg,Exchange=0,Monitoring=0,Measurement=0,Enforcement=0) ","(A,Exchange=3,Monitoring=11,Measurement=0,Enforcement=0
) ","(B,Exchange=3,Monitoring=0,Measurement=5,Enforcement=0) "]

Sim>
```

B.2 Output of the conflict simulation

```
Sim> simU 11

 ------------- WorldState: initial worldstate-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_pers
on "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6,UseLand "P" 0]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage] )

Land "P" ""

 ------------- WorldState: 0-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_pers
on "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[Message "RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_p
erson "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6,UseLand "P" 0]
 ACTIONS=[ActUseLand "P" "B"]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

 ------------- WorldState: 1-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_pers
on "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_p
erson "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,UseLand "P" 1]
 ACTIONS=[ActUseLand "P" "A"]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 2-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_pers
on "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )
```

```
Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[LandUse "A" "B" "P"]
 STATUS=(Defendant "B" <m> 2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transfe
rring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0)
 (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7,DoComplaint "P" "B" 2]
 ACTIONS=[]
 OUTBOX=[Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message "RegAg" "A" (Sell "X" "A" "P") Accept_Appli
cation NoMessage)] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 3-------------

Agent( AID=CourtAg
 INBOX=[Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message "RegAg" "A" (Sell "X" "A" "P") Accept_Applic
ation NoMessage)]
 PHYPERCEPS=[]
 STATUS=(ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Cour
tAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[DoServeComplaint (Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message "RegAg" "A" (Sell "X" "A"
 "P") Accept_Application NoMessage)) 3,DoJudgement (Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message
"RegAg" "A" (Sell "X" "A" "P") Accept_Application NoMessage)) 5]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Serve_Complaint NoMessage] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Defendant "B" <m> 2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transfe
rring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0)
 (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Par
cel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )
```

```
Land "P" "B"

 ------------- WorldState: 4-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Cour
tAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[DoJudgement (Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message "RegAg" "A" (Sell "X" "A" "P")
 Accept_Application NoMessage)) 5]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Defendant "B" <m> 2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transfe
rring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0)
 (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Serve_Complaint NoMessage]
 PHYPERCEPS=[]
 STATUS=(Defendant "B" <m> 4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg"
0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[DoAnswerComplaint (Message "CourtAg" "B" (Complaint "A" "B" "P") Serve_Complaint NoMessage) 4]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[]
 OUTBOX=[Message "B" "CourtAg" (Complaint "A" "B" "P") Answer_Complaint NoMessage] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 5-------------

Agent( AID=CourtAg
 INBOX=[Message "B" "CourtAg" (Complaint "A" "B" "P") Answer_Complaint NoMessage]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[DoJudgement (Message "A" "CourtAg" (Complaint "A" "B" "P") Legal_Action (Message "RegAg" "A" (Sell "X" "A" "P")
 Accept_Application NoMessage)) 5]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Judgement NoMessage,Message "CourtAg" "A" (Complaint "A" "B" "P")
 Judgement NoMessage] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Defendant "B" <m> 2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transfe
rring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0)
 (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Defendant "B" <m> 4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg"
0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
```

```
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 6-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[Message "CourtAg" "A" (Complaint "A" "B" "P") Judgement NoMessage]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "Cour
tAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[Message "CourtAg" "B" (Complaint "A" "B" "P") Judgement NoMessage]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal
_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[DoAbandonLand (Message "CourtAg" "B" (Complaint "A" "B" "P") Judgement NoMessage) 6]
 OBJECTIVES=[UseLand "P" 6]
 ACTIONS=[ActAbandonLand "P",ActUseLand "P" "B"]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 7-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
2) (Plaintiff "A" <m> 2) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "Cour
tAg" 0) (RegAg "RegAg" 0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[UseLand "P" 7]
 ACTIONS=[ActUseLand "P" "A"]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal
_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 8-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[LandUse "A" "B" "P"]
 STATUS=(Applicant_Execution "A" <m> 8) (JudgementExecutor "SheriffAg" <m> 8) (ExecutingJudgement <m> 8) (Loser_Complain
t "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m> 2) (Plaintiff "A" <m> 2
) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg"
0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[ApplyExecution "P" "B" 8]
 ACTIONS=[]
 OUTBOX=[Message "A" "SheriffAg" (Complaint "A" "B" "P") Apply_Execution (Message "CourtAg" "A" (Complaint "A" "B" "P")
Judgement NoMessage)] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal
_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_per
son "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 9-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Applicant_Execution "A" <m> 8) (JudgementExecutor "SheriffAg" <m> 8) (ExecutingJudgement <m> 8) (Loser_Complain
t "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m> 2) (Plaintiff "A" <m> 2
) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg"
0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal
_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[Message "A" "SheriffAg" (Complaint "A" "B" "P") Apply_Execution (Message "CourtAg" "A" (Complaint "A" "B" "P") J
udgement NoMessage)]
 PHYPERCEPS=[]
 STATUS=(Applicant_Execution "A" <m> 9) (JudgementExecutor "SheriffAg" <m> 9) (ExecutingJudgement <m> 9) (CourtAg "Court
Ag" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[DoExecution (Message "A" "SheriffAg" (Complaint "A" "B" "P") Apply_Execution (Message "CourtAg" "A" (Complaint
"A" "B" "P") Judgement NoMessage)) 9]
 OBJECTIVES=[]
 ACTIONS=[ActEvictLand "P"]
 OUTBOX=[Message "SheriffAg" "A" (Complaint "A" "B" "P") Accept_Execution NoMessage] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" "B"

 ------------- WorldState: 10-------------

Agent( AID=CourtAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 5) (Execution_Title "A" "B" "P" <m> 5) (PronouncingJudgement <m> 5) (PronouncingJudgeme
nt <m> 5) (ServingComplaint <m> 3) (Defendant "B" <m> 3) (Plaintiff "A" <m> 3) (Judge "CourtAg" <m> 3) (Suing <m> 3) (Co
urtAg "CourtAg" 0) (SheriffAg "SeriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=A
 INBOX=[Message "SheriffAg" "A" (Complaint "A" "B" "P") Accept_Execution NoMessage]
 PHYPERCEPS=[]
 STATUS=(Applicant_Execution "A" <m> 8) (JudgementExecutor "SheriffAg" <m> 8) (ExecutingJudgement <m> 8) (Loser_Complain
t "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m> 2) (Plaintiff "A" <m> 2
) (Judge "CourtAg" <m> 2) (Suing <m> 2) (Owner "A" "P" <m> 0) (Transferring <m> 0) (CourtAg "CourtAg" 0) (RegAg "RegAg"
0) (Legal_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=B
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Loser_Complaint "B" <m> 6) (Execution_Title "A" "B" "P" <m> 6) (PronouncingJudgement <m> 6) (Defendant "B" <m>
4) (Plaintiff "A" <m> 4) (Judge "CourtAg" <m> 4) (ServingComplaint <m> 4) (CourtAg "CourtAg" 0) (RegAg "RegAg" 0) (Legal
_person "A" 0) (Legal_person "B" 0) (SheriffAg "SheriffAg" 0) (Parcel "P" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=SheriffAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(Applicant_Execution "A" <m> 9) (JudgementExecutor "SheriffAg" <m> 9) (ExecutingJudgement <m> 9) (CourtAg "Court
Ag" 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_person "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Agent( AID=RegAg
 INBOX=[]
 PHYPERCEPS=[]
 STATUS=(CourtAg "CourtAg" 0) (Owner "A" "P" <m> 0) (SheriffAg "SheriffAg" 0) (RegAg "RegAg" 0) (Parcel "P" 0) (Legal_pe
rson "A" 0) (Legal_person "B" 0)
 DUTIES=[]
 OBJECTIVES=[]
 ACTIONS=[]
 OUTBOX=[] )

Land "P" ""

********Costs*******
["(CourtAg,Exchange=0,Monitoring=0,Measurement=0,Enforcement=8) ","(A,Exchange=12,Monitoring=0,Measurement=0,Enforcement
=6) ","(B,Exchange=12,Monitoring=0,Measurement=0,Enforcement=9) ","(SheriffAg,Exchange=0,Monitoring=0,Measurement=0,Enfo
rcement=8) ","(RegAg,Exchange=0,Monitoring=0,Measurement=0,Enforcement=0) "]

Sim>
```