



DISSERTATION

Formalisierung von Gesetzen

Am Beispiel des Österreichischen Allgemeinen Grundbuchsgesetzes

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

Prof. Andreas Frank

E 127

Institut für Geoinformation und Landesvermessung

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

von

Gerhard Navratil

Matr.Nr. 8726525

Parkstraße 54

2170 Wetzelsdorf

Wien, am 23. Mai 2002

Für meine Eltern

Kurzfassung

Die vorliegende Arbeit untersucht Gesetze und Algebra. Auf den ersten Blick sehen diese beiden Themen sehr unterschiedlich aus. Gesetze sind Textwerke, die das Zusammenleben der Menschen regeln, Algebren sind eine mathematische Methode zur Beschreibung von Eigenschaften. Bei näherer Betrachtung fällt allerdings auf, dass in beiden Fällen Systeme modelliert werden und sich die verwendeten Methoden sehr ähnlich sind. In beiden Fällen wird mit Klassenbildung, Abstraktion und axiomatischer Definition gearbeitet.

Anhand eines Vergleiches der beiden Modellierungssysteme wird eine Methodik entwickelt, wie man Gesetze in Algebren übersetzen kann. Drei Schritte sind notwendig:

- Identifikation der Klassen
- Definition von Datentypen, Operationen und Axiomen
- Testen anhand einer einfachen Realisierung

Wichtig ist dabei vor allem, welchen Beitrag einzelne Paragraphen liefern. Ein Paragraph kann einen Datentyp spezifizieren, eine Operation beschreiben oder ein Axiom festlegen. Die Beschreibung der Operationen zerfällt wiederum in drei Bereiche. Einerseits muss die Notwendigkeit der Operation gegeben sein. Zusätzlich benötigt eine Operation Parameter und produziert ein Ergebnis. Schließlich muss die Operation auch noch einer Klasse zugeordnet werden. All diese Aufgaben werden von Paragraphen erfüllt. Dabei wirken oft mehrere Paragraphen zusammen um eine Aufgabe gemeinsam zu lösen.

Die entwickelte Methode wurde anhand eines Beispiels getestet. Das Testbeispiel ist das österreichische allgemeine Grundbuchsgesetz. Dieses wurde ohne Verwendung von Praxiswissen in ein algebraisches Modell überführt. Alle notwendigen Informationen wurden dabei dem allgemeinen Grundbuchsgesetz oder anderen Gesetzen (z.B. dem allgemeinen bürgerlichen Gesetzbuch) entnommen.

Die algebraische Umsetzung wurde in Haskell, einer funktionalen Programmiersprache, durchgeführt. Der Vorteil liegt in der mathematischen Reinheit der Sprache, die sie anderen Programmiersprachen voraus hat und in der Ausführbarkeit, die sie von der Mathematik abhebt.

Abstract

The thesis discusses laws and algebras. These two topics seem to be quite different. Laws are text books that regulate the life of humans. Algebras are a mathematical method for the description of properties. However, both describe systems and the methods used seem to be similar. The methods are in both cases classification, abstraction and axiomatic definition.

The thesis describes a method to translate laws into algebras. Starting point is a comparison of the modeling systems. The method requires three steps:

- Identification of the classes
- Definition of data types, operations, and axioms
- Testing with a simple model realisation

It is important to know the contribution of a single paragraph for the model. A paragraph can specify a data type, describe an operation, or define an axiom. The description of the operation consists of three parts. First the operation must be necessary. The operation then requires parameters and produces a result. Finally, the operation must be member of a class. Paragraphs fulfill all tasks. Rather frequently the combined content of several paragraphs solves one task.

The method developed was tested with an example. The example used here was the Austrian land registration law (allgemeines Grundbuchsgesetz). It was translated into an algebraic model without knowledge from the real system. All information was taken from the land registration law or other laws like the civil law code (allgemeines bürgerliches Gesetzbuch).

The algebraic modeling was done in Haskell, a functional programming language. The advantage of Haskell if compared to other programming languages is the mathematical purity. This would also be true for pure mathematics but Haskell is also executable which allows testing the code.

Danksagung

Zunächst möchte ich mich bei meinen Eltern bedanken. Sie haben mir die Ausbildung an der Universität ermöglicht und haben mich immer darin bestärkt, den eingeschlagenen Weg auch bis zum Ende zu beschreiten. Sie standen mir immer mit Rat und Tat, aber auch mit Trost, Ermutigung und Liebe zur Seite. Deshalb möchte ich ihnen auch diese Arbeit widmen.

Ich möchte auch meinem Betreuer Andrew Frank für seine geduldige Betreuung danken. Seine Kommentare halfen mir bei der Erarbeitung des Themas und beim Erfassen des Umfangs der Auswirkungen meiner Arbeit. Besonders dankbar bin ich ihm für die Anregungen, die er mir nach Lesen der Rohfassung der Arbeit gegeben hat und die sich stark in der vorliegenden Arbeit niederschlagen.

Bei meinem Zweitbetreuer, Christoph Twaroch möchte ich mich bedanken, dass er trotz seiner Tätigkeit im Bundesministerium f. Wirtschaft und Arbeit Zeit für meine Arbeit gefunden hat.

Ich möchte mich auch bei allen Kollegen am Institut für Geoinformation bedanken, mit denen ich während der letzten vier Jahre gearbeitet habe. Sie haben in Gesprächen und Diskussionen viel dazu beigetragen, dass ich neue Erkenntnisse gewonnen und unklare Teile des Themenkomplexes überarbeitet habe. Hervorzuheben ist vor allem Edith Unterweger, die mir bei allen administrativen Fragen zur Seite stand und die immer für Gespräche Zeit fand.

Schließlich möchte ich mich auch noch bei Susanne Eibl-Kosz bedanken, die durch Korrekturlesen der Endfassung für die Verbesserung meiner Ausdrucksweise gesorgt hat.

1	EINLEITUNG	1
1.1	MOTIVATION.....	1
1.1.1	<i>Probleme mit Gesetzen</i>	<i>1</i>
1.1.2	<i>Umsetzung von Gesetzen.....</i>	<i>4</i>
1.1.3	<i>Vorteile einer Formalisierung</i>	<i>5</i>
1.2	HYPOTHESE UND GLIEDERUNG DER ARBEIT	7
1.3	ABGRENZUNG DER ARBEIT	8
1.4	DAS BEISPIEL.....	8
2	GESETZE UND COMPUTER.....	11
2.1	AUFBAU EINER RECHTSORDNUNG	11
2.1.1	<i>Rechtsordnung</i>	<i>11</i>
2.1.2	<i>Rechtsquellen.....</i>	<i>12</i>
2.1.3	<i>Aufbau und Arten von Rechtsnormen</i>	<i>13</i>
2.1.4	<i>Gliederung der Rechtsordnung.....</i>	<i>13</i>
2.1.5	<i>Inkonsistenzen zwischen Rechtsnormen.....</i>	<i>15</i>
2.2	MODELLIERUNG MIT GESETZEN.....	16
2.3	SPEICHERUNG VON GESETZESTEXTEN IN DATENBANKEN	17
2.3.1	<i>Probleme dieser Lösung</i>	<i>18</i>
2.3.2	<i>Verbesserungen?.....</i>	<i>20</i>
2.3.3	<i>Semantische Suche.....</i>	<i>21</i>
2.3.4	<i>Einsatz beim Lösen weiterer Probleme.....</i>	<i>21</i>
2.4	EXPERTENSYSTEME	21
2.4.1	<i>Logikbasierter Ansatz</i>	<i>22</i>
2.4.2	<i>Repräsentationssprache.....</i>	<i>24</i>
2.4.3	<i>Agentenbasierter Ansatz</i>	<i>24</i>
2.5	ZUSAMMENFASSUNG	25
3	ALGEBRAISCHE MODELLE.....	27
3.1	MATHEMATISCHER HINTERGRUND	27
3.2	ALGEBRAISCHE MODELLE	29
3.2.1	<i>Teile einer Algebra</i>	<i>30</i>
3.2.2	<i>Kombination von Algebren</i>	<i>31</i>

3.2.3	<i>Teile Algebraischer Modelle</i>	31
3.2.4	<i>Funktionale Programmiersprachen</i>	32
3.2.5	<i>Verwendung von Haskell für algebraische Modelle</i>	33
3.2.5.1	Substitution in Programmiersprachen.....	33
3.2.5.2	Beschreibung von Algebren mit Haskell.....	34
3.3	OBJEKTORIENTIERTE MODELLIERUNG.....	35
3.4	UNTERSCHIEDE ZU OBJEKTORIENTIERTER PROGRAMMIERUNG.....	36
3.5	ZUSAMMENFASSUNG.....	36
4	GESETZESTEXTE UND ALGEBRAISCHE MODELLE – EIN VERGLEICH.....	39
4.1	ÄHNLICHKEITEN.....	39
4.1.1	<i>Beschreibung eines Modells</i>	39
4.1.2	<i>Zerlegung des Modells in kleine Teile</i>	40
4.1.3	<i>Abstrakte Beschreibung von Situationen und Folgen</i>	41
4.1.4	<i>Axiomatische Definitionen und Beschreibungen</i>	41
4.1.5	<i>Verknüpfungen</i>	42
4.2	UNTERSCHIEDE.....	44
4.2.1	<i>Unschärfe Definitionen</i>	44
4.2.2	<i>Organisatorische Inhalte</i>	45
4.2.2.1	Zeitlicher Aspekt.....	46
4.2.2.2	Umsetzung.....	48
4.2.3	<i>Möglichkeit von Widersprüchen</i>	48
4.3	ZUSAMMENFASSUNG.....	49
5	WIE FORMALISIERT MAN GESETZE?.....	51
5.1	ERSTE SCHRITTE.....	51
5.2	BEITRAG DER EINZELNEN PARAGRAPHEN.....	52
5.2.1	<i>Typdefinitionen</i>	52
5.2.2	<i>Operationen</i>	53
5.2.2.1	Angabe von notwendigen Operationen.....	53
5.2.2.2	Zuordnung der Operationen zu Algebren.....	55
5.2.2.3	Festlegung der Parameter für Operationen.....	56
5.2.3	<i>Axiome</i>	57
5.3	BAUEN EINER MODELLREALISIERUNG FÜR TESTZWECKE.....	58

5.4	GRENZEN DER MODELLIERUNG	59
5.4.1	<i>Organisatorische Inhalte</i>	60
5.4.2	<i>Interaktion</i>	61
5.4.3	<i>Ermessensspielraum</i>	61
5.5	ZUSAMMENFASSUNG	63
6	FORMALISIERUNG DES ALLGEMEINEN GRUNDBUCHSGESETZES 1955	65
6.1	AUS ANDEREN GESETZEN ÜBERNOMMENE STRUKTUREN	65
6.1.1	<i>Datentypen</i>	65
6.1.2	<i>Klassen und Operationen samt Axiomen</i>	68
6.1.2.1	Rechte	69
6.1.2.2	Dokumente	72
6.1.2.3	Datumsangaben.....	74
6.1.3	<i>Instanziierung der Klassen</i>	75
6.2	OBJEKTSTRUKTUR DES GBG55	77
6.3	EINTRAGUNGEN	81
6.3.1	<i>Generelle Vorgangsweise</i>	84
6.3.2	<i>Einverleibung</i>	87
6.3.3	<i>Vormerkung</i>	88
6.3.4	<i>Anmerkung</i>	90
6.4	ABFRAGEN	92
6.5	AUTOMATISCHE ABLÄUFE	93
6.5.1	<i>Löschen von Vormerkungen</i>	95
6.5.2	<i>Löschen von Rangordnungen</i>	97
6.5.3	<i>Löschen illegaler Eintragungen</i>	98
6.5.4	<i>Löschen gegenstandsloser Eintragungen</i>	98
6.5.5	<i>Löschen leerer Grundbuchkörper</i>	102
6.6	ZUSAMMENFASSUNG	103
7	ZUSAMMENFASSUNG UND AUSBLICK.....	104
7.1	ZUSAMMENFASSUNG	104
7.2	DISKUSSION DER ERGEBNISSE	105
7.3	AUSBLICK	106
	REFERENZEN	108

ANHANG A: SUCHERGEBNIS FÜR 'GRUNDSTÜCK' UND 'BETRETEN'

ANHANG B: HASKELL

ANHANG C: HASKELL PRELUDE

ANHANG D: GBG55

ANHANG E: MODELL DES GBG55

1 Einleitung

Gesetze bilden die Grundlage unserer Gesellschaft und beeinflussen nahezu jeden Bereich unseres Lebens. Sie definieren beispielsweise das Alter, ab dem man ein Kraftfahrzeug lenken darf, die Rechte eines Arbeitnehmers oder die Vorgangsweise beim Eigentumserwerb. Oft ist es im täglichen Leben entscheidend über diese Dinge Bescheid zu wissen. Beispielsweise kann es einen finanziellen Verlust verursachen, wenn man den Prozess des Erwerbs von Grundeigentum nicht kennt und man beim Kauf betrogen wird. Das Wissen um die gesetzlichen Bestimmungen wird dabei sogar vorausgesetzt (,Unwissenheit schützt vor Strafe nicht’).

Außerdem ist es für Gesetze wichtig, in sich konsistent zu sein und auch untereinander keine Widersprüche aufzuweisen, da solche Widersprüche oft schwerwiegende Probleme mit sich ziehen können. Die vorliegende Arbeit untersucht daher die Struktur von Gesetzen aus technischer Sicht, um eine Grundlage für zusätzliche Methoden zur Prüfung neuer Gesetze zu schaffen.

1.1 Motivation

1.1.1 Probleme mit Gesetzen

Rechtsordnungen entwickeln sich während langer Zeiträume mit wechselnden Regierungssystemen und laufend neuen technischen Errungenschaften. Österreich beispielsweise hatte im 20. Jahrhundert drei unterschiedliche Regierungsformen, nämlich Monarchie, Diktatur und Demokratie. Ähnliche Veränderungen fanden auch in anderen Staaten statt. Jede Regierungsform hinterlässt Spuren in der Rechtsordnung, da sie mit anderen Gesetzesvorstellungen arbeitet und diese einsetzt. Neu entwickelte Technologien und neue soziale Probleme benötigen ebenfalls neue Regelungen. Die Eisenbahn benötigt beispielsweise unter anderem ein Beförderungsgesetz und das Eisenbahnteilnehmungsgesetz. Ein weiteres Beispiel ist das Grundbuchsstellungsgesetz, das für die Umstellung des Grundbuchs vom analogen auf das digitale System benötigte wurde. Daher wächst die Menge der existierenden Gesetze ständig.

Alte Gesetze werden oftmals nicht explizit außer Kraft gesetzt. Gesetze können in zwei Fällen nicht mehr gebraucht werden. Falls die Ausgangssituation nicht mehr entstehen kann, gehören die Regelungen zum ,toten Recht’. So gehören Regelungen für Leibeigenschaft zum toten Recht, da es Leibeigenschaft in Österreich nicht mehr gibt. Weiters können Gesetze durch andere Gesetze ersetzt werden ohne dass dieser Umstand explizit ausgedrückt wird

(Derogation). Derogation bezeichnet die Lösung von Situationen in denen es mehrere, voneinander abweichende Regelungen gibt.

Gesetze können sehr strikte oder auch sehr breite Definitionen beinhalten. Ein typisches Beispiel für eine breite Definition ist der subjektive Eigentumsbegriff: (Dittrich and Tades 1997)

„Als ein Recht betrachtet, ist Eigentum das Befugnis, mit der Substanz und der Nutzung einer Sache nach Willkür zu schalten, und jeden anderen davon auszuschließen.“ [§ 354 ABGB¹]

Diese Gesetzesstelle definiert die Bedeutung eines Begriffes (‚Eigentum‘) durch Angabe der rechtlichen Konsequenzen. Eigentum hat die rechtlichen Konsequenzen, dass der Eigentümer mit der Sache tun kann was er will ohne jemand fragen zu müssen. Dieses Recht beinhaltet auch das Recht, mit einer Sache nichts zu tun oder sie zu zerstören. Andererseits gibt es jedoch auch viele Gesetzesstellen, die diese breite Definition einschränken. Ein einfaches Beispiel ist das Eigentum an einer Waffe. Die Verwendung von Waffen gegen Menschen ist im allgemeinen verboten. In manchen Situationen, wie zum Beispiel der Selbstverteidigung ist die Verwendung einer Waffe jedoch erlaubt. Diese Fälle stehen in strikteren Regelungen und sind üblicherweise Ausnahmen der allgemeinen Regel.

Nicht nur einzelne Regelungen, sondern auch Gesetze können umfassend oder strikt sein. Einige Gesetze, wie zum Beispiel das ABGB, sind umfassend in ihren Ausdrücken. Das ABGB definiert nur die Prinzipien der bürgerlichen Rechte und Pflichten und benötigt zur Umsetzung oft strikere Gesetze. Eines dieser Gesetze ist das Allgemeine Grundbuchsgesetz GBG55, (Dittrich, Angst et al. 1979), das eine strikte Definition der Prozesse für eine spezielle Situation (Nachweis von Eigentums- und Belastungsverhältnissen an Grundstücken) beinhaltet.

Diese Komplexität der Rechtsordnung schafft drei große Probleme:

- Der Staatsbürger kann eine Situation juristisch nur schlecht einschätzen, da er im allgemeinen die notwendigen Regelungen nicht oder nicht exakt genug kennt. Der durchschnittliche Staatsbürger kennt nur manche Grundprinzipien und die wichtigsten Regeln, die er für das tägliche Leben braucht. Es besteht daher die Gefahr ungewöhnliche Situationen falsch zu beurteilen und ungewollt Gesetzesübertretungen zu begehen. Die Folge einer solche Fehlbeurteilung kann die erzwungene Rücknahme einer Aktion sein.

¹ Allgemeines Bürgerliches Gesetzbuch

Solche Rücknahmen sind meist auch mit finanziellen Einbußen verbunden. Eine zu große Anzahl von rechtlichen Schwierigkeiten kann Auswirkungen auf die wirtschaftliche Entwicklung haben, da Entscheidungen langsamer und damit später getroffen werden. Allgemein gilt, dass komplexe Regelwerke die Effizienz von Transaktionen reduzieren und somit die Transaktionskosten erhöhen (North 1997).

- Häufig hat sogar der Gesetzgeber Schwierigkeiten mit der Einschätzung von rechtlichen Situationen oder den Auswirkungen auf andere Bereiche. Fehler in der Gestaltung von Gesetzen erfordern Novellierungen der Gesetze. Die Gefahr solcher Situationen steigt mit wachsender Komplexität der Rechtsordnung. Niemand kann alle relevanten gesetzlichen Regelungen für ein größeres Gebiet kennen. Daher werden neue Regelungen nicht vollständig an die bestehende Situation angepasst sein. Hier können prinzipiell zwei Fehler auftreten: Entweder ist eine spezielle Situation nicht geregelt oder zwei Regeln widersprechen sich.

Im Fall einer nicht geregelten Situation muss nach anderen Regeln (die eigentlich für andere Situationen geschaffen wurden) entschieden oder das Gesetz novelliert werden. Bei widersprüchlichen Regeln muss ebenfalls eine Novelle des Gesetzes erfolgen. In beiden Fällen entsteht zusätzlicher Aufwand, der bei einem fehlerfreien Gesetz nicht notwendig gewesen wäre.

- Internationale Regelungen verursachen ebenfalls Probleme. Die in der internationalen Regelung verwendeten Begriffe müssen in allen Unterzeichnerstaaten dieselbe Auswirkung haben. Wenn beispielsweise eine Abmachung über Grundeigentum im Ausland getroffen werden soll, müssen bestimmte Begriffe in allen Unterzeichnerstaaten gleich definiert sein. Einige diese Begriffe könnten sein: Grund, Eigentum, ausländischer Staatsbürger. Zusätzlich müssen aber auch die Prozesse, die mit diesen Begriffen verbunden sind, gleiche rechtliche Auswirkung haben. Beides kann kaum in vollem Umfang gegeben sein. Deshalb muss die Abmachung besonders vorsichtig verfasst sein und alle relevanten Begriff eindeutig definiert werden. Andernfalls könnte die Abmachung in verschiedenen Ländern unterschiedliche rechtliche Bedeutung haben. Das würde dem Sinn einer internationalen Abmachung grundlegend widersprechen.

Ein Vergleich unterschiedlicher Gesetzeslagen ist jedoch aus zweierlei Gründen schwierig. Es gilt nicht nur sprachliche Barrieren zu überwinden, sondern es können auch Unterschiede in der Entwicklung der Rechtsordnung eine Rolle spielen. Politische Aspekte nehmen hier ebenso einen besonderen Stellenwert ein. Beispielsweise ver-

meiden manche Nachfolgestaaten der ehemaligen UdSSR die Verwendung des Begriffes ‚Grundeigentum‘. Der Begriff wird durch ‚Nutzungsrecht‘ ersetzt, entspricht aber in weiten Teilen der Bedeutung des österreichischen Grundeigentums, da dieses Recht verkauft, vererbt und belehnt werden kann. Somit kann eine einfache Übersetzung des Gesetzestextes zu falschen Ergebnissen führen.

1.1.2 Umsetzung von Gesetzen

In unserer technisierten Welt setzen oft Computerprogramme Gesetze um. Die Umsetzung einer Richtlinie per Computerprogramm spart Zeit für Entscheidungen die schematisch getroffen werden, da Computer Standardaufgaben wie das Berechnen von Steuersätzen wesentlich schneller lösen können als Menschen. Nur bei Spezialfällen, die in kein Schema passen ist in der Folge menschlicher Eingriff notwendig.

Der Computer benötigt zur Durchführung seiner Arbeit Programme, die sich an rechtliche Vorgaben halten müssen. Ein Beispiel dafür ist die Bestimmung der Höhe der Steuer für ein Grundstück. Wenn also das Programm mit falschen Steuersätzen arbeitet wird automatisch ein falsches Ergebnis bestimmt. Dasselbe gilt auch für Zinsberechnung in Banken. Die vorgeschriebenen Berechnungsmethoden müssen genau eingehalten werden, da sonst kostspielige Schadenersatzforderungen von Kunden erfolgen könnten. Daher müssen Arbeitsweise des Programms und Gesetzeslage übereinstimmen.

Das Allgemeine Grundbuchsgesetz (samt seiner Nebengesetze) wird ebenfalls mit Hilfe von Computerprogrammen realisiert. Das Grundbuch existiert seit einigen Jahren als Datenbank und das ‚echte‘ Buch ist somit überholt. Die Methoden mit denen man Daten hinzufügen, ändern, löschen und abfragen kann sind von der Datenbank-Software und der Benutzerschnittstelle vorgegeben. Diese Methoden müssen den rechtlichen Bestimmungen entsprechen. Obwohl nur das Abfragen von Daten gänzlich ohne menschliches Zutun funktioniert, ist der Systemvorteil klar: Eigentums- und Belastungsinformationen sind innerhalb von Sekunden abrufbar, da eine Abfrage über Internet möglich ist. Die Kosten für eine Veräußerung oder Belehnung eines Grundstückes sinken, da zeitintensive Gerichtswege wegfallen.

Der Beweis, dass ein Programm genau so arbeitet wie es das Gesetz verlangt, ist jedoch schwierig. Die Beweisführung besteht aus zwei Schritten. Zuerst muss die Vollständigkeit des Programms gezeigt werden. Das Programm darf nur plausible Eingangparameter akzeptieren und muss eine fehlerhafte Bedienung bedingungslos ausschließen. Es muss also für jede mögliche Situation eine vordefinierte Reaktion des Programms geben. Ein zweiter Schritt muss

zeigen, dass Programm und Gesetz übereinstimmen. Dieser Punkt ist schwieriger zu erfüllen, da hier eine Vorgangsweise (das Programm) mit einer Idee (dem Gesetz) verglichen wird.

1.1.3 Vorteile einer Formalisierung

Die formale Beschreibung von Systemen bietet Möglichkeiten, die bei nicht-formalen Beschreibungen fehlen. Formale Beschreibungen müssen vorgegebenen Regeln gehorchen. Die Einhaltung dieser Regeln ist überprüfbar. Zusätzlich ist die Vollständigkeit einer Beschreibung ebenso prüfbar, wenn eine geeignete formale Methode für die Beschreibung verwendet wird. Ein möglicher Formalismus ist die Algebra. Die Formalisierung von Gesetzen bietet dadurch mehrere Vorteile:

- *Formale Analyse eines Gesetzes:* Während der Formalisierung werden Widersprüche oder Lücken innerhalb eines Gesetzes oder zwischen verschiedenen Gesetzen festgestellt. Außerdem entstehen genaue Definitionen der vorkommenden Fachbegriffe durch die Beschreibung ihrer Wirkung. Dadurch werden die Begriffe von ihrer Bedeutung getrennt, was beispielsweise länderübergreifende Vergleiche ermöglicht.
- *Direkte Übertragung der Änderungen von Gesetzen in die formale Beschreibung:* In einer Gesetzesnovelle geänderte Regeln können direkt auch in der formalen Beschreibung geändert und sofort getestet werden. Man kann die Vorgangsweise aber auch umkehren und zunächst die Regeln im formalen Modell ändern beziehungsweise testen und erst dann in einer Novelle umsetzen.
- *Prüfung der Übersetzung:* Formale Beschreibungen basieren auf Regeln. Die Einhaltung dieser Regeln ist überprüfbar.
- *Aufdecken des Bezuges zu anderen Gesetzen:* Manchmal sind bei der formalen Beschreibung Definitionen notwendig, die im behandelten Gesetz fehlen, da sie von anderen Gesetzen übernommen wurden. Solche Definitionen sind mitunter explizit ausgewiesen¹, und werden des Öfteren erst im Verlauf der formalen Beschreibung ersichtlich².

¹ §9 BGB55 verweist auf Definitionen aus dem ABGB: „Im Grundbuch können nur dingliche Rechte und Lasten, ferner das Wiederkaufs- und Vorkaufsrecht (§§ 1070 und 1073 ABGB) sowie das Bestandrecht (§1095 ABGB) eingetragen werden.“

² Welche dinglichen Rechte und Lasten gibt es?

- *Aufdecken der Struktur des Gesetzes:* Die Zusammenstellung der Gesetze mit Hilfe einer Formalisierung fasst zusammengehörige Teile zusammen und ermöglicht somit einen einfacheren Überblick. Zusätzlich treten die Unterschiede zwischen generellen Regeln und Spezialfällen offen zu Tage, da die Regeln für Spezialfälle die allgemeinen Regeln überschreiben müssen.

Eine spezielle Art von formaler Beschreibung ist ein Computerprogramm. Selbiges besteht prinzipiell aus drei Teilen: Eingabe, Bearbeitung und Ergebnis. Jedes Programm lässt nur eine bestimmte Menge von Eingaben zu, alle übrigen werden ignoriert. Die akzeptierten Eingaben werden einem oder mehreren Bearbeitungsschritten unterworfen und liefern ein Ergebnis. Unterschiedliche Eingaben können dabei unterschiedliche Bearbeitungsmethoden zur Folge haben. Eine Formalisierung definiert die möglichen Bearbeitungsschritte als Methoden und beschreibt diese Methoden in einer einfachen Sprache. Die Parameter der Methoden stellen dabei die Eingabe dar. Ein Beispiel für eine solche Sprache ist die Mathematik. Die Einfachheit der Mathematik ermöglicht die Übersetzung des Modells in eine Programmiersprache und das Ergebnis kann als Grundgerüst eines Computerprogramms verwendet werden. Es fehlen dann nur noch Teile wie etwa die verbesserte Interaktion mit dem User (beispielsweise über Ein- und Ausgabemasken).

Während der Erstellung des formalen Modells fallen Widersprüche auf. Situationen, für die es im Gesetz widersprüchliche Regelungen gibt, sind nicht modellierbar. Formale Modelle schreiben für die jeweilige Situation eine eindeutige Reaktion vor. Widersprüchliche Vorschriften erfüllen diese Voraussetzung nicht. Ebenso fallen Abweichungen von einer generellen Vorschrift auf. Das ist meist erwünscht, kann aber auch irrtümlich entstanden sein. Beispiele für erwünschte Abweichungen sind Enteignung und Vererben von Grundstücken. Jede Eintragung ins Grundbuch (also auch die Eigentumsübertragung) erfordert die Zustimmung des Eigentümers. In beiden angegebenen Fällen ist diese Zustimmung nicht zu erlangen. Im Fall der Enteignung findet die Eigentumsübertragung gegen den Willen des Eigentümers statt, im Falle des Vererbens ist der Eigentümer bereits tot. In beiden Fällen ist die Abweichung notwendig, ja sogar erwünscht. Während des Modellierens fallen beide Arten von Widersprüchen auf und es muss bei jedem Widerspruch geprüft werden, ob er erwünscht ist oder nicht.

Beim Vergleich rechtlicher Situationen in unterschiedlichen Ländern muss man auf eine einheitliche Sprache zurückgreifen. Eine Übersetzung eines Gesetzestextes in eine andere Sprache ist schwierig, da die verwendeten Begriffe nur im Kontext des Gesetzes gültig sind. Wald bedeutet im forstrechtlichen Sinn beispielsweise, dass der Boden forstwirtschaftlich ge-

nutzt wird. Dazu sind allerdings keine Bäume nötig, was der umgangssprachlichen Bedeutung des Wortes widerspricht. Solche Probleme gibt es üblicherweise in Gesetzen aller Länder. Daher sollte man keine natürliche Sprache für den Vergleich verwenden. Es bietet sich daher an, eine formale Sprache zu verwenden. In einer solchen Sprache kann man die jeweils gültigen Definitionen ausdrücken und hat somit eine Vergleichsmöglichkeit. Das eröffnet auch die Möglichkeit, die rechtlichen Situationen in ihrer Gesamtheit zu erfassen und zu vergleichen.

1.2 Hypothese und Gliederung der Arbeit

Der Ausgangspunkt der Arbeit ist die Annahme, dass Gesetze ein Modell beschreiben. Gesetze benötigen für diese Beschreibung eine Struktur. Mit Hilfe der Mathematik kann man ebenfalls Modelle strukturiert beschreiben. Eine Art mathematischer Beschreibung ist beispielsweise die Algebra (Gutttag and Horning 1978; Horebeek and Lewi 1989; Breu 1991; Lin 1998). Bei näherer Betrachtung fällt auf, dass die Art der Beschreibung von Gesetzen und Algebren Ähnlichkeiten aufweist ist. Die Hypothese der vorliegenden Arbeit lautet daher:

Es ist möglich, die Struktur von Gesetzen mit algebraischen Modellen unter Beibehaltung der Paragraphenstruktur zu beschreiben.

Der erste Schritt ist eine Diskussion von Gesetzen und algebraische Modellen. Es gibt mehrere Methoden, Gesetze für einen Computer lesbar zu machen. Kapitel 2 beschreibt die wichtigsten Methoden und zeigt ihre Nachteile bei der Anwendung für den Vergleich verschiedener Rechtsordnungen. Kapitel 3 beschreibt dann algebraische Modelle und ihre Entwicklung. Das Kapitel schließt mit der Vorstellung einer Programmiersprache, mit deren Hilfe man algebraische Modelle erstellen und überprüfen kann.

Der nächste Schritt ist ein Vergleich von Gesetzen und algebraischen Modellen. Es gibt zwischen diesen beiden sowohl Ähnlichkeiten als auch Unterschiede. In Kapitel 4 werden zunächst die Ähnlichkeiten untersucht und dann die Unterschiede zwischen den Repräsentationen aufgezeigt.

Um vom Gesetzestext zum algebraischen Modell zu gelangen, ist eine Umformung unerlässlich. Diese wiederum erfordert eine schrittweise Formalisierung des Gesetzestextes. Gesetzestexte sind in Paragraphen organisiert. Die Durchführung der Formalisierung benötigt somit vorab eine Diskussion der Beiträge, welche ein einzelner Paragraph zum Modell leisten kann. Erst danach kann eine Beschreibung des Umformungsprozesses stattfinden. Kapitel 5 erläutert den Prozess und schließt mit einer Diskussion der Grenzen der Formalisierung.

Das folgende Kapitel zeigt an Hand einiger Beispiele wie die Formalisierung durchgeführt wird. Die Beispiele sollen zum besseren Verständnis der Methode beitragen und zeigen, dass die Methode anwendbar ist. Die komplette Formalisierung befindet sich im Anhang.

Die Arbeit schließt mit einer Zusammenfassung und einer Beschreibung von zukünftigen Forschungsfragen. Die entwickelte Methode ist noch nicht perfekt und benötigt noch Feinschliff. Die Absicht dieser Arbeit ist es jedoch nicht, eine perfekte Methode zu präsentieren, sondern es geht vielmehr darum, aufzuzeigen, dass es möglich ist, Gesetze in Algebren umzuwandeln. Daher werden nur Lösungen für die wichtigsten Probleme gezeigt. Die Entwicklung besserer Lösungen sollte ebenso Teil der zukünftigen Forschungsarbeit sein, wie die Lösung selten auftretender Probleme.

1.3 Abgrenzung der Arbeit

In der Arbeit sind einige Bereiche bewusst ausgeklammert. Dabei handelt es sich vor allem um Prüfung der Sinnhaftigkeit des Gesetzes und die Abbildung der Gesetzesauswirkungen in der Realität. Die Arbeit enthält keine Analyse des Grundbuchgesetzes. Es wird somit nichts darüber ausgesagt, ob bestimmte Regelungen sinnvoll sind oder nicht. Nicht abgehandelt wird auch, wie das Grundbuch in Österreich funktioniert oder nach welchen Prinzipien es entwickelt wurde. Die Bearbeitung solcher Fragestellungen ist Juristen vorbehalten. Ebenfalls ausgeklammert wurden Aspekte der Auswirkungen in der Realität, da diese in der Dissertation von Steffen Bittner bearbeitet wurden (Bittner 2001). Daraus folgend sind auch Interaktionen zwischen dem Grundbuch und seinen Benutzern nicht Teil der Arbeit.

1.4 Das Beispiel

Als Beispiel wird das Österreichische Allgemeine Grundbuchgesetz behandelt. Dieses Gesetz wurde aus mehreren Gründen gewählt:

- *Altes und ausgereiftes Gesetz:* Das allgemeine Grundbuchgesetz stammt in seiner heutigen Form aus dem Jahre 1955, ist aber von der Konzeption her wesentlich älter. Daher sind alle inhaltlichen Probleme bereits bereinigt und somit sollte einer formalen Beschreibung nichts im Wege stehen.
- *Sehr striktes Gesetz:* Das Grundbuch behandelt Rechte an Boden. Das Eigentumsrecht ist in der österreichischen Rechtsordnung sehr stark verankert. Daher muss das allgemeine Grundbuchgesetz sehr strikt sein, um das Recht garantieren zu können. So

entsteht eine sehr enge Beschreibung des Systems ‚Grundbuch‘, was wiederum die formale Beschreibung vereinfacht.

- *Wenige Voraussetzungen:* Das allgemeine Grundbuchsgesetz basiert auf wenigen anderen Gesetzen. Es ist das Grundgesetz über die Sicherung von Rechten an Boden und basiert hauptsächlich auf dem allgemeinen bürgerlichen Gesetzbuch und der Geschäftsordnung für Gerichte.

2 Gesetze und Computer

Dieses Kapitel beschreibt Rechtsordnungen und die Möglichkeiten des Computereinsatzes in Rechtsfragen. Rechtsordnungen sind komplexe Strukturen aus Vorschriften unterschiedlicher Gewichtung, die über lange Zeiträume entstanden sind. Das mitteleuropäische System beispielsweise zeigt starke römische und germanische Einflüsse. Die Komplexität der Rechtsordnungen erfordert den Einsatz von Hilfsmitteln, um den Überblick bei Rechtsprechung und Schaffung neuer Gesetze zu behalten. Computer sollten dabei helfen. Die einfachste Möglichkeit ist es, die Gesetzestexte einfach in Textform im Computer zu speichern. Abschnitt 2.2 zeigt ein solches System, enthält eine Diskussion der Grenzen des Systems und zeigt Möglichkeiten zur Verbesserung auf. Komplexere Ansätze versuchen, mit Hilfe von Expertensystemen, Rechtsprechung zu unterstützen und nachzuvollziehen. Abschnitt 2.4 zeigt einige dieser Ansätze und beschreibt ihre Vor- und Nachteile.

2.1 Aufbau einer Rechtsordnung

Zentrales Anliegen hier ist zunächst eine Definition des Begriffes ‚Rechtsordnung‘, da dieser Begriff bisher ohne Definition verwendet wurde. Abschnitt 2.1.1 enthält diese Definition. Danach ist eine Diskussion des Aufbaues und der Entstehung einer Rechtsordnung notwendig. Schließlich gibt es auch Methoden um Widersprüche in Rechtsordnungen aufzulösen.

2.1.1 Rechtsordnung

Heinz Krejci definiert die Rechtsordnung folgendermaßen (Krejci 1995, S. 1):

Rechtsordnung heißt die Summe jener Normen, die das Zusammenleben der Menschen regeln und mit staatlicher Zwangsgewalt durchsetzbar sind.

In dieser Definition gibt es einige Begriffe, die einer Erläuterung bedürfen:

- *Normen* sind Verhaltensanordnungen die definieren, was man tun soll bzw. nicht tun darf (Verbote und Gebote).
- Unter *Zusammenleben der Menschen* versteht man das äußere menschliche Verhalten in der Gemeinschaft. Normen sind daher nicht an Tiere gerichtet, obwohl sie auch von Tieren handeln können (z.B. Beißkorbpflicht in öffentlichen Verkehrsmitteln).
- Die *staatliche Zwangsgewalt* soll das Recht der einzelnen Bürger durchsetzen. Die Rechtsordnung soll den Frieden zwischen den Menschen sichern (*Friedensfunktion*) und

daher beansprucht der Staat das *Gewaltmonopol*. Somit wird die Anwendung das ‚Recht des Stärkeren‘ vermieden und der Staat garantiert die Rechte der Schwächeren.

Außer den Rechtsnormen gibt es noch weitere Normen, die aus anderen Quellen stammen. Moral, Sitte und Religion definieren ebenfalls Normen. Diese Normen werden allerdings nicht mit staatlicher Gewalt durchgesetzt. Sitten sind von Menschen laufend ausgeübte Handlungsweisen, in denen jedoch keine Rechtsausübung gesehen wird. Unter Moral versteht man die wertende Beurteilung von Handlungen, also die Billigung oder Missbilligung. Schließlich definiert auch die Religion Normen (Krejci 1995).

Manchmal widersprechen sich Normen aus unterschiedlichen Quellen. Ein Beispiel dafür wäre die Abtreibung. Die religiöse Norm für römisch-katholische Christen in der gegenwärtigen Interpretation besagt, dass es sich dabei um Mord handelt und somit verboten ist. Die gesetzliche Norm definiert eine Frist, innerhalb der Abtreibung erlaubt ist. Solange man sich also innerhalb der Frist bewegt, in der das Gesetz die Abtreibung erlaubt, handelt man gegen die religiöse Norm, nicht jedoch gegen die Rechtsnorm.

Ein weiterer Unterschied zwischen Rechtsnormen und den übrigen Normen ist, dass Rechtsnormen niedergeschrieben werden. Rechtsnormen müssen für alle Staatsbürger zugänglich sein, da die Staatsbürger sie befolgen müssen. Somit muss gewährleistet sein, dass die Normen auch von allen Staatsbürgern gleich verstanden werden. Das kann nur dann geschehen, wenn Normen in eindeutiger Weise dokumentiert sind.

Üblicherweise werden Rechtsnormen, die einen bestimmten Lebensbereich behandeln zu einem größeren Schriftstück, dem Gesetz zusammengefasst. Diese Vorgangsweise hat den Vorteil, dass man bei der Suche nach Regeln schneller zum gewünschten Ergebnis gelangt, da man sich auf das entsprechende Regelwerk konzentrieren kann. Will man beispielsweise die Definition von Eigentum finden, schlägt man im Allgemeinen Bürgerlichen Gesetzbuch nach. Außerdem ermöglicht diese Zusammenfassung eine einfache Referenzierung der Rechtsnorm. Die Referenzierung erfolgt über den Namen des Gesetzes (oder dessen Abkürzung) und die Nummer der Norm innerhalb des Gesetzes (des Paragraphen). Sie ist eindeutig, kurz und unabhängig von einer bestimmten Druckausgabe oder einer Seitenzahl.

2.1.2 Rechtsquellen

Rechtsquellen sind Phänomene, aus denen Recht entsteht bzw. die Recht erkenntlich machen. Somit spricht man auch von Entstehungsquellen und Erkenntnisquellen. Entstehungsquellen

sind Orte, an denen in vorgegebenen Prozessen Rechtsvorschriften geschaffen werden (entstehen). Diese Rechtsvorschriften können Gesetze, Verordnungen, Verträge, aber auch Gerichtsurteile sein. Erkennbar werden sie aber erst durch die Niederschrift und Veröffentlichung. Die Erkenntnisquellen sind somit die Bundes- und Landesgesetzblätter, Vertragsurkunden und ähnliches.

Man unterscheidet außerdem zwischen generellen und individuellen Rechtsquellen. Generelle Rechtsquellen richten sich an eine bestimmte Gruppe von Menschen. Das können alle Menschen sein (die Allgemeinheit), bestimmte Berufsgruppen (Ingenieurkonsulenten für Vermessungswesen, Beamte, Arbeitnehmer) oder auch Gruppen die nach anderen Merkmalen bestimmt wurden (Wahlberechtigte, Schwangere, Entmündigte, Schulpflichtige). In diese Kategorie fallen beispielsweise Gesetze und Verordnungen, also die Rechtsnormen. Individuelle Rechtsquellen richten sich an ganz bestimmte Personen, die auch individuell bezeichnet werden. Das sind vor allem Gerichtsurteile, Bescheide und Verträge.

2.1.3 Aufbau und Arten von Rechtsnormen

Die einzelne Rechtsnorm ist der kleinste Baustein der Rechtsordnung (Lachmayer 1977). Rechtsnormen definieren rechtliche Konsequenzen. Sie definieren also zunächst abstrakte Verhaltensweisen oder Situationen (den Tatbestand), in denen die Norm angewendet werden muss. Danach definieren sie rechtliche Konsequenzen (die Rechtsfolge). Eine Norm besteht somit aus einer Bedingung und einer Vorschrift.

Es gibt unterschiedliche Arten der Einteilung von Rechtsnormen. Einerseits spricht man von materiellem und formellem Recht, andererseits unterscheidet man zwischen zwingendem und dispositivem Recht. Materielles Recht definiert die Ordnung, nach welcher die Menschen zusammenleben. Formelles Recht regelt das Verfahren der Rechtsdurchsetzung vor den dafür zuständigen staatlichen Stellen. Zwingendes Recht ist Recht auf das nicht verzichtet werden kann, auch wenn die Parteien etwas anderes vereinbaren. Im Gegensatz dazu lässt dispositives Recht eine abweichende Regelung zu.

2.1.4 Gliederung der Rechtsordnung

Es gibt einen Stufenbau der Rechtsordnung (Kühne 1985). Es handelt sich dabei um ein rechtslogisches Modell zur Einordnung von Rechtsnormen in ein bestehendes Normensystem: Eine Rechtsnorm gilt, wenn sie auf eine bestehende, ‚höhere‘ Rechtsnorm rückführbar ist. Die Abfolge dieser Rechtsnormen nennt man Stufenbau.

Jeder Stufe der Rechtsordnung kann eine Rechtsquelle zugeordnet werden. Daraus ergibt sich folgende Rangordnung der Rechtsquellen:

- Verfassungsgesetze
- Einfache Gesetze
- Verordnungen
- Urteile – Bescheide – Verträge

Die einzelnen Rechtsquellen müssen in der nächsthöheren Stufe verankert sein. Somit müssen einfache Gesetze in der Verfassung ihre Deckung finden, sich Verordnungen nach einfachen Gesetzen richten usw. Je niedriger dabei die Stufe einer Rechtsnorm ist, desto konkreter ist ihre Aussage.

Verfassungsgesetze und einfache Gesetze entstehen in drei Schritten. Zunächst wird ein Gesetzesvorschlag erarbeitet. Das kann von der Regierung, von einzelnen Abgeordneten, von 100.000 Stimmberechtigten oder vom Bundesrat erstattet werden. Dieser Vorschlag wird im Nationalrat zwei Mal mit jeweils anschließender Debatte verlesen. Wurde der Vorschlag bis dahin nicht verworfen, erfolgt die Abstimmung. Dafür sind folgende Quoren (Anzahl der anwesenden bzw. zustimmenden Abgeordneten) zu erreichen:

	Verfassungsgesetz	Einfaches Gesetz
Präsenzquore	$\frac{1}{2}$	$\frac{1}{3}$
Konsensquore	$\frac{2}{3}$	$\frac{1}{2} + 1$ Stimme

Sind die erforderlichen Quoren erreicht, so liegt ein Gesetzesbeschluss des Nationalrates vor. Dieser wird dem Bundesrat vorgelegt, der dann acht Wochen Zeit für einen Einspruch hat. Der Bundespräsident beurkundet das verfassungsmäßige Zustandekommen und der Bundeskanzler veröffentlicht das Gesetz dann im Bundesgesetzblatt.

Verordnungen werden von Behörden erlassen und richten sich gegen jedermann. Verordnungen dienen dazu, die allgemeinen Regelungen der Gesetze in durchführbare Abläufe zu übertragen und veränderliche Parameter festzulegen. Urteile, Bescheide und Verträge richten sich gegen individuelle Personen. Die Unterschiede liegen bei den beteiligten Personen. Ein Urteil wird von einem Gericht ausgestellt, ein Bescheid von einer Behörde. Beide richten sich

an explizit genannte Personen. Der Vertrag hingegen wird zwischen Privatpersonen abgeschlossen.

2.1.5 Inkonsistenzen zwischen Rechtsnormen

Es kann vorkommen, dass Rechtsnormen zueinander im Widerspruch stehen. Es gibt also Regeln, die in einem bestimmten Fall anzuwenden sind, die sich aber widersprechen. In diesem Fall spricht man von Inkonsistenz. Dabei gibt es zwei unterschiedliche Ausprägungen:

- Scheinbare Inkonsistenzen treten bei unterschiedlichen Geltungsbereichen auf und können aufgelöst werden. Die Regeln behandeln also nicht die gleiche Menge von möglichen Fällen, sondern nur ähnliche Mengen. Diese Inkonsistenzen sind meist sogar erwünscht.

Die Geltungsbereiche können sich in verschiedener Hinsicht unterscheiden. Einerseits können Unterschiede hinsichtlich des persönlichen oder sachlichen Geltungsbereiches auftreten, andererseits kann es sich um verschiedene zeitliche Geltungsbereiche handeln. Die zum Auflösen der Inkonsistenz angewendete Methode heißt ‚Derogation‘ (Krejci 1995). Es gibt vier mögliche Vorgangsweisen:

- Die ältere Regel ist gültig
- Die jüngere Regel ist gültig
- Die allgemeinere Regel ist gültig
- Die speziellere Regel ist gültig

Üblicherweise funktioniert die Derogation. Problematisch ist jedoch, dass es keine Regel darüber gibt, welche der vier Vorgangsweisen wann anzuwenden ist. Diese Entscheidung muss für jeden aufgefundenen Widerspruch neu getroffen werden. Der Prozess der Entscheidungsfindung dauert allerdings Zeit und verursacht somit Verzögerungen bei Entscheidungen. Da Verzögerungen meistens auch mit Kosten verbunden sind, sollte die Anzahl der Widersprüche möglichst gering gehalten werden.

- Echte Inkonsistenzen können nicht aufgelöst werden. Sie stellen ein Problem dar, das nur durch Aufhebung einer der Normen gelöst werden kann. Voraussetzung für einen echten Widerspruch ist, dass zeitlicher, persönlicher und sachlicher Geltungsbereich übereinstimmen. Es könnte beispielsweise vorkommen, dass ein Unternehmen die zu zahlenden Steuern nach zwei unterschiedlichen Regeln berechnen muss, welche zu

unterschiedlichen Beträgen führen. Wenn nun keine Möglichkeit besteht, über Derogation eine der Möglichkeiten zu eliminieren, so liegt eine echte Inkonsistenz vor.

2.2 Modellierung mit Gesetzen

Eine Rechtsordnung besteht aus einer Struktur von Gesetzen, die in Relation zueinander stehen und bestimmte Probleme abhandeln. Oft gibt es zu einem Problem mehrere Gesetze, die jeweils bestimmte Aspekte klären. Beispielsweise gibt es über das Grundbuch nicht nur ein Gesetz, sondern eine Anzahl von gesetzlichen Regelungen (Marent and Preisl 1994):

- Allgemeines Grundbuchslegungsgesetz: Definiert den inneren Aufbau eines Grundbuches und das Verfahren zur Anlegung von Grundbüchern
- Allgemeines Grundbuchsgesetz: Definiert die Aufgabenstellung und Arbeitsweise eines Grundbuches ohne detailliert auf die innere Struktur einzugehen
- Grundbuchsumstellungsgesetz: Enthält Bestimmungen zur Umstellung des Grundbuches von analoger auf digitale Führung
- Grundverkehrsgesetze der Bundesländer, Tiroler Höfegesetz: Definieren Richtlinien für den Verkehr von land- und forstwirtschaftlichen Grundstücken sowie den Grundstücksverkehr für Ausländer
- Liegenschaftsteilungsgesetz: Reguliert die Durchführung von Grundteilungen
- Vermessungsgesetz: Reguliert die Art und den Umfang der für den Kataster notwendigen Vermessungen
- Wohnungseigentumsgesetz: Definiert Wohnungseigentum und die damit verbundenen Rechte und Pflichten

Jedes dieser Gesetze unterteilt das Themengebiet dann noch in kleinere Bereiche. Das Allgemeine Grundbuchsgesetz enthält beispielsweise folgende Bereiche:

- Allgemeines: Definitionen, Begriffsbestimmungen und grundsätzliche Regeln
- Eintragungen
 - Allgemeines: Definition der Arten der Eintragung und der eintragbaren Elemente
 - Einverleibung: Voraussetzungen für die Einverleibung

- Vormerkung: Voraussetzungen für die Vormerkung und Durchführung der Rechtfertigung
- Anmerkung: Elemente die angemerkt werden dürfen
- Abschreibung: Definition der Abschreibung
- Verfahren in Grundbuchssachen: Beschreibung der unterschiedlichen Verfahren
 - ...
- Bereinigung und Berichtigung des Grundbuches: Vorgangsweise bei unzulässigen oder unrichtigen Eintragungen
 - ...
- Schlussbestimmungen: Inkrafttreten, Ersetzen anderer Regelungen, Verweise auf andere Gesetze, Vollziehungsbestimmungen

Jeder Bereich wird wiederum in kleine Teile gesplittet und in Paragraphen behandelt. Zusätzlich werden Aufzählungen zur weiteren Gliederung verwendet. Hierbei kann es mehrere Gliederungsebenen geben.

2.3 Speicherung von Gesetzestexten in Datenbanken

Die einfachste Methode für die Verwendung von Computern zur Verwaltung von Gesetzen ist die Speicherung in einer Datenbank. Das ermöglicht die Schlagwortsuche, also nach bestimmten Begriffen und Phrasen im Text. Ein Beispiel dafür ist das österreichische Rechts-Informationssystem (RIS). Die Datenbank beinhaltet alle Bundes- und Landesgesetze, internationale Verträge und Erkenntnisse von Verwaltungs- und Verfassungsgerichtshof. Die Daten können über Internet (<http://www.ris.bka.gv.at/>) abgefragt werden.

Das Beispiel zeigt die Verwendung von Datenbanken bei der Beurteilung anhand folgender Fragestellung: Ist es erlaubt, ein bestimmtes Grundstück zu betreten und wenn ja, in welchen Situationen? Bei dieser Fragestellung hat das Grundstück mehrere Attribute, die das Ergebnis der Frage stark beeinflussen. Jedes Grundstück hat einen Eigentümer. Falls der Eigentümer die fragende Person ist, so ist die Antwort ein uneingeschränktes ‚Ja‘. Dasselbe gilt für Grundstücke, die als öffentliches Gut gewidmet sind, auch wenn es hier bereits Einschränkungen gibt (eine Autobahn ist beispielsweise als öffentliches Gut gewidmet, darf aber nur in Ausnahmefällen ‚betreten‘ werden). In allen übrigen Fällen wird die Antwort prinzipiell ‚Nein‘ sein. Prinzipiell deshalb, weil bestimmte Personenkreise wie Polizisten oder Feuerwehrmänner

das Grundstück immer betreten dürfen, wenn es aus dienstlichen Gründen notwendig ist. Ausnahmen sind hier wieder Gebäude der UNO und diplomatische Missionen. Dieser kurze Diskussion zeigt bereits, dass zusätzliche Informationen notwendig sind, um die gegebene Frage zu beantworten. Es wäre aber denkbar, alle Gesetze anzugeben, die möglicherweise zur Wirkung kommen können um einen Überblick über die allgemeine gesetzliche Situation zu liefern.

2.3.1 Probleme dieser Lösung

Eine einfach Antwort auf die Frage kann folgendermaßen geliefert werden: Wenn die Gesetzestexte in einer Datenbank gespeichert sind, genügt eine einfache Textsuche um alle relevanten Textstellen zu finden. In der Praxis treten bei dieser Methode jedoch einige Probleme auf. Eine Abfrage mit den Suchworten ‚Grundstück‘ und ‚betreten‘ liefert beim RIS folgende Lösung:

Art der Rechtsstelle	# der Quellen	# der Fundstellen
Staatsverträge	4	4
Bundesgesetze	15	19
Bundesverordnungen	4	5
Landesgesetz und Landesverordnungen	51	55
Erkenntnisse und Beschlüsse des Verfassungsgerichtshofes	15	15
Erkenntnisse und Beschlüsse des Verwaltungsgerichtshofes	119	119
	208	217

Das Ergebnis der Suche sind also 217 Gesetzesstellen. Das Problem ist nun, dass diese Texte zwar beide Worte enthalten, es aber nicht notwendig ist, dass auch die Bedeutung der Textstelle zur Abfrage passt. Folgende (fiktive) Textstelle würde beispielsweise ebenfalls gefunden werden:

Der Gehsteig darf nicht betreten werden, wenn auf dem benachbarten Grundstück ein Baum steht.

Der Text hat nichts mit dem Betreten von Grundstücken zu tun, enthält aber die geforderten beiden Begriffe. Somit ist klar, dass viele der gefundenen Textstellen nichts mit der Fragestellung zu tun haben. Die Anzahl dieser Textstellen sollte minimal sein, da die Textstellen zwar überprüft werden müssen, aber nicht für die Antwort verwendet werden können.

Die Gesetzesstellen stammen aus sehr unterschiedlichen Rechtsquellen. Es sind einerseits Gesetze und Verordnungen (sowohl auf Bundes- als auch auf Landesebene), andererseits aber auch Entscheidungen der obersten Gerichtshöfe und sogar Staatsverträge gefunden worden. Die große Anzahl unterschiedlicher Quellen liegt teilweise am Suchverfahren, das ganz simpel nach den vorgegebenen Worten sucht. Andererseits gibt es sehr viele Sonderregelungen für bestimmte Grundstücke. Beispielsweise betreffen drei der vier Staatsverträge den jeweiligen Amtssitz von UNO-Kommissionen in Wien. Die davon betroffenen Grundstücke unterliegen Sonderregeln bezüglich des Betretens. Andere Gesetze beinhalten Regeln für bestimmte Berufsgruppen (z.B. Sicherheitspolizeigesetz, Vermessungsgesetz) oder Gebiete (z.B. Berggesetz, Forstgesetz, Gesetze bezüglich Naturschutzgebieten). Zusätzlich gibt es noch Erkenntnisse und Beschlüsse der obersten Gerichtshöfe, die eingetretene, bisher nicht eindeutig entschiedene Situationen regeln. Alle diese Rechtsnormen regeln das Betreten von Grundstücken. In einem speziellen Fall werden aber nur wenige davon zur Anwendung kommen.

Eine Möglichkeit, die von diesem Beispiel nicht in Betracht gezogen wurde, ist die Verwendung anderer Begriffe. Wenn man die Abfrage beispielsweise mit ‚Boden betreten‘ ausführt, erhält man viele neue Treffer, wie etwa die Durchführungsbestimmungen zu Abwehr und Tilgung von Tierseuchen, die Bauarbeiterschutzverordnung, die Fleischhygieneverordnung, die Schieß- und Sprengmittelmonopolsverordnung und andere mehr. Wird versucht, alle übrigen Möglichkeiten ebenso abzufragen, steht man vor dem Problem, dass nicht alle Möglichkeiten bekannt sind, da sie nicht explizit beschrieben sind. Beispielsweise könnte die Polizei ermächtigt sein, Orte zu untersuchen. Eine Voraussetzung dafür ist natürlich die Befugnis zum Betreten des Ortes, die jedoch nicht direkt im Gesetz stehen muss, da dieser Umstand von vornherein klar ist.

Diese Begriffe können auch automatisch generiert werden. Der Benutzer kann dabei nach den eingegebenen Suchbegriffen und nach Synonymen suchen. Synonyme von ‚betreten‘ sind beispielsweise die Begriffe ‚begehen‘, ‚hineingehen‘, ‚eindringen‘ und ‚einbrechen‘. Das Beispiel mit der Polizei, die einen Ort untersuchen darf, ist allerdings auch durch Synonyme nicht korrekt lösbar. Somit liefern Synonyme zwar eine bessere Lösung, sind aber kein Garant für ein vollständiges Ergebnis.

Der Grund für dieses Problem liegt in der unterschiedlichen Auffassung von Computer und Mensch. Der Mensch denkt in Begriffen und hat keine Probleme mit Synonymen oder unterschiedlichen Bedeutungen eines Wortes. Der Computer hingegen arbeitet bei seiner Suche mit Buchstabenfolgen. Er nimmt einfach eine vorgegebene Buchstabenfolge und prüft, ob diese Folge in einem Text vorkommt. Daher wird er weder andere Schreibweisen (Änderung der Rechtschreibung) noch andere Begriffe mit derselben Bedeutung finden, dafür aber Vorkommen des Wortes in anderen Zusammenhängen liefern.

Das Problem bei der einfachen Datenbankabfrage ist also die Treffergenauigkeit. Eine Datenbankabfrage liefert Kandidaten für die Antwort auf eine Frage. Der Computer kann bei der Suche nach zutreffenden Gesetzesstellen nur verwenden, was der Benutzer eingespeist hat. Da der Computer den Sinn des von ihm verwalteten Textes nicht versteht, kann er nur eine wörtliche Suche durchführen. Dabei werden inhaltlich gleiche aber wörtlich unterschiedliche Gesetzesstellen nicht gefunden. Somit tritt ein Fehler auf, der bei Datenbankabfragen nicht passieren sollte: Mögliche Kandidaten werden nicht gefunden.

2.3.2 Verbesserungen?

Der letzte Abschnitt hat gezeigt, dass eine ausschließliche Speicherung des Textes zwar für die Publikation von Rechtsnormen nützlich ist, bei der Beurteilung rechtlicher Situationen aber nicht wirklich hilft. Welche Möglichkeiten der Verbesserung gibt es also? Zunächst kann man Stichworte hinzufügen. Eine weitere Möglichkeit ist die automatische Generierung von Synonymen. Schließlich könnte man dem Anwender noch ein Wörterbuch zur Verfügung stellen. Alle drei Varianten sind beim Rechts-Informationssystem vorhanden.

Stichworte sind vorgegebene Suchbegriffe, die bestimmten Textstellen bei der Erfassung zugeordnet wurden. Die Anzahl der vorhandenen Stichworte ist eingeschränkt. Das ermöglicht die Angabe einer vollständigen Liste, aus welcher der Anwender die relevanten Begriffe herausucht. Die methodischen Probleme sind in erster Linie die Vollständigkeit der Stichwortliste und die Zuordnung zu den Gesetzesstellen. Sollten Stichworte fehlen, wird eine bestimmte Art der Fragestellung unmöglich, da das Vokabular für diese Fragen fehlt. Fehlt ein bestimmtes Stichwort bei einer Textstelle, so wird diese Stelle nicht gefunden.

Mit Hilfe eines Wörterbuches kann man Begriffsdefinitionen verfeinern. Bei der Eingabe des Begriffes ‚Boden‘ könnten beispielsweise folgende Begriffe vorgeschlagen werden: Bodenabfertigung, Bodenabgabe, Bodenabschwemmung, Bodenabstand, Bodenabtrag, Bodenanalyse, Bodenart, Bodenaufteilung, etc. Aus der erhaltenen Liste kann der Benutzer dann exaktere

Begriffe auswählen und zu seiner Abfrage hinzufügen. Das Problem dabei ist, dass mit einem Wörterbuch rein alphabetisch gesucht wird und so Begriffe mit unterschiedlichem Wortanfang nicht gefunden werden (Grundteilung - Liegenschaftsteilung).

2.3.3 Semantische Suche

Eine mögliche Lösung ist eine semantische Suche. Dabei ‚versteht‘ der Computer die Fragestellung indem die Suchbegriffe Themenkreisen zugeordnet werden. Aus den Themenkreisen ergibt sich dann der Inhalt der Abfrage. Neue Suchbegriffe können dann zur Abfrage hinzugefügt werden oder die durchsuchten Texte auf ihre Relevanz bezüglich eines Themenkreises geprüft werden. Diese Möglichkeit ist beim Rechts-Informationssystem nicht umgesetzt.

2.3.4 Einsatz beim Lösen weiterer Probleme

Eine Anwendung einer Gesetzesdatenbank ist die Verfassung neuer Gesetze. Bei der Verfassung neuer Gesetze ist die einheitliche Verwendung von Begriffen notwendig. Das bezieht sich nicht nur auf das zu schaffende Gesetz, sondern auch auf alle bestehenden Gesetze. Eine Datenbankabfrage kann alle Gesetzesstellen liefern, in denen ein bestimmter Ausdruck verwendet wird. Anhand dieser Liste kann dann die Verwendung des Ausdrucks geprüft werden. Beim Einsatz einer semantischen Suche könnte man auch umgekehrt vorgehen und nach der zu beschreibenden Situation suchen. Wenn es für die Situation schon einen definierten Begriff gibt, liefert die Abfrage die betreffenden Textstellen und der Begriff kann konsistent weiterverwendet werden.

2.4 Expertensysteme

Expertensysteme dienen nicht nur der Suche nach Textstellen in Gesetzen, sondern sollen Juristen bei der rechtlichen Einschätzung von Situationen aktiv unterstützen. Dabei soll ein Expertensystem nicht nur alle auf eine bestimmte Situation anzuwendenden Gesetzesstellen finden, sondern auch noch die entsprechenden Schlüsse für die rechtliche Beurteilung ziehen. Notwendig dazu ist eine Methodik für die Speicherung der gesetzlichen Regeln. Dazu muss man nicht nur wissen, welche Bausteine ein Gesetz hat, sondern auch, wie diese zusammenhängen. Man benötigt also eine Ontologie der Welt und der rechtlichen Behandlung.

Es gibt verschiedene Ansätze für Expertensysteme. Ein Überblick über wichtige Vertreter kann in (Bench-Capon and Visser 1996) gefunden werden. Es beginnt bei einfachen

logikbasierten Ontologien, die Normen als logische Regeln darstellen und endet bei funktionalen oder rahmen-basierten Ansätzen.

2.4.1 Logikbasierter Ansatz

Ein Ansatz stammt von Friedrich Lachmayer (Lachmayer 1977). Der Ausgangspunkt seiner Arbeit ist die Norm als vorgeschriebene Verhaltensweise. Die Verhaltensweise darf, muss aber nicht, von einer Bedingung abhängig sein. Somit werden Normen in einer oder mehreren logischen Regeln dargestellt, die, je nach Ausgangslage, ein bestimmtes, vorgeschriebenes Verhalten zur Folge haben.

Die Formalisierung des Englischen Einwanderungsgesetzes (Sergot, Sadri et al. 1986) basiert ebenfalls auf einem logischen Ansatz. Hier wurde 1st-order Logik verwendet, um aus dem Gesetz ein Programm zu bauen. Problematisch ist jedoch, dass diese Art der Logik keine Möglichkeit bietet, Abläufe abzubilden. Zeitliche Abfolgen, wie sie im Grundbuch vorhanden sind, werden daher nicht klar sichtbar.

Edgar Morscher wählte ebenfalls einen logikbasierten Ansatz für seine Überlegungen zur Formalisierung von Recht (Morscher 1988). Er unterteilt die Formalisierung in:

- Formalisierung von Begriffen (die A-Formalisierung): Dabei sind die Begriffe selbständige sprachliche Ausdrücke, die keine ganzen Sätze bilden. Die Begriffe dürfen auch keineswegs mit der Bedeutung der Ausdrücke verwechselt werden. Die Formalisierung beschränkt sich nun darauf, die mehrdeutigen Begriffe eindeutig zu machen, vage Begriffe zu präzisieren oder qualitative Begriffe zu metrisieren.
- Formalisierung von Sätzen (die B-Formalisierung): Die Formalisierung eines Satzes ist entweder das Herausarbeiten der logischen Struktur eines Satzes oder die symbolische Abkürzung des Satzes.
- Formalisierung von Argumenten (die C-Formalisierung): Dabei ist ein Argument eine Folge von Sätzen. Von einem dieser Sätze (der Konklusio) wird der Anspruch erhoben, dass er aus den anderen (den Prämissen) folgt. Morscher nennt folgendes Beispiel:
 1. Alle Parscher sind Salzburger
 2. Alle Salzburger sind Österreicher
 3. Alle Österreicher sind Europäer
 4. Daher: Alle Parscher sind Europäer

Die Formalisierung sieht dann folgendermaßen aus:

1. $(\forall x)(Ax \rightarrow Bx)$; 2. $(\forall x)(Bx \rightarrow Cx)$; 3. $(\forall x)(Cx \rightarrow Dx)$; 4. Daher : $(\forall x)(Ax \rightarrow Dx)$

Dabei kommen die Prädikatvariablen ‚A‘ bis ‚D‘ frei vor während die Individualvariable ‚x‘ gebunden ist.

- Formalisierung einer Sprache (die D-Formalisierung): Bei Formalisierung der Sprache wird diese syntaktisch beschrieben. Dabei wird zunächst das Vokabular definiert (als die Menge aller erlaubten Aneinanderreihung von Buchstaben, also ohne ihre Bedeutung). Danach werden Regeln angegeben, wie diese Vokabel miteinander verknüpft werden dürfen, also die Grammatik der Sprache.
- Formalisierung der Logik oder eines logischen Systems (die E-Formalisierung): Diese Formalisierung ist eine D-Formalisierung, die um die Angabe von Umformungsregeln erweitert wurde. Dadurch werden Ableitungen und Beweise möglich.
- Formalisierung einer Theorie (die F-Formalisierung): Eine Formalisierung einer Theorie besteht aus einer Menge deskriptiver Zeichen Δ und einer Menge von Sätzen Γ , die mit Hilfe von Δ und logischen Ausdrücken formuliert werden können. Dabei muss Γ deduktiv abgeschlossen sein, also jeden Satz enthalten, der aus Γ folgt.
- Formalisierung einer semantischen Eigenschaft oder der Semantik insgesamt (die G-Formalisierung)

Bei der Diskussion der Vor- und Nachteile zeigt Morscher auf, dass Formalisierungen auf eindeutige Weise beschreiben, was eine Norm aussagt. Diese Beschreibung kann dann mit der Intention der Norm verglichen und die Norm gegebenenfalls korrigiert werden, bis sie genau das ausdrückt, was sie ausdrücken soll. Problematisch ist jedoch die Schreibweise, die für Laien nur schwer lesbar ist. Bereits das kleine Beispiel bei der C-Formalisierung enthält nur noch Variablen, die nichts mehr über den Inhalt aussagen. Das liegt möglicherweise daran, dass Morscher logische Ausdrücke verwendet, die schwer les- und überprüfbar sind. Wird eine besser lesbare Formalisierungsart verwendet, kann dieser Nachteil umgangen werden. Bei der Diskussion über die Grenzen einer Formalisierung kommt er zu dem Schluss, dass eine Formalisierung theoretisch möglich sein sollte. Lediglich bei der Formalisierung der Semantik (G-Formalisierung) stößt man auf Probleme, da hier keine sprachlichen Elemente mehr formalisiert werden, sondern semantische Eigenschaften und Relationen. Die sind aber nicht immer formalisierbar, wie das Beispiel einer Individuenkonstante zeigt, deren semantische Eigenschaft darin besteht, dass sie ein Ding bezeichnet (Carnap 1943).

2.4.2 Repräsentationssprache

Weitere Möglichkeiten zur Darstellung von Recht sind Repräsentationssprachen. Die Grundzüge einer solchen Sprache stammen von Thorne McCarty. Er entwickelte die ‚Language for Legal Discourse‘, LLD (McCarty 1989). Sein Ausgangspunkt für die Entwicklung von LLD ist das juristische Gespräch, sein Ziel ist eine Repräsentationssprache für juristisches Wissen mit deren Hilfe wissensbasierte Systeme Schlussfolgerungen ziehen könnten, die denen von Juristen gleichen.

Zur Modellierung verwendet McCarty einige Hilfsmittel, die bei reiner Logikprogrammierung nicht möglich sind. Auch bei LLD bestehen Regeln aus einem Basisausdruck auf der linken Seite und einem zusammengesetzten Ausdruck auf der rechten Seite. Neu ist allerdings, dass es Standardregeln und -beweise gibt, dass Zeit modelliert werden kann und dass es Ereignisse und Handlungen gibt.

Ein LLD-Modell besteht aus unterschiedlichen Elementen. Ein Beispiel für die Modellierung von Verpflichtungen kann in (McCarty to appear) gefunden werden. McCarty definiert Objekte und Beziehungen zwischen diesen Objekten. Die Beziehungen ordnen dabei die Objekte. Beispielsweise unterteilen sich Akteure in Personen und Firmen. Es gibt aber auch Beziehungen zwischen Objekten unterschiedlicher Art. Einer Firma ist eine Person zugeordnet, die die Aktionen der Firma kontrolliert. Zusätzlich gibt es Ereignisse und Handlungen sowie Modalitäten wie ‚gestattet‘, ‚verboten‘, ‚verpflichtend‘ und ‚ermöglicht‘.

Der Nachteil von LLD liegt bei der Modellierung von Systemen. LLD soll logische Argumentationen abbilden. Es gibt keine Möglichkeit die Regelwerke zu kapseln. Somit fehlt die Möglichkeit der Zusammenfassung von Teilen des Codes zu Einheiten, was die Lesbarkeit für den Benutzer stark erleichtern würde.

2.4.3 Agentenbasierter Ansatz

Ronald Stamper schlägt mit NORMA einen agentenbasierten Ansatz vor (Stamper 1991). Er geht von einer Kritik an traditioneller Logik aus. Traditionelle Logik verlässt sich auf symbolische Repräsentation und hat nur eine sehr schwache Verbindung mit den Konzepten der Realität. Zusätzlich ist es eine zu starke Vereinfachung, wenn man juristisches Wissen als reine Sammlung von Regeln repräsentiert. NORMA (Logik von Normen und Affordanzen), sein Vorschlag, enthält folgende ontologische Konzepte:

- *Agenten* sind Personen, die in der Realität existieren. Sie sammeln Wissen und beeinflussen die Realität, indem sie agieren und für ihre Aktionen die Verantwortung übernehmen. Dieses Konzept kann ausgeweitet werden, indem man es auf bürokratische Organisationen, Firmen und Staaten ausdehnt.
- *Verhaltensmässige Invariante* sind Beschreibungen von Situationen, deren Eigenschaften über einen bestimmten Zeitraum invariant bleiben. Beispielsweise ist ein Grundstück eine zusammenhängende Menge von Orten auf der Erdoberfläche, die von einer bestimmten geometrischen Form umschlossen sind und bestimmte Eigentums- und Belastungsverhältnisse aufweisen. Mögliche Aktionen mit Grundstücken sind Betreten, Bebauen, Verkaufen, Belasten, etc.
- *Realisierungen* sind eine Kombination der ersten beiden Punkte mit einem speziellen Agenten in einer speziellen Situation, also beispielsweise ‚das Grundstück wird verkauft‘.

Problematisch an dieser Lösung ist, dass der Zusammenhang zwischen Schaffung der Rechtsordnung und Speicherung im Expertensystem verloren geht. Die Rechtsquelle als Recht schaffendes Instrument ist nur in allgemeiner Form als Agent vorhanden. Daher sind Rückschlüsse vom Expertensystem auf die Gesetzestexte nur schwer möglich. Es ist beispielsweise unmöglich, erkannte Schwachstellen in der Rechtsordnung zunächst im Expertensystem ausmerzen und dann im Rahmen einer Gesetzesnovelle umzusetzen, da keinerlei Hinweis auf die Gesetzestexte vorhanden ist und man somit nicht sagen kann, wo die Novelle ansetzen muss.

Weitere Ansätze, die teilweise auch Ontologien von Normen aufbauen, stammen von Andre Valente (Valente 1995) beziehungsweise von Robert van Kralingen und Pepijn Visser (van Kralingen 1995; Visser 1995). Beide Ansätze sehen Gesetze im Kontext mit der Realität. Gesetze sollen die Gesellschaft auf bestimmte Weise beeinflussen. Sie sollen auf gesellschaftliches Verhalten reagieren und es in gewünschte Bahnen lenken. Das Hauptaugenmerk liegt auf den Auswirkungen der Gesetze auf die Gesellschaft. Es fehlen Methoden zur internen Überprüfung der Gesetze.

2.5 Zusammenfassung

Es hat sich gezeigt, dass es mit den bisherigen Ansätzen Probleme bereitet, Gesetze für den Computer verarbeitbar zu machen, Probleme bereiten. Die Methode, bei der der Text einfach in einer Datenbank gespeichert und der Computer benutzt wird um Textstellen zu finden, ist

unzureichend. Der Einsatz von Logik wiederum schafft es nicht, die in den Gesetzen enthaltene Struktur abzubilden.

Ein Problembereich bei logischen Systemen ist auch die Abbildung von zeitlichen Abfolgen. Eine Lösung für dieses Problem ist das ‚Situationskalkül‘ (McCarthy and Hayes 1969), der Veränderungen der Welt durch Aktionen zulässt. Was nicht modelliert werden kann ist der Umstand, dass jedes Gesetz in andere gesetzliche Regeln eingebettet ist. Die Abgrenzung zwischen dem zu modellierenden Gesetz und den notwendigen Voraussetzungen ist schwierig, da es nur eine Menge von logischen Regeln ohne Bezug zu Gesetzestexten gibt.

3 Algebraische Modelle

Wenn man mit dem Bau eines Modells beginnt muss man sich überlegen, welche Ansprüche man an die Methodik stellt. Dazu muss man sich zunächst vor Augen führen, welche Aufgaben ein Modell erfüllen soll (Ehrich, Gogolla et al. 1989, S. 7):

- *Korrektheit des Modells:* Es sollte möglich sein, zu prüfen, ob das Modell dem entspricht, was man damit ausdrücken wollte. Dazu benötigt man die Möglichkeit, Konsistenz- und Vollständigkeitsprüfungen durchzuführen. Auch zum Nachweis erwarteter Eigenschaften sollte das Modell herangezogen werden können.
- *Korrektheit der Implementierung:* Sollte ein Modell als Grundlage für eine Realisierung dienen, so muss die Möglichkeit vorhanden sein, Modell und Implementierung zu vergleichen um die Korrektheit der Implementierung zu überprüfen.
- *Automatische Entwurfshilfe:* Es sollte möglich sein, automatische Entwurfshilfen für die Modellierung zu schaffen, die Hilfestellung bei Entwurf und Implementierung des Modells bieten.

Aus diesen Aufgaben kann man Anforderungen ableiten, welche die für die Modellierung verwendete Sprache erfüllen muss. Eine Liste von Anforderungen könnte wie folgt aussehen:

- Die verwendete Sprache sollte unabhängig von der natürlichen Sprache sein.
- Man sollte die Semantik ausdrücken, also Struktur und Operationen definieren können.
- Die verwendete Logik sollte möglichst einfach sein.
- Die resultierenden Dokumente sollten kurz sein.
- Das Modell sollte testbar sein, also sollte die Sprache ausführbar sein.

Die ersten vier Punkte sind bei Verwendung von Algebren erfüllt. Probleme gibt es nur beim letzten Punkt. Hier muss man sich ein geeignetes Hilfsmittel suchen, mit dem man Algebren definieren kann. Abschnitt 3.2.5 schlägt eine geeignete Programmiersprache vor.

3.1 Mathematischer Hintergrund

Algebra wird in vielen Büchern über Mathematik behandelt (Athen and Bruhn 1980; Reinhardt and Soeder 1991). Dort findet man auch eine Definition von Algebra. Hier ist ein Beispiel (Reinhardt and Soeder 1991, S. 37):

Die Algebra beschäftigt sich mit den Eigenschaften algebraisch strukturierter Mengen (z.B. Gruppen, Ringe, Körper, Moduln, Vektorräume). Die Mengen können endlich oder unendlich sein. Die algebraische Struktur wird durch innere und äußere Verknüpfungen vermittelt, die spezielle Eigenschaften besitzen (z.B. assoziatives Gesetz, kommutatives Gesetz, Existenz eines neutralen Elementes und inverser Elemente, distributives Gesetz).

Die Algebra beschäftigt sich somit mit Mengen, die eine algebraische Struktur besitzen. Es gibt auch andere Arten von Strukturen nämlich Ordnungsstrukturen und Topologische Strukturen. Eine strukturierte Menge oder Struktur liegt dann vor, wenn Verknüpfungen für die Menge definiert sind, die wiederum durch Axiomensysteme definiert werden. Die in den Strukturen auftretenden abstrakten Begriffe können mit mathematischen Modellen belegt werden. Ein Ergebnis, welches aus dem Studium der Strukturen hervorgeht, gilt dann für sämtliche Modelle dieser Struktur.

Eine Verknüpfung \bullet besteht also aus einer Vorschrift, nach der zwei Elemente $a \in M_1$ und $b \in M_2$ genau ein Ergebnis $\bullet(a; b) \in M$, das Verknüpfungsergebnis, liefern. Ein Spezialfall, die innere Verknüpfung, liegt dann vor, wenn die beiden Elemente a und b derselben Menge M wie das Verknüpfungsergebnis entstammen. Ein Beispiel für eine solche innere Verknüpfung wäre die Multiplikation auf der Menge der ganzen Zahlen, da man dann eine Abbildung der Form $Z \times Z \rightarrow Z$ hat.

Eine abstraktere Definition findet sich in (Kaiser, Mlitz et al. 1985):

Universale Algebra: Unter einer solchen versteht man ein Paar $\langle A, \mathbf{W} \rangle$, wobei A eine beliebige, nichtleere Menge ist (die man Grundmenge oder Trägermenge nennt) und \mathbf{W} ein System von Operationen auf A ist.

Ein Beispiel für eine Algebra ist die Menge der ganzen Zahlen mit den zugeordneten Verknüpfungen Addition und Subtraktion, die eine Abelsche (oder kommutative) Gruppe bilden (Frank 1999):

Algebra AbelscheGruppe (Z)

Verknüpfungen $+, - :: Z \rightarrow Z \rightarrow Z$

negieren $:: Z \rightarrow Z$

Null $:: Z$

<i>Axiome</i>	$a + b = b + a$	<i>Kommutativgesetz</i>
	$(a + b) + c = a + (b + c) = a + b + c$	<i>Assoziativgesetz</i>
	$Null + a = a + Null = a$	<i>Nullelements</i>
	$a + (\text{negieren } a) = Null$	<i>Inverses Element</i>
	$a - b = a + (\text{negieren } b)$	<i>Definition der Subtraktion</i>

Die verwendeten Begriffe wie ‚+‘ oder ‚-‘ gelten nur innerhalb des Kontextes in dem sie in der Algebra definiert sind. Außerhalb dieses Kontextes haben sie keine Bedeutung. Da die Definition des Begriffes Teil der Algebra ist, ist man von der natürlichen Sprache unabhängig.

Die Definition von Addition und Subtraktion ist kurz und einfach. Wenn man versucht, dieselbe Definition in Worten niederzuschreiben wird man bald auf Probleme mit der Allgemeingültigkeit der Aussagen stoßen. Um diese Probleme lösen zu können, muss die Textlösung um einiges länger sein als die mathematische Lösung.

3.2 Algebraische Modelle

Für eine Formalisierung benötigt man dieselben Elemente wie sie bei Algebren vorhanden sind. Der letzte Abschnitt hat gezeigt, dass eine Algebra aus Mengen und Verknüpfungen besteht. Diese Einteilung kann man auch verwenden um andere Dinge als Zahlenmengen zu beschreiben. Die Modellierung eines Stack kann beispielsweise folgendermaßen aussehen – nach der Definition von (Guttag, Horowitz et al. 1978):

Algebra Stack (Stack von a, a)

<i>Verknüpfungen</i>	$Push :: a \rightarrow Stack\ von\ a \rightarrow Stack\ von\ a$	<i>Konstruktor</i>
	$New :: Stack\ von\ a$	<i>Konstruktor</i>
	$Pop :: Stack\ von\ a \rightarrow Stack\ von\ a$	<i>Observer</i>
	$Top :: Stack\ von\ a \rightarrow a$	<i>Observer</i>
<i>Axiome</i>	$Top (Push\ a\ s) = a$	
	$Pop (Push\ a\ s) = s$	
	$Top (New) = Error$	
	$Pop (New) = Error$	

Dabei wird nicht beschrieben, aus welchen Elementen die Menge besteht. Es wird nur beschrieben, wie sich die Elemente verhalten, also welche Eigenschaften sie haben. Im ange-

gebenen Beispiel wird nie definiert, was für Elemente der Stack speichert oder wie der Stack realisiert ist. Es wird somit nicht auf die Implementierung eingegangen, sondern nur eine Beschreibung geliefert. Es wird nur angegeben, wie sich ein Objekt ‚Stack‘ in bestimmten Situationen zu verhalten hat.

Der Parameter ‚a‘ im Beispiel oben ist ein Parameter für einen beliebigen Datentyp. Datentypen werden benötigt, um Informationen darzustellen. Für die Darstellung von Informationen braucht man einen Vorrat an Datenelementen wie beispielsweise Bitfolge, Zahlen, Buchstabenfolgen, etc. Die Gesamtheit der möglichen Datenelemente wird üblicherweise in Datentypen unterteilt wobei jeder Datentyp für eine bestimmte Menge an Datenelementen (z.B. die Menge aller Buchstabenfolgen) steht. Wenn man allgemeine Verhaltensweisen beschreiben will, kann man nicht von vornherein sagen, welche Datentypen später tatsächlich verwendet werden. Verwendet wird daher ein Platzhalter, der anzeigt, dass an den Stellen, an denen der Parameter vorkommt, bei der Realisierung der beschriebenen Algebra ein spezieller Datentyp eingesetzt werden muss.

3.2.1 Teile einer Algebra

Ein algebraisches Modell besteht aus drei Teilen. Typnamen beschreiben Typparameter und geben dadurch an, aus welchen Teilen die Algebra besteht. Operationen definieren die inneren und äußeren Verknüpfungen der Algebra. Schließlich gibt es Axiome, welche das Verhalten der Algebra beschreiben.

Typnamen stehen in einer Algebra als Parameter für spezifische Typen. Zum Zeitpunkt der Beschreibung einer Algebra ist es nicht wichtig, wie die einzelnen Typen tatsächlich realisiert sind. Es ist nur wichtig, bestimmte Eigenschaften der Typen zu kennen. So kann es beispielsweise bei manchen Typen notwendig sein, dass es sich um ganze Zahlen handelt, andere sind rationale Zahlen oder Boolesche Werte. Jede Kategorie erlaubt ganz bestimmte Operationen. Für ganze Zahlen sind beispielsweise Addition, Subtraktion und Multiplikation definiert, während bei rationalen Zahlen auch die Division möglich ist. Wie diese Kategorien allerdings realisiert werden, wird nicht ausgesagt und ist für die Modellierung auch nicht notwendig. Zu beachten ist ebenfalls, dass Parameter nur innerhalb der Algebra gültig sind. Das ist vergleichbar mit den Parameternamen in mathematischen Funktionen. Die Parameter ‚a‘ in $f(x,y,a) = \dots$ und $g(z,a,b) = \dots$ haben nichts miteinander zu tun.

Axiome beschreiben das Verhalten von Operationen. Axiome werden so niedergeschrieben, dass sie unabhängig von tatsächlichen Werten für die Parameter sind. Die Anzahl

der Axiome, die für eine vollständige Definition notwendig sind, kann in der Praxis so bestimmt werden, dass man jeden Konstruktor durch jeden Observer ausdrückt. Für das Beispiel ‚Stack‘ heißt das, dass bei zwei Constructoren (Push, New) und zwei Observern (Pop, Top) vier Axiome notwendig sind.

3.2.2 Kombination von Algebren

Die Kombination von Algebren soll an einem einfachen Beispiel erläutert werden. Es beschreibt die Erweiterung einer bestehenden Algebra um neue Verknüpfungen. Die Theorie zur Kombination von Algebren und Beispiele dazu findet man in (Ehrich, Gogolla et al. 1989; Loeckx, Ehrich et al. 1996; Frank 1999).

Wenn man die Abelsche Gruppe aus Abschnitt 3.1 mit der Multiplikation ergänzen will, so benötigt man vier neue Verknüpfungen und sieben neue Axiome:

Verknüpfungen $*$, $/$, *Kehrwert*, *Eins*

<i>Axiome</i>	$a * b = b * a$	<i>Kommutativgesetz für die Multiplikation</i>
	$a * (b * c) = (a * b) * c = a * b * c$	<i>Assoziativgesetz für die Multiplikation</i>
	$a * \text{Eins} = \text{Eins} * a = a$	<i>Einselement</i>
	$a * (\text{Kehrwert } a) = \text{Eins}$	<i>Existenz der Inversen</i>
	$a * (b + c) = a * c + b * c$	<i>Distributivgesetze</i>
	$(b + c) * a = b * a + c * a$	
	$a / b = a * (\text{Kehrwert } b)$	<i>Ableitung der Division</i>

Dabei muss angegeben werden, dass es sich bei ‚a‘, ‚b‘ und ‚c‘ um Zahlen handelt. Die Abelsche Gruppe wird vorausgesetzt. Daher sind Addition und Subtraktion für die Parameter definiert. Die entsprechenden Verknüpfungen können somit bei den Axiomen für die Multiplikation verwendet werden.

3.2.3 Teile Algebraischer Modelle

Ein algebraisches Modell setzt sich aus unterschiedlichen Algebren zusammen. Diese Algebren werden miteinander verknüpft, indem einfache Elemente zu komplizierteren Elementen verbunden werden. Zusätzlich zu den aus Abschnitt 3.2.1 bekannten Teilen der Algebra kommen also noch Vererbungsinformationen. Somit besteht ein algebraisches Modell aus folgenden Teilen:

- *Typparameter*: Die Algebra verwendet unterschiedliche Datentypen, die hier als Parameter angegeben werden.
- *Voraussetzungen für die Typparameter*: Jeder Typparameter kann Eigenschaften von anderen Algebren genügen. Zur Laufzeit müssen also für die verwendeten Datentypen die entsprechenden Algebren realisiert sein.
- *Konstruktoren/Observer*: Diese Arten von Funktionen dienen dazu, die Beschreibung von der Realisierung unabhängig zu machen. Sie ermöglichen Zerlegen und Zusammen setzen der Daten, die im Typparameter zusammengefasst sind.
- *Operationen*: Diese Funktionen sind aus Konstruktoren, Observern und Operationen der eigenen Algebra und der vorausgesetzten Algebren zusammengesetzt. Eine solche Operation hat jeweils zwei Teile:
 - *Signatur*: Die Signatur definiert den Namen der Operation, die Typen der Eingangsparameter und den Typ des Ergebnisses.
 - *Axiome*: Die Axiome liefern die Vorschriften, wie man von den Parametern zum Ergebnis kommt.

3.2.4 Funktionale Programmiersprachen

Die Operationen können auf reine Funktionen beschränkt werden (Abadi and Cardelli 1996). Funktionen besitzen Eingangsparameter und ein eindeutiges Ergebnis, das nur von den Eingangsparametern abhängig ist. Die Eingangsparameter selbst werden nicht verändert. Operationen, welche die Eingangsparameter ändern, können umgeschrieben werden, sodass sie zu Funktionen werden. Konstante Werte, wie beispielsweise Pi, können als konstante Funktionen angesehen werden. Die Angabe der Typen für Eingangsparameter und Ergebnis ist die Signatur der Operation. Man kann die Operationen in drei Kategorien unterteilen:

- *Konstruktoren* sind Funktionen, die als Eingangsparameter die Einzelteile des Gesamttyps haben und den Gesamttyp als Ergebnis liefern. Ein Konstruktor einer rationalen Zahl hat zwei ganze Zahlen als Eingangsparameter und kombiniert sie zu einer rationalen Zahl. Konstruktoren sind von der gewählten Implementierung abhängig.
- *Observer* sind das Gegenstück zu den Konstruktoren. Sie haben den Gesamttyp als Eingangsparameter und liefern einen Teil davon zurück. Für die rationalen Zahlen benötigt

man beispielsweise zwei Observer, von denen einer den Zähler und einer den Nenner liefert. Observer sind ebenfalls von der gewählten Implementierung abhängig.

- *Abgeleitete Funktionen* sind Abkürzungen für Kombinationen von Konstruktoren, Observern und anderen abgeleiteten Funktionen. So kann beispielsweise die Subtraktion unter Verwendung der Negation definiert werden: $a - b = a + (\text{negieren } b)$. Sie dienen also der Zusammenfassung von einfachen Operationen zu komplexeren Operationen.

Abgeleitete Funktionen können oft unabhängig von dem später verwendeten Datentyp definiert werden. Um beispielsweise die Länge einer Liste zu bestimmen ist keine Kenntnis über den Typ der Listenelemente notwendig. Es wird einfach so lange das erste Element der Liste entfernt, bis die Liste kein Element mehr enthält. Die Anzahl der durchlaufenen Arbeitsschritte wird dabei mitgezählt.

3.2.5 Verwendung von Haskell für algebraische Modelle

Um algebraische Modelle definieren zu können benötigt man eine Programmiersprache, die mathematisch ‚sauber‘ ist. Mathematische Aussagen sind bezüglich ihrer Korrektheit überprüfbar. Programme, die auf Mathematik basieren, sind somit ebenfalls überprüfbar, wenn die verwendete Sprache Konzepte der Mathematik einsetzt. Das von funktionalen Programmiersprachen verwendete Hilfsmittel ist die Substitution.

3.2.5.1 Substitution in Programmiersprachen

Die Verwendung von Substitution vereinfacht die Modellierung, da komplexe Probleme auf einfache Probleme zurückgeführt werden. Die Korrektheit der einfachen Probleme sowie die Korrektheit der Substitution ist leicht zu beweisen. Das mathematische Prinzip der Substitution wird beim Umformen und Kombinieren algebraischer Axiome häufig verwendet. Eine prozedurale Programmiersprache erlaubt folgende problematische Zuweisung:

$$A := A + 1$$

Die Zuweisung erhöht den Wert einer Variablen ‚A‘ um den Wert 1. Das Problem mit dieser Formel ist, dass der Ausdruck ‚A‘ auf der linken Seite der Zuweisung nicht identisch ist mit dem Ausdruck ‚A‘ auf der rechten Seite. Der Beweis dafür ist einfach zu führen:

$$A := A + 1$$

Einsetzen des Ausdruckes für ‚A‘ in die rechte Seite der Zuweisung gibt:

$$A := (A + 1) + 1$$

$$A := A + 2$$

Durch Substitution hat man eine neue Situation erhalten, die eindeutig mit der ursprünglichen Zeile nicht übereinstimmt. Substitution ist also kein erlaubtes Hilfsmittel, wenn es Variablen gibt, die ihren Wert ändern können. Da alle prozeduralen Programmiersprachen solche Zuweisungen erlauben, bleiben nur mehr funktionale Programmiersprachen übrig. Eine geeignete funktionale Programmiersprache ist Haskell (Peyton Jones, Hughes et al. 1999).

3.2.5.2 Beschreibung von Algebren mit Haskell

Andrew Frank und Werner Kuhn schlagen Haskell für eine formale Spezifizierung von OpenGIS vor (Frank and Kuhn 1995; Frank 1999; Frank and Kuhn 1999). Auch Stephan Winter und Silvia Nittel (Winter and Nittel to appear) schlagen vor, Haskell statt der momentan verwendeten Sprache UML einzusetzen, da diese Methode folgende Vorteile besitzt:

- Die Semantik der beschriebenen Schnittstellen kann formal beschrieben werden.
- Die Beschreibung ist vollständig unabhängig von der Implementierung.
- Es steht sofort ein ausführbarer Prototyp zur Verfügung.
- Die algebraische Struktur ist problemlos erweiterbar.

Ein kurzes Beispiel soll zeigen, wie man Algebren mit Haskell definieren kann. Das Beispiel wurde aus dem Haskell Prelude (siehe Anhang C) entnommen. Die Syntax der Sprache kann in (Thompson 1996; Hudak, Peterson et al. 1997; Bird 1998) gefunden werden.

```
class (Eq a, Show a) => Num a where
    (+), (-)    :: a -> a -> a
    negate     :: a -> a

    -- Minimal complete definition: All, except negate or (-)
    x - y      = x + negate y
    negate x   = 0 - x
```

Diese Klasse definiert die Abelsche Gruppe aus Abschnitt 3.1. Als Voraussetzung für die Elemente der behandelten Menge ist verlangt, dass die Klassen ‚Eq‘ (Gleichheit und Ungleichheit) sowie ‚Show‘ (Darstellung des Inhaltes) für die Elemente definiert sind. Daher könnten die in diesen Klassen enthaltenen Funktionen hier verwendet werden. In der ursprünglichen Abelschen Gruppe waren die Funktionen ‚+‘, ‚-‘, ‚negieren‘ und ‚Null‘ enthalten. Mit Ausnahme der Funktion ‚Null‘ (neutrales Element der Addition) sind alle Funktionen auch in

der Klasse enthalten. Wie beim mathematischen Vorbild ist die Subtraktion über Addition und Negation definiert. Unterschiedlich ist allerdings, dass auch der umgekehrte Fall (Definition der Negation über die Subtraktion) vorhanden ist. Die beiden Axiome verwenden sich somit gegenseitig bei der Definition. Bei einer Realisierung der Klasse für einen spezifischen Datentyp muss also eines der beiden Axiome neu definiert werden, um eine ausführbare Spezifikation zu erhalten.

Um die Klasse nun durch die Division zu ergänzen, muss man eine neue Algebra auf der bereits existierenden Algebra aufsetzen:

```
class (Num a) => Fractional a where
  (/)          :: a -> a -> a
  recip        :: a -> a

  -- Minimal complete def.: fromRational and (/) or recip)
  recip x      = 1 / x
  x / y        = x * recip y
```

Von den vier Funktionen, die in Abschnitt 3.2.2 als notwendige Funktionen aufgelistet wurden, werden zwei von dieser Erweiterung realisiert. Die Multiplikation als dritte Funktion ist bereits in der Klasse ‚Num‘ enthalten und braucht somit hier nicht noch einmal definiert werden. Die einzige fehlende Funktion ist das neutrale Element bezüglich der Multiplikation. Eines der Axiome für Division und Kehrwert benötigen wieder eine Definition bei der Realisierung für einen bestimmten Datentyp, da sie sich gegenseitig aufrufen.

3.3 Objektorientierte Modellierung

Objektorientierte Modelle fassen Funktionen und Daten zusammen. Die Gruppe von Funktionen und die Datenstruktur heißen ‚Klasse‘. Die Datenstruktur legt dabei die Repräsentation der Klasse durch Daten und somit die Implementierung fest. Eine Klasse beschreibt also sowohl die Funktionalität als auch die Implementierung. Eine Realisierung mit einem bestimmten Datensatz ist dann ein Objekt.

Vererbung verwendet die Funktionen einer Klasse, um eine neue Klasse zu definieren. Man leitet dabei von einer allgemeinen Klasse eine spezifischere Klasse ab. Dadurch werden Hierarchien unterschiedlicher Abstraktion geschaffen. Je allgemeiner nun eine Operation ist, desto früher wird sie definiert.

Parametrisierte Klassen ermöglichen die Definition von Axiomen, unabhängig von der Repräsentation. Dabei werden die verwendeten Datentypen nur als Typparameter angegeben. Diese Typparameter werden zur Laufzeit durch den tatsächlich gegebenen Datentyp ersetzt. Daher muss der Datentyp erst zur Laufzeit bekannt sein.

Eine Einschränkung vieler Modellierungstools ist die unvollständige Unterstützung objektorientierter Methoden. Einschränkungen gibt es hauptsächlich bei Mehrfachvererbung, bei der eine Klasse Methoden mehrerer anderer Klassen erbt. Die Notwendigkeit von Mehrfachvererbung haben Max Egenhofer und Andrew Frank für das Beispiel GIS gezeigt (Egenhofer and Frank 1992). Auch im Recht ist Mehrfachvererbung notwendig, da Aktionen Rechtsfolgen in verschiedenen Rechtsbereichen haben können. Ein Beispiel wäre eine Körperverletzung. Einerseits handelt es sich um eine Straftat, die nach öffentlichem Recht geahndet wird, andererseits ist aber auch ein zivilrechtlicher Schmerzensgeldanspruch damit verbunden. Somit erben die Rechtsfolgen Eigenschaften aus dem zivilrechtlichen und dem öffentlich-rechtlichen Bereich. Parametrisierte Klassen sind ebenfalls nur in wenigen Programmiersprachen möglich.

3.4 Unterschiede zu objektorientierter Programmierung

Sowohl objektorientierte auch algebraische Modelle fassen Funktionen zu Einheiten zusammen. Bei algebraischen Modellen heißt eine solche Einheit Algebra, bei objektorientierten Modellen heißt sie Klasse. Anders als die Algebra enthält eine Klasse nicht nur die Funktionen sondern auch eine Beschreibung der Repräsentation. Das bedeutet, dass Code und Repräsentation nicht getrennt sind. Dadurch ist die Wiederverwendbarkeit des Codes eingeschränkt, da für andere Repräsentationen neue Funktionen geschrieben werden müssen. Diese Einschränkung kann durch parametrisierte Klassen umgangen werden.

3.5 Zusammenfassung

Dieses Kapitel hat die Grundlagen von Algebren gezeigt. Zunächst wurden die mathematischen Grundlagen der Algebra behandelt. Danach wurde aufgezeigt, wie man mit Hilfe von Algebren Modelle bauen kann. Schließlich wurde noch eine Programmiersprache vorgestellt, mit der Algebren nicht nur aufgeschrieben, sondern auch ausgeführt werden können. Das hat den Vorteil, dass Fehler in der Algebra sofort auffallen und daher einfacher zu beheben sind.

Die Vorteile einer Modellierung mit Algebren sind die Zerlegung in einfache Probleme, die Einfachheit des erzeugten Programmcodes und die Beweisbarkeit der Vollständigkeit des Modells. Bei der Modellierung mit Algebren zerlegt man ein Problem so lange es sinnvoll

durchführbar ist. Diese Zerlegung erfolgt in vielen kleinen Schritten. Sieht man sich nun das Modell vom kleinsten Teil zum gesamten fortschreitend an, so kommen bei jedem Schritt einige Funktionen hinzu. Da aber die Schritte klein gehalten sind, ist die Anzahl der neuen Funktionen überschaubar. Dadurch kann man das gesamte Modell einfacher durchschauen als dies bei anderen Arten der Modellierung der Fall ist. Hinzu kommt noch, dass durch diese kleinen Schritte auch die Funktionen selbst einfach werden und somit der Programmcode leicht verständlich ist, da jede Funktion nur eine kleine Aufgabe ausführt. Schließlich bieten Algebren auch die Möglichkeit, die Vollständigkeit der Funktionen zu überprüfen. Das beginnt bei der Vollständigkeit der Observer und endet bei der Möglichkeit, die Vollständigkeit der Pattern-Matching-Zeilen zu überprüfen.

4 Gesetzestexte und algebraische Modelle – ein Vergleich

Bisher wurden Gesetze und Algebren getrennt voneinander betrachtet. Der nächste Schritt muss nun ein Vergleich der beiden Systeme sein. Dieses Kapitel enthält daher zunächst eine Diskussion der Gemeinsamkeiten, wobei das Hauptaugenmerk darauf gerichtet ist, welche Strukturen der Gesetzestexte auch in Algebren vorkommen. Im zweiten Teil dieses Kapitels werden dann die Unterschiede zwischen Gesetzestexten und Algebren untersucht.

4.1 Ähnlichkeiten

Sowohl Algebren als auch Gesetzestexte beschreiben Ideen. Beide Methoden beschreiben Modelle der Ideen. Die verwendete Syntax ist zwar unterschiedlich, aber trotzdem arbeiten beide Methoden mit vergleichbaren Hilfsmitteln, nämlich Zerlegung, Abstraktion und Axiomatisierung.

4.1.1 Beschreibung eines Modells

Die Beschreibung eines Modells benötigt eine Sprache, mit deren Hilfe man die Elemente des Modells und ihr Zusammenwirken festlegt. Gesetzestexte bedienen sich der natürlichen Sprache für diese Beschreibung, während Algebren die mathematische Notation nutzen. Trotz der unterschiedlichen Sprache bestehen beide Modelle aus Teilen, die zu einem Ganzen zusammengesetzt werden. Zwei Beispiele, ein Gesetzestext (§§ 1 und 2 GBG55) und eine Algebra sollen die Ähnlichkeit zeigen.

§ 1. Das Grundbuch besteht aus dem Hauptbuch und der Urkundensammlung.

§ 2. (1) Das Hauptbuch wird aus den Grundbuchseinlagen gebildet.

(2) Die Grundbuchseinlagen sind bestimmt zur Eintragung:

1. der Grundbuchskörper und ihrer Änderungen;

*2. der sich auf die Grundbuchskörper beziehenden dinglichen Rechte
und ihrer Änderungen.*

Diese beiden Paragraphen aus dem Allgemeinen Grundbuchsgesetz führen Begriffe und die Zusammenhänge zwischen ihnen ein. Abbildung 1 zeigt die im Gesetz beschriebene Struktur. In diesem kleinen Ausschnitt werden keine Operationen definiert. Außerdem werden die Begriffe nur vorgegeben und bedürfen deshalb einer näheren Definition. So wird zum Beispiel die Urkundensammlung erwähnt. Erst anschließend (§ 6 GBG55) wird gesagt, dass diese durch

Abschriften von Urkunden gebildet wird. Die Definition der Begriffe ist somit teilweise im weiteren Verlauf des Gesetzestextes enthalten. Teilweise ist sie aber nur in anderen Gesetzen zu finden. Ein Beispiel dafür wurde in Abschnitt 1.1.3 mit der Definition von dinglichen Rechten und Lasten gegeben. Dasselbe gilt auch für die Zusammenhänge zwischen den einzelnen Begriffen. Der Vorteil dieser Vorgangsweise ist, dass der Begriff im Gesetzeskontext eindeutig definiert ist. Problematisch ist jedoch, dass der Begriff unverändert beibehalten wird, auch wenn er die tatsächliche Situation nicht mehr genau trifft oder sich die sprachliche Bedeutung des Wortes ändert. Beispielsweise heißt das Grundbuch noch immer ‚Buch‘, obwohl es mittlerweile digital geführt wird. Daher ist es eigentlich kein Buch mehr sondern eine Datenbank.

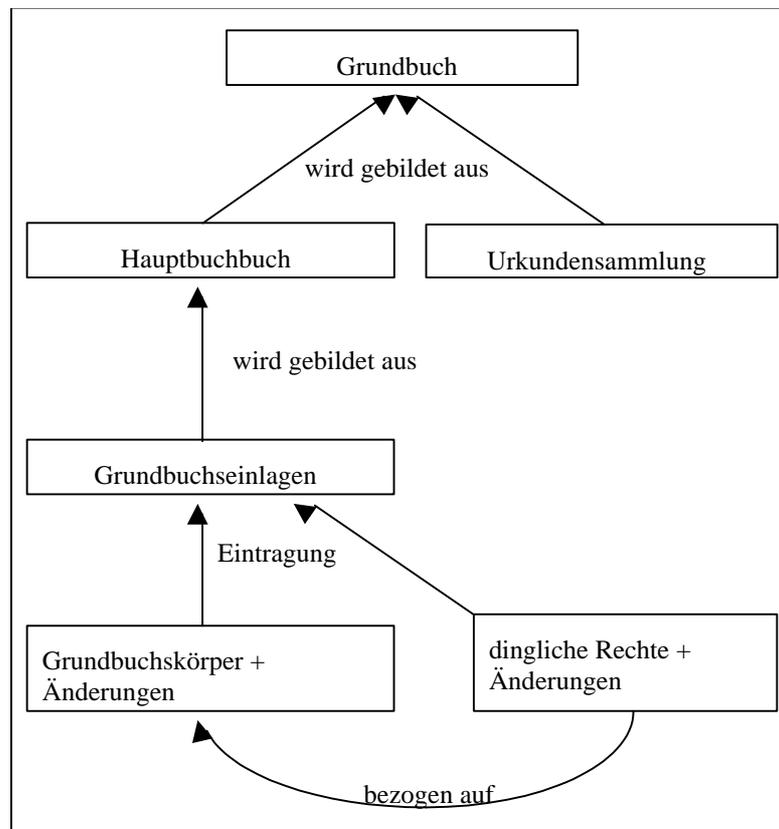


Abbildung 1: Modellierung im Gesetzestext

4.1.2 Zerlegung des Modells in kleine Teile

Beide Methoden versuchen, das Gesamtproblem in Teilaspekte aufzuspalten und einzeln zu beschreiben. Die Rechtsordnung gliedert sich in Gesetze für unterschiedliche Rechtsbereiche. Algebraische Modelle sehen ähnlich aus. Auch bei ihnen gibt es für unterschiedliche Aspekte unterschiedliche Algebren. Beispielsweise kann ein schiffbarer Fluss sowohl als Schifffahrtsweg, als auch als (fließendes) Gewässer, aber auch als Hindernis (beispielsweise im Straßenbau)

angesehen werden. Zu jedem dieser Aspekte gehört eine eigene Algebra, da die Aspekte unterschiedliche Aktionen ermöglichen oder verlangen.

Innerhalb des Gesetzes erfolgt eine weitere Unterteilung. Einerseits werden Begriffe definiert, andererseits Vorgangsweise, Abläufe und Auswirkungen vorgegeben. Innerhalb einer Algebra erfolgt ebenfalls eine weitere Gliederung. Eine Algebra definiert Verknüpfungen, also mögliche Operationen. Somit definieren diese Verknüpfungen die Interaktionsmöglichkeiten der Algebra mit anderen Algebren und (im Fall von inneren Verknüpfungen) die Aspekte die sich aus den Eigenschaften der Algebra selbst ergeben. Ein Beispiel für einen solchen Aspekt wäre die Länge eines Vektors, der sich aus den Grundelementen (den Längen in den Achsrichtungen) ergibt.

4.1.3 Abstrakte Beschreibung von Situationen und Folgen

Sowohl Gesetzestexte, als auch algebraische Modelle versuchen, Situationen möglichst allgemein zu beschreiben. Ein Paragraph besteht aus den Teilen Tatbestand und Rechtsfolge. Ein Tatbestand ist eine typische Verhaltensweise oder Situation, die Rechtsfolge die sich daraus ergebende Wirkung. Manchmal (wie beispielsweise in §1 GBG55) gibt es keine Rechtsfolge, da einfach ein Begriff definiert wurde. Dann kann die Rechtsfolge auch entfallen. In beiden Fällen ist es notwendig, den Tatbestand beschreiben zu können. Prinzipiell gibt es dafür zwei Möglichkeiten. Einerseits kann man die Situation durch prototypische Beispiele charakterisieren, andererseits kann man typische Eigenschaften angeben. Üblicherweise wird der zweite Weg gewählt.

4.1.4 Axiomatische Definitionen und Beschreibungen

Sowohl Gesetzestexte, als auch Algebren definieren die notwendigen Verknüpfungen durch Axiome. In Gesetzen werden diese Axiome als Richtlinien definiert. Ein Beispiel dafür ist § 21 GBG55:

§ 21. Eintragungen sind nur wider den zulässig, der zur Zeit des Ansuchens als Eigentümer der Liegenschaft oder des Rechtes, in Ansehung deren die Eintragung erfolgen soll, im Grundbuch erscheint oder doch gleichzeitig als solcher einverleibt oder vorgemerkt wird.

Hier wird eine allgemeingültige Bedingung definiert. Es wird von der ‚Eintragung‘, dem ‚Ansuchen‘, dem ‚Eigentümer‘ und dem ‚Grundbuch‘ gesprochen. Keiner dieser Begriffe wird näher beschrieben oder eingeschränkt. Es werden nur bestimmte Beziehungen zwischen den

Begriffen hergestellt und die so geschaffene Situation mit einem Ergebnis verknüpft. Somit gilt die Regel für alle Situationen, die dem vorgegebenen Muster entsprechen. Die Auswirkung des Paragraphen auf den Vorgang der Eintragung ist durch den Satzteil ‚... ist nur wider den zulässig, ...‘ definiert. Immer dann, wenn die angegebene Bedingung nicht erfüllt ist, liegt eine unzulässige Eintragung vor. Die Rechtsfolge einer unzulässigen Eintragung ist aber wieder nicht Teil der Definition.

Algebren verwenden ebenfalls Axiome um Verknüpfungen zu beschreiben und Eigenschaften zu definieren. Um beispielsweise ein neutrales Element zu definieren, kann man folgenden Ausdruck verwenden:

$$n \circ x = x \circ n = x \quad \forall x \in M$$

Dieser Ausdruck beschreibt für eine Menge M ein neutrales Element n bezüglich einer Operation \circ dadurch, dass die Auswirkung der Verwendung von n beschrieben wird. Im Fall des neutralen Elementes ist die Auswirkung einfach die, dass die Operation weggelassen werden kann, da das Ergebnis gleich dem zweiten Eingangsparameter ist.

4.1.5 Verknüpfungen

Gesetze enthalten oft Verweise auf andere Gesetze in expliziter Form. Diese Verweise sind für den Leser hilfreich, da sie auf den Zusammenhang zwischen verschiedenen Gesetzen hinweisen. Somit kann es nicht mehr so leicht passieren, dass ein Zusammenhang übersehen wird. Ein solcher Verweis kann in unterschiedlicher Form auftreten:

- *Als Anmerkung:* Zum reinen Gesetzestext gibt es oft Anmerkungen, die den Text näher erläutern und Missverständnissen vorbeugen sollen. In solchen Anmerkungen kann ein Verweis auf ein anderes Gesetz vorkommen (z.B. Anm. zu § 1 GBG55: *Zum Hauptbuch siehe insbesondere § 2 Abs. 1 sowie § 68 Abs. 1 AllgGAG, BGBl. Nr. 2/1930, zur Urkundensammlung § 6 sowie § 13 AllgGAG.*)
- *Direkt im Paragraphen* (z.B. § 9 GBG55: *Im Grundbuch können nur dingliche Rechte und Lasten, ferner das Wiederkaufs- und das Vorkaufsrecht (§§ 1070 und 1073 ABGB.) sowie das Bestandsrecht (§ 1095 ABGB.) eingetragen werden.*)

Verweise in Anmerkungen deuten auf zusätzlich zu beachtende Regelungen hin. Anmerkungen dienen dazu, dem Leser das Verständnis des Paragraphen zu erleichtern. In diesem Sinne ist es oft hilfreich, andere Gesetzestexte zu lesen, die sich mit den Problemen aus einem anderen Gesichtspunkt beschäftigen, wie der vorliegende Text. Im oben angegebenen Beispiel dient der

Verweis dazu, nähere Angaben zu den Begriffen ‚Hauptbuch‘ und ‚Urkundensammlung‘ zu liefern. An den angegebenen Stellen stehen dann Definitionen, die entweder Spezialfälle oder neue Zusammenhänge beschreiben.

§ 2. (1) Die Grundbuchseinlagen je einer Katastralgemeinde bilden zusammen ein Hauptbuch.

§ 6. (1) Jede Grundbuchseinlage besteht aus dem Gutsbestandsblatte, dem Eigentumsblatte und dem Lastenblatte.

(2) Bei Grundbuchskörpern, die im Miteigentume mehrerer Personen stehen, können für die einzelnen Miteigentumsanteile, ferner bei materiell geteilten Häusern für die einzelnen Hausanteile, sofern sie nicht als abgesonderte Grundbuchkörper behandelt werden (§ 5, Absatz 4), abgesonderte Eigentums- und Lastenblätter eröffnet werden, wenn dadurch die Übersicht erleichtert wird.

§ 13. Die Form der Urkundensammlung, die zu führenden Verzeichnisse sowie die sonstigen Vormerke und Behelfe werden vom Bundesminister für Justiz bestimmt.

§ 68. (1) Die Grundbuchseinlagen, welche landtäfliche Liegenschaften enthalten, bilden zusammen ein Hauptbuch. Landtäfliche Liegenschaften sind diejenigen Liegenschaften, die zur Zeit des Inkrafttretens dieses Gesetzes in der Landtafel eingetragen sind.

Verweise direkt im Paragraphen stehen anstelle von Definitionen. Es wird dann die dort zu findende Definition verwendet. Es handelt sich dann bei der Definition nicht um eine genauere Spezifizierung wie im Fall der Anmerkung, sondern um die generelle Definition, die im vorliegenden Gesetz verwendet wird ohne sie durch eine andere Definition zu ersetzen oder zu erweitern. In den §§ 1070, 1073 und 1095 ABGB werden beispielsweise Wiederkaufsrecht, Vorkaufsrecht und Bestandrecht definiert.

§ 1070. Der Vorbehalt des Wiederkaufes findet nur bei unbeweglichen Sachen statt und gebührt dem Verkäufer nur für seine Lebenszeit. Er kann sein Recht weder auf die Erben noch auf einen anderen übertragen. Ist das Recht in die öffentlichen Bücher einverleibt, so kann die Sache auch einem Dritten abgefordert werden und dieser wird nach Beschaffenheit seines redlichen oder unredlichen Besitzes behandelt.

§ 1073. Das Vorkaufsrecht ist in der Regel ein persönliches Recht. In Rücksicht auf unbewegliche Güter kann es durch Eintragung in die öffentlichen Bücher in ein dingliches verwandelt werden.

§ 1095. Wenn ein Bestandvertrag in die öffentlichen Bücher eingetragen ist; so ist das Recht des Bestandnehmers als ein dingliches Recht zu betrachten, welches sich auch der nachfolgende Besitzer auf die noch übrige Zeit gefallen lassen muß.

Bei Algebren sind solche Querverweise im Allgemeinen nicht notwendig. Algebren verwenden Definitionen von anderen Algebren indem sie angeben, welche anderen Algebren verwendet werden. Daher ist bei Verwendung der Definition eine Angabe der Quelle nicht mehr notwendig. Zusätzlich wird bei Algebren eine vollständige Kenntnis des erzeugten algebraischen Systems vorausgesetzt, d.h. jede Definition ist bekannt und kann verwendet werden.

Trotzdem gibt es auch bei algebraischen Modellen ein ähnliches Konzept. Wenn Bezeichnungen mehrmals vorkommen und unterschiedliche Bedeutung haben, muss man mit Modulen arbeiten. Dabei werden die einzelnen Teile des Modells gekapselt. Jedes Modul hat einen eigenen ‚Name Space‘. Somit kann in allen Modulen eine Operation mit gleichem Namen vorkommen. Will man dann auf eine bestimmte Definition der Operation zugreifen, gibt man an, auf welches Modul (auf welchen ‚Name Space‘) man zugreifen will. Wenn man nun das Modell eines Gesetzes als ein Modul ansieht, entspricht ein solcher Zugriff auf ein anderes Modul genau dem Verweis innerhalb eines Gesetzes.

4.2 Unterschiede

Zwischen Gesetzestexten und Algebren gibt es auch Unterschiede. Gesetzestexte versuchen zu beschreiben, wie das Zusammenleben der Menschen funktionieren sollte. Sie definieren Regeln für das Zusammenleben und versuchen die Menschen zur Beachtung dieser Regeln zu bewegen. Die Umsetzung der Regeln erfolgt somit in der realen Welt und ist nicht Teil des Gesetzes. Algebraische Modelle hingegen erstellen ein mathematisches Regelwerk, welches nur innerhalb der Mathematik gültig ist. Dort ist es dann allerdings auch überprüfbar.

4.2.1 Unscharfe Definitionen

Gesetze enthalten oft sehr unscharfe Definitionen. Dafür gibt es zwei Ursachen:

- *Vermeidung von festgelegten Werten:* Gesetze schaffen Definitionen, die über längere Zeiträume gültig sind. Wenn man nun Aussagen über den Wert eines Objektes trifft, steht man vor dem Problem, dass sich der Wert im Laufe der Zeit ändert. Das betrifft nicht nur den zahlenmäßigen Wert, sondern auch die Relation, in welcher der Wert des Objektes zum Wert anderer Objekte steht, also den relativen Wert. Eine Lösung dieses Problems ist die Angabe einer unscharfen Definition wie beispielsweise der Ausdruck ‚geringfügige Grundbuchssachen‘ im GBG55.
- *Ermessensspielraum:* Ein Gesetz kann nicht für jede mögliche Situation eine Definition enthalten. Oft ist es notwendig nach dem Sinn des Gesetzes zu fragen und entsprechend zu handeln. Das ist aber nur dann möglich, wenn das Gesetz eine gewisse Variationsbreite für Reaktionen zulässt. Ein Beispiel dafür ist das Strafausmaß bei Gesetzesübertretungen. Hier wird immer eine untere und eine obere Schranke und kein einzelner Wert angegeben. Dadurch hat das Gericht die Möglichkeit, das Strafausmaß an die tatsächliche Begebenheit anzupassen und Umstände in die Beurteilung einfließen zu lassen, die im Gesetz nicht behandelt werden. Dadurch wird einerseits der Umfang des Gesetzes reduziert und andererseits die Umsetzung der gesetzlichen Regelungen erleichtert.

Das Grundbuchgesetz hält den Ermessensspielraum mit Absicht klein, da es Privatrechte sichern soll. Somit müssen alle dinglichen Rechte und Lasten eingetragen werden. Ermessensspielraum für den Richter besteht hauptsächlich beim Akzeptieren von Urkunden. Hier muss der Grundbuchrichter entscheiden, ob eine Urkunde glaubhaft ist oder nicht. Ist eine Urkunde allerdings akzeptiert, so gibt es genaue Regeln, wie der Vorgang der bücherlichen Eintragung zu erfolgen hat.

4.2.2 Organisatorische Inhalte

Gesetze sind als Teil der Organisation des Staates Veränderungen unterworfen. Ein problemloser Übergang von einer Regelung auf eine neue Regelung benötigt Bestimmungen, wie dieser Übergang zu erfolgen hat. Diese Regelungen bestimmen das Datum, ab dem das Gesetz in Kraft ist, die Behörde, die für die Umsetzung verantwortlich ist und Angaben über Gesetze, die durch dieses Gesetz ersetzt werden. Diese Regelungen, die im Abschnitt ‚Übergangsregelungen‘ zusammengefasst sind, kann man als organisatorische Inhalte bezeichnen. Algebren benötigen keine organisatorischen Inhalte. Eine Algebra ist entweder vorhanden oder nicht, sie hat kein Datum, an dem ihre Gültigkeit beginnt. Zusätzlich benötigt eine Algebra auch niemanden, der

sich um die Umsetzung kümmert. Daher gibt es in einer Algebra auch keine Möglichkeit, solche Inhalte darzustellen.

4.2.2.1 Zeitlicher Aspekt

Rechtsordnungen ändern sich im Laufe der Zeit. Jede Novellierung und jedes neue Gesetz verändert einen Teil der Rechtsordnung. Die Änderungen erfolgen nicht kontinuierlich, sondern treten immer zu einem bestimmten Zeitpunkt ein. Dieser Zeitpunkt ist das Datum des Inkrafttretens und wird bei der Veröffentlichung des Gesetzes angegeben. Oftmals ersetzen Gesetzestexte ältere Versionen desselben Gesetzes oder einzelne Abschnitte anderer Gesetze. Dieser Umstand muss im Gesetz angemerkt werden. Das Grundbuchsgesetz definiert den zeitlichen Aspekt in § 137, der folgendermaßen lautet:

§ 137. (1) Dieses Bundesgesetz tritt drei Monate nach seiner Kundmachung in Kraft.

(2) Gleichzeitig treten außer Kraft:

1. das Gesetz vom 25. Juli 1871, RGBl. Nr. 95, über die Einführung eines Allgemeinen Grundbuchsgesetzes;

2. die §§ 1 und 3 des Gesetzes vom 4. Juni 1882, RGBl. Nr. 67, enthaltend Bestimmungen über die Entbehrlichkeit der Legalisierung gewisser Unterschriften auf Tabularurkunden und über Erleichterungen des Beweises der Identität einer Person bei Legalisierungen und anderen Beurkundungen;

3. ...

4. ...

5. die §§ 37 erster und zweiter Satz, 38 bis 41, 44 bis 46, 47 Abs. 1, 48 bis 50 sowie die Worte „des § 30 des Allgemeinen Grundbuchsgesetzes und“ in § 51 Abs. 2 der Kaiserlichen Verordnung vom 19. März 1916, RGBl. Nr. 69, über die dritte Teilnovelle zum Allgemeinen Bürgerlichen Gesetzbuch;

6. ...

(3) Andere Vorschriften grundbuchsrechtlichen Inhaltes bleiben unberührt.

Die erste Ziffer des Paragraphen definiert in allgemeiner Form das Datum des Inkrafttretens. Die allgemeine Form der Aussage hat den Vorteil, dass kein spezielles Datum eingesetzt ist, sondern nur eine Abhängigkeit zwischen Datum der Kundmachung und Datum des In-

krafttretens angegeben wird. Da das Datum der Kundmachung sich durch die Fertigstellung und Akzeptanz des Gesetzes und durch die Struktur des Kundmachungsorgans automatisch ergibt.

Die zweite Ziffer definiert, welche anderen Bestimmungen außer Kraft treten. Beispielhaft sind einige der insgesamt 11 Punkte herausgenommen. Der erste Punkt betrifft ein Gesetz, das als ganzes aus der Rechtsordnung genommen wird. Es handelt sich dabei um das Gesetz über die Einführung des Allgemeinen Grundbuchgesetzes. Da dieses Gesetz nach der erfolgten Einführung keine Bedeutung mehr hat, kann es außer Kraft gesetzt werden. Der zweite Punkt betrifft zwei Paragraphen eines anderen Gesetzes. Der fünfte Punkt betrifft einzelne Punkte und sogar Sätze bestimmter Paragraphen.

Man kann also sagen, dass Gesetze in unterschiedlichem Umfang außer Kraft gesetzt werden können. Prinzipiell gibt es drei Möglichkeiten:

- *Gesetz*: Hier wird ein ganzes Gesetz mit allen seinen Bestimmungen außer Kraft gesetzt. Das passiert meistens dann, wenn ein Gesetz durch ein anderes ersetzt wird.
- *Paragraph*: Hier werden ein oder mehrere Paragraphen außer Kraft gesetzt. Die Paragraphen müssen dabei genau (mit Angabe von Nummer des Paragraphen, Name, Kundmachungsdatum und -Organ des entsprechenden Gesetzes) aufgelistet werden.
- *Teil eines Paragraphen*: Es können ganze Absätze von Paragraphen, einzelne Bereiche bei Aufzählungen oder einzelne Sätze aus Paragraphen herausgenommen werden.

Diese Änderungen sind eine Folge des Zeitablaufes, da Gesetze an neue Gegebenheiten angepasst werden müssen. In einem algebraischen Modell hat man üblicherweise keinen Zeitablauf wie er in der Realität vorkommt. Eine Algebra beschreibt üblicherweise den Zustand zu einem bestimmten Zeitpunkt. Man benötigt also die Zeit als Teil des Modells, wenn man solche Abfolgen modellieren will. Dann ist es in Abhängigkeit von der Modellzeit möglich, unterschiedliche Regelungen zu verwenden:

```
data Zeitpunkt = Datum Integer Integer Integer
zeitablauf :: Zeitpunkt -> (a -> a)
zeitablauf zp f | zp < (Datum 11 6 1955) = alteRegeln f
                | zp >= (Datum 11 6 1955) = neueRegeln f
```

Die Funktion `zeitablauf` unterscheidet, ob das alte oder das neue Gesetz anzuwenden ist. Dazu wird das aktuelle Datum als Parameter verwendet. Da das GBG55 am 11. Juni 1955 in

Kraft getreten ist, wird das aktuelle Datum mit diesem Datum verglichen und entsprechend jeweils die alten oder die neuen Regeln verwendet.

4.2.2.2 Umsetzung

Bei Gesetzen ist es notwendig, sie in der Realität umzusetzen. Gesetze dienen dem Zusammenleben der Menschen und haben keine Daseinsberechtigung, wenn sie nur auf dem Papier existieren und in der Realität ignoriert werden. Daher sind Organisationen notwendig, die auf die Einhaltung der Gesetze achten. Mit der Vollziehung des Grundbuchgesetzes ist beispielsweise (bis auf kleine Ausnahmen die explizit angegeben sind) das Bundesministerium für Justiz betraut. Diese Angabe kann in § 140 GBG55 gefunden werden.

Algebren sind eine Möglichkeit der Modellierung der Realität. Da diese Modelle zwar die Realität widerspiegeln sollen, nicht jedoch in die Realität umgesetzt werden müssen, ist es nicht notwendig, Organisationen mit der Umsetzung zu betreuen. Deshalb enthalten Algebren keine Angaben zur Umsetzung in der Praxis, da die Verwendung von Algebren auf algebraische Modelle beschränkt bleibt.

4.2.3 Möglichkeit von Widersprüchen

Gesetze können sich gegenseitig widersprechen ohne dass dies sofort auffällt. Es könnte beispielsweise an einer Stelle stehen, dass aus einem Umstand A eine Rechtsfolge B folgt und an anderer Stelle könnte stehen dass aus Umstand A eine Rechtsfolge C entsteht. Wenn nun B und C nicht identisch sind, liegt ein Widerspruch vor. Solche Widersprüche fallen erst dann auf, wenn ein realer Fall untersucht wird, da nur in diesem Fall sämtliche Aspekte des rechtlichen Problems untersucht werden. Bei einer solchen Untersuchung werden auch die beiden unterschiedlichen Rechtsfolgen zu Tage treten. Solange eine solche Untersuchung aber nicht stattfindet, spielen Widersprüche im Gesetzssystem keine Rolle. Widersprüche im Bereich des Leibeigentums beispielsweise, spielen heutzutage keine Rolle, da es das Leibeigentum selbst nicht mehr gibt. Sollten also in den Regeln zum Leibeigentum Widersprüche vorhanden sein, so bleiben diese bestehen, bis alle Paragraphen über das Leibeigentum explizit gelöscht werden.

Bei Algebren sind keine Widersprüche zulässig. Eine Algebra ist eine mathematisch eindeutige Definition. Widersprüchliche Axiome verletzen jedoch die Eindeutigkeit. Wenn es für eine Situation A beispielsweise die Ergebnisse B und C gibt und $B \neq C$ gilt, so ist das ein mathematischer Widerspruch, da eine Operation nicht zwei unterschiedliche Ergebnisse haben darf. Daher ist es nicht möglich, widersprüchliche Gesetzesstellen in Algebren überzuführen.

Widersprüche müssen vor oder während der Modellierung gelöst werden. Erschwerend ist allerdings der Umstand, dass funktionale Programmiersprachen wie Haskell Widersprüche nicht automatisch finden können. Haskell löst das Problem beispielsweise so, dass immer die erste gefundene Definition verwendet wird. Ein Aufruf der nachfolgenden Operation `op` mit dem Parameter 5 liefert beispielsweise das Ergebnis 3, da diese Definitionszeile früher im Quellcode steht.

```
op :: Int -> Int
op a = 3
op a = a
```

4.3 Zusammenfassung

Sowohl Gesetzestexte als auch Algebren werden verwendet, um Modelle zu definieren. Dabei werden ähnliche Vorgangsweisen gewählt. In beiden Fällen wird das Gesamtproblem in kleine Teile zerlegt und diese Teilprobleme dann durch einzelne Definitionen beschrieben. Zusätzlich wird in beiden Fällen versucht, möglichst abstrakt zu bleiben, da nur so sichergestellt ist, dass man mit einem kleinen Regelwerk einen großen Bereich der Realität abdecken kann. Die Vorgangsweise ist also in groben Zügen identisch.

Andererseits hat es sich aber auch herausgestellt, dass es Bereiche gibt, in denen die Methoden nicht übereinstimmen. Ein Hauptunterschied ist die Umsetzung von Gesetzestexten in der Realität. Daher enthalten Gesetzestexte Teile, die keine Entsprechung bei Algebren haben. Das sind vor allem die organisatorischen Inhalte, die sich weniger auf den Inhalt des Gesetzes beziehen, sondern vielmehr die Umsetzung des Gesetzes steuern. Es gibt auch Vereinfachungen, die das Schreiben von Gesetzen erleichtern sollen. Die wichtigste Vereinfachung ist dabei der Umstand, dass die Konsistenz des Systems erst im Bedarfsfall geprüft wird, während diese Prüfung bei Algebren sofort und umfassend ausgeführt wird.

5 Wie formalisiert man Gesetze?

Im letzten Kapitel wurden die Ähnlichkeiten zwischen Gesetzestexten und Algebren gezeigt. Nun ist die Vorgangsweise für eine Umwandlung von Gesetzen in Algebren zu beschreiben. Zunächst müssen die Elemente des Modells beschrieben und die Struktur für das Modell aufgebaut werden. Danach werden die Operationen definiert und den Elementen zugeordnet. Zum Testen der Modellierung muss eine Realisierung des Modells erfolgen. Eine Diskussion zeigt abschließend die Grenzen der Modellierung.

5.1 Erste Schritte

Der erste Schritt ist, sich einen Überblick über das Gesetz zu verschaffen. Dabei werden die vorkommenden Elemente und ihre Zusammenhänge identifiziert. Man erkennt auch die vorausgesetzten übrigen Gesetze. Aus diesem Vorgang erhält man also zwei Ergebnisse:

- Die Objektstruktur bildet die Basis für die Definition von Datentypen und Klassen für die Algebren. Jedes dieser Objekte benötigt eine eigene Algebra. Es entsteht somit gleichzeitig eine Struktur der Algebren.
- Die gefundenen Gesetze zeigen die übrigen in das Modell einzubeziehenden Gesetze. Die Regelungen dieser Gesetze sind meist nur auszugsweise nötig. Auch hier erhält man eine Struktur von Algebren.

Ein Teil einer Objektstruktur wurde in Abbildung 1 (siehe Abschnitt 4.1.1) gezeigt. Diese Struktur muss im ersten Bearbeitungsschritt noch näher ausgearbeitet werden. Manche Elemente enthalten Einzelwerte, andere eine Liste von Werten. Ein Grundbuch enthält genau ein Hauptbuch und eine Urkundensammlung. Das Hauptbuch besteht aber aus vielen Grundbucheinlagen und die Urkundensammlung aus vielen Urkunden. Diese Zusammenhänge müssen geklärt werden, da sie einen starken Einfluss auf die Datentypen und später auf die Operationen haben. Zu späte Identifikation dieser Unterschiede kann ein Umarbeiten großer Teile des Modells notwendig machen. Daher ist eine frühzeitige Klärung dieser Fragen notwendig.

Das Grundbuch dient der Sicherung von dinglichen Rechten an Grund und Boden. Der Begriff des dinglichen Rechtes wird jedoch im Grundbuchgesetz nicht geklärt. Diese Definition muss dem ABGB entnommen werden. Dabei muss immer entschieden werden, wie weit Definitionen für das Modell notwendig sind beziehungsweise was weggelassen oder vereinfacht werden kann.

5.2 Beitrag der einzelnen Paragraphen

Der Gesetzestext liefert Informationen zum Modell in Form von Paragraphen. Der Inhalt jedes dieser Paragraphen muss auch im algebraischen Modell enthalten sein. Da ein algebraisches Modell aus Typparametern, Operationen und Axiomen (siehe 3.2.1) besteht, sollten auch nur diese Teile im resultierenden Modell vorkommen. Im folgenden wird genauer untersucht, in welcher Form diese Teile vorkommen können.

5.2.1 Typdefinitionen

Typparameter definieren die Teile, aus denen sich die Algebra zusammensetzt. Sie entsprechen also den abstrakten Begriffsdefinitionen des Gesetzestextes. Es gibt prinzipiell zwei Möglichkeiten, wie Begriffe zusammengesetzt werden können. Die erste Möglichkeit ist das Zusammenfassen von Elementen gleichen Typs. Die zweite Möglichkeit, nämlich Elemente unterschiedlichen Typs zusammenzufassen, gliedert sich allerdings wiederum in zwei Arten. Unterschiedliche Elemente können sich gegenseitig ausschließen oder gemeinsam vorkommen. Abbildung 2 zeigt die Arten der Zusammenfassung.

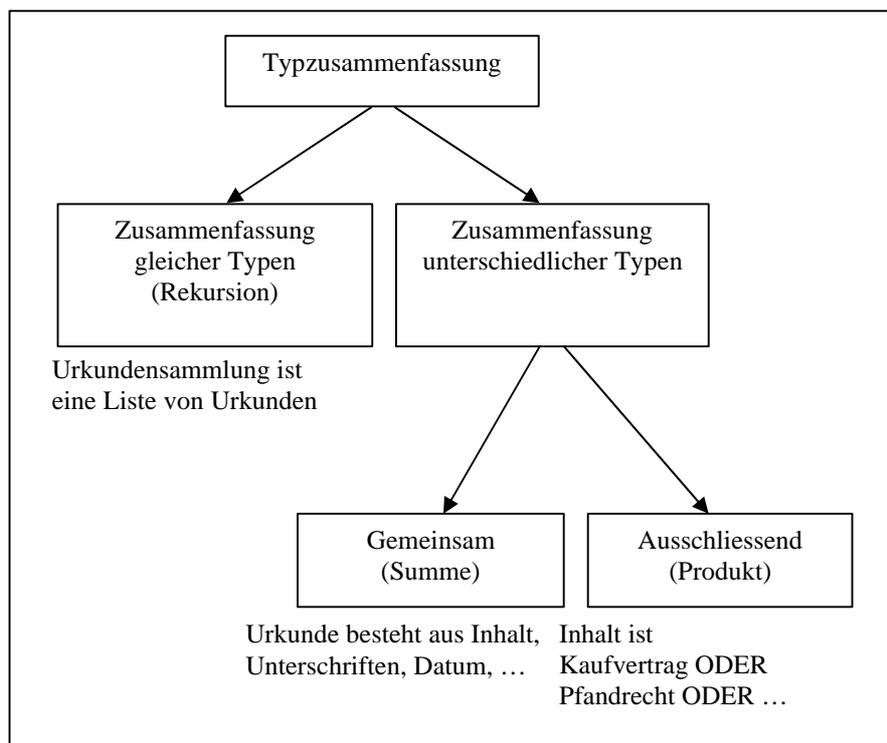


Abbildung 2: Arten der Typzusammenfassung

Die einfachste Art der Typdefinition ist die Zusammenfassung von Elementen gleichen Typs zu einem neuen Element. Die Urkundensammlung besteht aus Urkunden. Jede Urkunde enthält

bestimmte Daten wie Inhalt der Urkunde, Unterzeichner oder Datum des Einlangens im Grundbuch. Der Inhalt der Urkunde beschreibt ein bestimmtes Rechtsgeschäft, beispielsweise einen Kauf oder die Einräumung eines Pfandrechtes. In Haskell-Notation sieht eine solche Definition folgendermaßen aus:

```

type GstNr      = ...
type Person     = ...
type Unterschrift = ...
type Datum      = ...

data Inhalt     = Kaufvertrag GstNr Person Person |
                Pfandrecht  GstNr Person Summe
data Urkunde    = Urkunde Inhalt [Unterschrift] Datum
type Urkundensammlung = [Urkunde]

```

Die ersten vier Zeilen definieren Datentypen, die für die Beschreibung von Grundstücken, Personen, Unterschriften und Datumsangaben benötigt werden. Die nächste Zeile definiert den Datentyp `Inhalt`. Ein Datensatz dieses Typs kann entweder ein Kaufvertrag oder die Einräumung eines Pfandrechtes, niemals aber beides sein. Diese Art der Zusammenfassung heißt Produkt. Danach wird der Datentyp `Urkunde` definiert. Eine Urkunde besteht immer aus `Inhalt`, Unterschriften und ein Datum. Der Datentyp `Urkunde` fasst die einzelnen Teile so zusammen, dass alle Teile vorkommen. Eine Urkunde ist somit ein Summentyp. Schließlich besteht der Datentyp `Urkundensammlung` aus mehreren Urkunden. Eine Urkundensammlung enthält nur Datensätze vom Typ `Urkunde`. Es ist aber nicht angegeben, wie viele Datensätze die Urkundensammlung enthält. Eine solche Zusammenfassung wird in Haskell als Liste bezeichnet und ist mathematisch betrachtet eine Rekursion.

5.2.2 Operationen

Die Paragraphen liefern auch Informationen für die Definition von Operationen. Dabei sind drei Punkte wichtig. Zunächst muss man wissen, welche Operationen benötigt werden. Danach müssen die einzelnen Operationen Algebren zugeordnet werden. Schließlich ist noch klarzustellen, welche Parameter für die Operation notwendig sind.

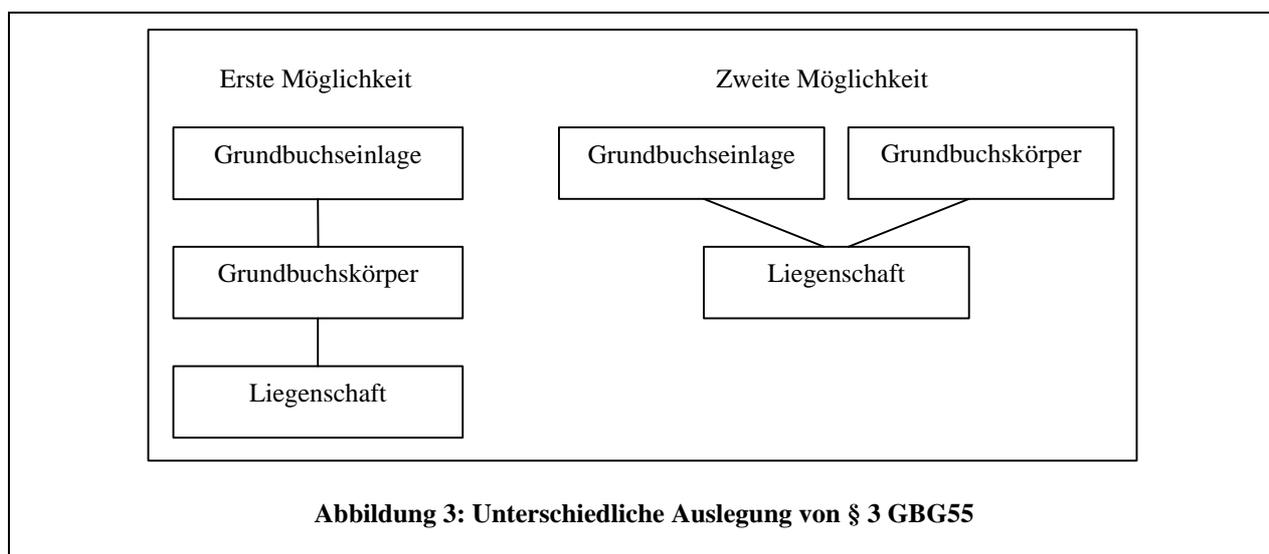
5.2.2.1 Angabe von notwendigen Operationen

Grundsätzlich gibt es zwei Arten, wie Operationen in Gesetzestexten vorkommen können. Eine Operation kann entweder direkt beschrieben sein oder implizit notwendig werden. Der erste Fall

tritt dann auf, wenn bei einem System Abläufe behandelt und dabei bestimmte Operationen beschrieben werden. Im zweiten Fall werden die Operationen nicht direkt erwähnt sondern es ergibt sich aus der Art der Aufgabenstellung, dass bestimmte zusätzliche Operationen notwendig sind. Als Beispiel für beide Fälle kann § 3 GBG55 herangezogen werden:

- § 3.** (1) *Jeder Grundbuchkörper ist als ein Ganzes zu behandeln.*
 (2) *Sein Umfang kann nur durch die grundbücherliche Ab- und Zuschreibung von einzelnen Liegenschaften oder Liegenschaftsteilen geändert werden.*
 (3) *Wenn alle in einer Grundbuchseinlage eingetragenen Liegenschaften abgeschrieben worden sind (§ 11) oder wenn sie aufgehört haben, ein Gegenstand des Grundbuches zu sein, ist die Einlage zu löschen.*

Grundsätzlich enthält dieser Paragraph Informationen über die Einordnung des Grundbuchkörpers in die Struktur des Grundbuches. Der Grundbuchkörper besteht aus Liegenschaften. Andererseits wird festgehalten, dass Liegenschaften zur Grundbuchseinlage gehören. Daraus ergeben sich zwei Möglichkeiten. Entweder ist der Grundbuchkörper ein Teil der Grundbuchseinlage oder die Grundbuchseinlage ist ein parallel zum Grundbuchkörper existierendes Konzept. Abbildung 3 zeigt den Unterschied zwischen den beiden Möglichkeiten. Aus einer anderen Quelle (§ 2 GBG55) ist ersichtlich, dass die erste Möglichkeit zutrifft.



§ 3 Abs. 1 kann für die Definition der Operationen vorerst vernachlässigt werden. Er dient nur dazu, die nachher definierten Operationen einer bestimmten Algebra (der Algebra ‚Grundbuchkörper‘) zuzuordnen. Das ist, wie im weiteren Verlauf gezeigt wird, leider nicht für den gesamten Paragraphen gültig.

§ 3 Abs. 2 sagt aus, dass der Umfang des Grundbuchskörpers nur durch bestimmte Operationen geändert werden kann. Was unter dem Begriff ‚Umfang des Grundbuchskörpers‘ zu verstehen ist, wird vorläufig nicht näher definiert. Diese Operationen heißen ‚grundbücherliche Abschreibung‘ und ‚grundbücherliche Zuschreibung‘. Des Weiteren ist ersichtlich, dass für diese Operationen Liegenschaften oder Liegenschaftsteile als Eingangsparameter benötigt werden. Somit ist auch eine indirekte Definition für ‚Umfang des Grundbuchskörpers‘ vorhanden, da der Grundbuchskörper nun als Menge von Liegenschaften aufgefasst werden kann.

§ 3 Abs. 3 definiert die Folge einer grundbücherliche Abschreibung aller in einer Grundbuchseinlage enthaltenen Liegenschaften. Die Folge ist das Löschen der Grundbuchseinlage. Problematisch ist hier der Wechsel von der Ebene des Grundbuchskörpers auf die Ebene der Grundbuchseinlage. Da dies zwei unterschiedliche Teile sind, gehört die Operation ‚Einlage löschen‘ nicht derselben Algebra an wie die beiden in § 3 Abs. 2 erwähnten Operationen.

Zusätzlich setzt § 3 Abs. 2 eine Operation voraus, die nicht erwähnt wird. Es wird von der Voraussetzung gesprochen, dass alle Liegenschaften abgeschrieben worden sind. Es ist also eine Operation notwendig, welche aussagt, ob der Grundbuchskörper Liegenschaften enthält oder nicht.

5.2.2.2 Zuordnung der Operationen zu Algebren

Die Operationen müssen den einzelnen Algebren zugeordnet werden. Wenn die Operation nur ein einzelnes Element behandelt ist die Zuordnung einfach. Die im letzten Kapitel erwähnte Operation zur Prüfung, ob ein Grundbuchskörper Liegenschaften enthält, gehört zur Algebra über Grundbuchskörper.

Es gibt aber auch Operationen die unterschiedliche Dinge miteinander verknüpfen. Ein Beispiel dafür wäre das Eintragen eines Rechtes ins Grundbuch. Dazu muss ein Dokument eingereicht werden. Somit sind mehrere Elemente beteiligt. Ausgangspunkt der Operation ist das Dokument und Resultat ist eine Eintragung in der Grundbuchseinlage. Dokument und Grundbuchseinlage bilden eigene Algebren. Daher kann die Operation beiden Teilen zugeordnet werden.

Eine sinnvolle Lösung des Problems ist das Aufspalten der Operation in Einzelteile. Operationen, bei denen dieses Zuordnungsproblem auftritt, beinhalten meist mehrere Teile. Im Fall der Generierung der Eintragung sind es folgende Arbeitsschritte:

- Auslesen der benötigten Daten aus dem Dokument

- Formatierung der Daten zu einer neuen Eintragung
- Hinzufügen der Eintragung zur Grundbuchseinlage

Die Schritte finden in unterschiedlichen Teilen des Modells, also in unterschiedlichen Algebren statt. Der erste Arbeitsschritt gehört zur Algebra der Dokumente, da es sich nur darum handelt, bestimmte Daten aus dem Dokument zu lesen. Der letzte Arbeitsschritt gehört zur Algebra der Grundbuchseinlage, da es sich darum handelt, neue Daten zu den Daten der Grundbuchseinlage hinzuzufügen. Der mittlere Teil gehört weder zum Dokument noch zur Grundbuchseinlage. Somit benötigt man eine Algebra für Eintragungen.

5.2.2.3 Festlegung der Parameter für Operationen

Die notwendigen Operationen benötigen Parameter. Das kann explizit ausgedrückt sein. Ein Beispiel dafür ist die Eintragung eines Rechtes in das Grundbuch. Die Operation gehört zur Algebra ‚Grundbuch‘, benötigt aber auch eine Urkunde, die das einzutragende Recht dokumentiert. Die Urkunde beeinflusst das Ergebnis der Operation ‚Eintragung‘ und muss somit der Funktion als Parameter übergeben werden.

Oft sind die notwendigen Parameter nicht direkt ersichtlich und lassen sich nur aus der Funktionalität ableiten. Wenn beispielsweise eine Operation ein eingetragenes Datum mit dem aktuellen Datum vergleichen soll, muss bei funktionaler Modellierung das aktuelle Datum als Parameter übergeben werden. Ein weiteres Beispiel ist die Festlegung der Grundbuchseinlage, zu der ein eingereichtes Dokument hinzugefügt werden muss. Im Gesetz wird meist angenommen, dass diese Selektierung bereits durchgeführt wurde. Nur § 85 erwähnt die Bezeichnungen von Grundbuchseinlagen (§ 85 Abs. 1). Diese Bezeichnung ist bei jedem Antrag an das Grundbuch anzugeben (§ 85 Abs. 2). Somit müssen Operationen auf einer höheren Ebene als der Grundbuchseinlage (Hauptbuch, Grundbuch) diese Bezeichnung als Parameter mitführen.

Um die Parameter aller Operationen vollständig zu bestimmen, ist eine komplette Durchsicht des Gesetzestextes notwendig. Parameter, die bei diesem Schritt übersehen werden, können im weiteren Verlauf der Modellierung noch hinzugefügt werden. Der Aufwand wächst allerdings wegen der notwendig werdenden Änderungen bei bereits definierten Axiomen. Da ein Axiom die Funktionalität durch Aufruf anderer Operationen definiert, bewirkt eine Änderung der Parameter eine Änderung des Aufrufes. Das kann sogar dazu führen, dass die Parameter der aufrufenden Operation ebenfalls ergänzt werden müssen.

5.2.3 Axiome

Zu weiten Bereichen des Modells gibt es im Gesetz Beschreibungen, wie der Ablauf aussehen muss. Wie in Abschnitt 3.2.1 gezeigt, können die in Algebren vorhandenen Funktionen in Konstruktoren, Observer und abgeleitete Funktionen unterteilt werden. Die abgeleiteten Funktionen sind dabei die Operationen, die durch Kombinationen von Konstruktoren, Observern und anderen Operationen beschrieben werden können. Ein Beispiel dafür ist die automatische Lösung einer abgelaufenen Rangordnung (§ 57 Abs.2 GBG55). Eine Rangordnung ist ein ‚reservierter Platz‘ für eine Eintragung. Die Wertigkeit der Eintragungen im Grundbuch ist nach ihrem ‚Rang‘ geordnet, also (mit wenigen Ausnahmen) nach der Reihenfolge der Eintragung. Eine Rangordnung ist ein Platzhalter für eine Eintragung, die zu einem späteren Zeitpunkt erfolgen wird. Nach Ablauf eines Jahres erlischt allerdings die Gültigkeit der Rangordnung und es erfolgt eine automatische Löschung. Vor der Löschung erfolgt ein Vergleich zwischen zwei Datumsangaben. Ist das aktuelle Datum nach dem Ablaufdatum der Rangordnung, so wird die Rangordnung gelöscht. Dabei ist für die Beschreibung nicht wichtig, wie eine Rangordnung aussieht oder was sie bewirkt. Somit können die Beschreibungen aus den Gesetzen in Form von Axiomen in das Modell eingebunden werden.

Abbildung 4 zeigt, wie das Beispiel der automatischen Löschung der Rangordnung umgesetzt werden kann. Ausgangspunkt ist eine Liste von Grundbucheintragungen. Für jede Eintragung wird der in Abbildung 4 gezeigte Ablauf abgearbeitet. Zunächst überprüft die Operation, ob die Eintragung eine Rangordnung ist. Dann wird weiter überprüft, ob der Gültigkeitsbereich überschritten wurde. Ist das der Fall, so wird die Rangordnung gelöscht. In jedem anderen Fall wird die Operation abgebrochen und die Eintragung bleibt unverändert.

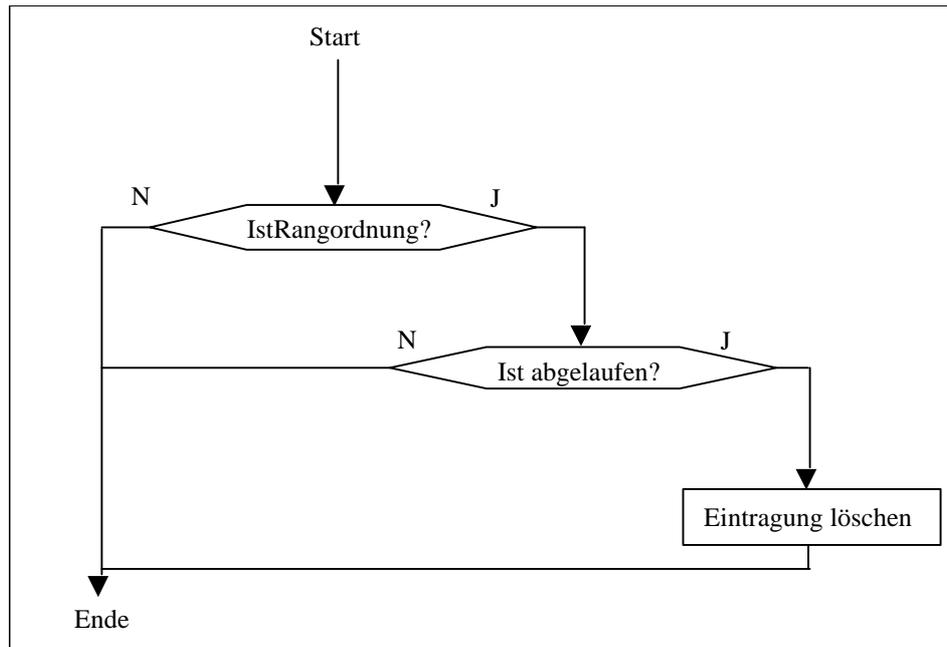


Abbildung 4: Umsetzung einer Ablaufbeschreibung

5.3 Bauen einer Modellrealisierung für Testzwecke

Um die Axiome auf Korrektheit testen zu können ist eine Realisierung des Modells notwendig. Diese Realisierung muss einfach sein. Dann ist es nämlich möglich, alle denkbaren Abfolgen von Operationen als Testdatensätze zu speichern und die Axiome auf korrektes Verhalten zu testen.

Explizite Datentypen beschreiben, wie die Datenstruktur des Modells aufgebaut ist. Zu einem großen Teil ist die Struktur der Datentypen bereits durch die Klassenstruktur und die Klassenkonstruktoren vorgegeben. Die Klassenkonstruktoren definieren, aus welchen Teilen sich die Klasse zusammensetzt. Es kann somit nur mehr gewählt werden in welcher Reihenfolge die Einzelteile aneinandergefügt werden. Die Klassenstruktur gibt an, in welcher Reihenfolge die Datentypen definiert werden. Eine Klassenstruktur wie in Abbildung 5 sagt beispielsweise, dass es einen Datentyp für Koordinaten geben muss. Zusätzlich gibt es einen Datentyp für Punkte, der den Datentyp Koordinaten verwendet, um die Position des Punktes zu speichern. Die Punkte werden dann wieder verwendet um Linien bzw. Polygone zu definieren.

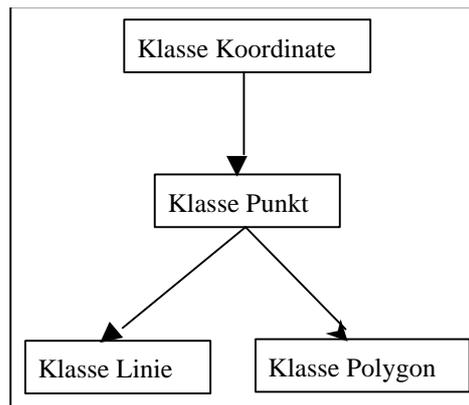


Abbildung 5: Beispiel Klassenstruktur für Geometrie

Die Klassen müssen dann für die entsprechenden Datentypen instanziiert werden. Die Instanzierung besteht nur darin, die Klassenkonstruktoren und die Observer für die Datentypen zu definieren, da es für alle übrigen Operationen Axiome gibt. Die Definition dieser Operationen kann erst jetzt geschehen, da die endgültigen Datentypen erst zum Zeitpunkt der Instanzierung bekannt sind.

Testdatensätze sollen alle möglichen Fälle enthalten. Ist die Anzahl der Möglichkeiten zu groß, um sie vollständig mit Testdatensätzen abzudecken, so sollten zumindest die wichtigsten Fälle und die häufigsten Ausnahmen vertreten sein. Die Aufgabe der Testdatensätze ist es, die korrekte Arbeitsweise der Operationen zu demonstrieren.

Die Tests müssen zwei Punkte nachweisen. Einerseits muss der Code korrekt sein. Das Programm muss also genau das tun, was das Gesetz vorschreibt. Dieser Teil des Tests prüft also, ob das Gesetz korrekt umgesetzt wurde. Andererseits muss aber auch überprüft werden, ob das Modell Lücken aufweist. Lücken können dabei entweder durch ein fehlerhaftes Gesetz entstehen, oder weil eine Verbindung zu einem anderen Gesetz übersehen wurde. Im ersten Fall muss das Gesetz verbessert, im zweiten das Modell ergänzt werden.

5.4 Grenzen der Modellierung

Probleme bei der Modellierung treten an den Schnittstellen zwischen Gesetz und realer Welt auf. Hier gibt es vor allem zwei Bereiche, in denen große Probleme bei der Modellierung entstehen. Einerseits sind das die organisatorischen Inhalte der Gesetze, andererseits gibt es aber auch Vorgänge, die Interaktion notwendig machen. Beide Bereiche können nur sehr schwer in das Modell eingebaut werden.

5.4.1 Organisatorische Inhalte

Wie in Abschnitt 4.2.2 gezeigt sind organisatorische Inhalte im Gesetz zwar notwendig, in einer Algebra allerdings nicht vorgesehen. Regelungen für die Umsetzung des Gesetzes können nicht abgebildet werden. Da die Organisationen, die mit der Umsetzung betraut wurden, nicht Teil des Modells sind, können diese Regelungen nicht eingebunden werden. Man kann versuchen, die Organisationen mitzumodellieren, steht dann aber vor einem zeitlichen Problem (siehe Abschnitt 4.2.2.1), wenn sich die Zuständigkeit ändert.

Eine Änderung der Zuständigkeit kann in zwei Fällen auftreten. Manchmal geht die Zuständigkeit von einer Organisation auf eine andere Organisation über, da sich die Aufgabenstellung geändert hat und somit die Aufgabe besser in die neue Organisation passt. Ein Beispiel dafür ist der Kataster, der 1919 aus dem Ressort des Staatsamtes für Finanzen losgelöst und dem Staatsamt für Handel und Verkehr, Industrie und Bauten einverleibt wurde (Lego 1968). Es kann aber auch vorkommen, dass eine Organisation einer anderen Organisation einverleibt wird und somit alle Agenden auf die neue Organisation übergehen.

Organisatorische Aspekte können im Modell nur in Form von Kommentaren angefügt werden. Beispielsweise kann man im Kopf des Modells alle Aspekte auflisten, die nicht berücksichtigt werden konnten.

Die einzigen organisatorischen Inhalte, die in das Modell übernommen werden können, sind Verweise auf andere Gesetze. Hier kommt es allerdings immer auf die Relevanz der betreffenden Gesetzesstelle für das zu schaffende Modell an. In Abschnitt 4.1.5 wurde als Beispiel der § 9 BGB zitiert, der auf das ABGB verweist. Die Verweise dienen hier nur dazu, Begriffe (Wiederkaufsrecht, Vorkaufsrecht, Bestandrecht) näher zu erläutern. Da dies nur für die Auswirkungen des Rechtes in der realen Welt von Bedeutung ist und keinen Einfluss auf das Modell hat, muss es im Modell nicht berücksichtigt werden. Anders liegt der Fall bei § 126. Hier wird für die Entscheidung des Rekursgerichtes auf § 13 AußStrG (Außerstreitgesetz) verwiesen in dem definiert wird, wann ein Revisionsrekurs erlaubt ist. In solchen Fällen muss man während der Modellierung entscheiden, ob und wie weit man diese Gesetzesstelle in das Modell einbaut. Das gilt allerdings nur für Verweise auf andere Gesetze. Verweise innerhalb eines Gesetzes müssen nicht modelliert werden, da sie hauptsächlich dem Leser als Orientierungshilfe dienen sollen und für eine Algebra nicht notwendig sind.

5.4.2 Interaktion

Interaktion tritt immer auf, wenn eine Auswahl aus mehreren Möglichkeiten notwendig wird. Ein Beispiel dafür ist ein Gerichtsbeschluss, der entweder beeinsprucht oder akzeptiert werden kann. Solche Entscheidungen sind üblicherweise innerhalb einer umfangreichen Operation notwendig. Daher muss die Operation unterbrochen und die Auswahl ermöglicht werden. Algebraische Modelle sehen solche Unterbrechungen nicht vor, da die Auswahl das Ergebnis der Operation beeinflusst. Die Operationen algebraischer Modelle haben aber eindeutig definierte Ergebnisse und können somit nicht von etwas abhängig gemacht werden, das kein Parameter der Operation ist.

Prinzipiell kann das Problem auf zwei Arten gelöst werden. Zunächst kann man die notwendigen Entscheidungen im Modell als Parameter mit übergeben. Der Nachteil dieser Lösung ist, dass Werte für alle Entscheidungen angegeben werden müssen, die notwendig werden können. Das bläht die Parameter der Operationen stark auf und reduziert dadurch die Übersichtlichkeit des Modells. Außerdem widerspricht es der praktischen Anwendung, da üblicherweise Entscheidungen nur dann eingeholt werden, wenn sie notwendig sind. Andernfalls würde man nämlich Entscheidungen treffen, von denen man nicht sicher ist, ob sie überhaupt getroffen werden müssen. Die zweite Art das Problem zu lösen ist das Aufteilen der Operationen in mehrere Teile, sodass alle möglichen Entscheidungspunkte zwischen diesen Operationen stattfinden. Somit fällt die Entscheidung selbst aus dem Modell heraus, da die jeweils notwendige Operation manuell aufgerufen werden kann. Der Nachteil dieser Lösung ist das Fehlen einer kompletten Operation, die einen Gesamtprozess auf einmal ablaufen lässt. Stattdessen stehen mehrere Operationen zur Verfügung, die nacheinander aufgerufen werden, wobei im Bedarfsfall zusätzliche Operationen eingeschoben werden. Beide Methoden bieten zwar einen Lösungsansatz, haben aber immer noch Probleme. Im ersten Fall ist es die Lesbarkeit des Modells, im zweiten Fall die Unmöglichkeit, eine Gesamtoperation zu definieren die immer verwendet werden kann.

5.4.3 Ermessensspielraum

Oft lassen Gesetze einen Spielraum für den Anwender. Dieser Spielraum kann sich in Form von Schranken für die Rechtsfolge äußern, oder so, dass keinerlei Angaben gemacht werden. Ein Beispiel für den ersten Fall ist §127 StGB, der die Rechtsfolge für Diebstahl definiert:

§ 127. Wer eine fremde bewegliche Sache einem anderen mit dem Vorsatz wegnimmt, sich oder einen Dritten durch deren Zueignung unrechtmäßig zu

bereichern, ist mit Freiheitsstrafe bis zu sechs Monaten oder mit Geldstrafe bis zu 360 Tagessätzen zu bestrafen.

Der zweite Fall tritt in § 43 GBG55 auf. Hier wird die Frist für die Rechtfertigung einer Vormerkung behandelt. Es wird dabei keinerlei Angabe über die Länge der Frist gemacht.

§ 43. (1) Die Frist zur Erhebung der Rechtfertigungsklage ist in dem Vormerknungsbeschuß auszudrücken. Sie kann aus erheblichen Gründen verlängert werden.

Der erste Fall kann somit erfasst werden, indem das Ergebnis der Operation ein Intervall ist. Im zweiten Fall ist das nicht möglich. Es gibt zwei Möglichkeiten zur Lösung:

- **Interaktion:** Durch Unterbrechung des Berechnungsvorganges und Abfrage eines Wertes kann der Entscheidungsprozess modelliert werden. Das ist auch bei angegebenen Schranken möglich.
- **Standardwert:** Es wird ein Standardwert angegeben und im Modell so getan, als ob dieser Wert auf korrekte Art bestimmt worden wäre. Es entstehen dadurch keine Nachteile für das Modell, da die Bestimmung des Wertes in der realen Welt, also außerhalb des modellierten Systems geschieht.

Bei Verwendung einer funktionalen Programmiersprache für die Modellierung ist die Methode des Standardwertes vorzuziehen. Interaktionen sind in funktionalen Programmiersprachen nur schwer umzusetzen. Interaktion verletzt nämlich das Prinzip, dass das Ergebnis einer Operation nur von den Parametern abhängig sein darf. Somit ist der Standardwert die einfachere Lösung. Dabei wird allerdings die Entscheidung über den zu verwendenden Wert ausgeklammert. Das kann Probleme verursachen, wenn man das Modell als Kern einer Software verwenden will, die das Gesetz umsetzt. Daher müssen diese Fälle markiert werden um spätere Fehler auszuschließen.

Ein weiteres Beispiel für Ermessensspielraum ist § 130. Er regelt die Löschung von unzulässigen Eintragungen und lautet:

§ 130. Ergibt sich aus einer Eintragung, daß ihr Inhalt nach dem Gesetz nicht Gegenstand einer grundbücherlichen Eintragung sein kann, so ist sie von Amts wegen als unzulässig zu löschen. Die Vorschriften des ersten bis dritten und fünften Hauptstückes, insbesondere über die Verständigung der Beteiligten und den Rekurs, sind entsprechend anzuwenden.

Diese Regelung ist zwar eindeutig, verursacht aber bei der Modellierung ein Problem. Für die Bestimmung der Fälle in denen dieser Paragraph Anwendung findet benötigt man eine Operation, welche die Legalität einer Eintragung beurteilt. Eintragungen, welche nicht im Gesetz vorgesehen sind, scheiden von vornherein aus, da das Grundbuch nur genau definierte Eintragungen zulässt. Eintragungen aber, die aus anderen Gründen illegal sind, können nicht entdeckt werden. Es kann sich beispielsweise um eine Dienstbarkeit handeln, bei welcher die Beschreibung unzulässige Bestimmungen enthält. Da es für die Beurteilung dieses Umstandes aber keine detaillierten Regeln gibt, kann die Beurteilung auch nicht modelliert werden. Hier muss angenommen werden, dass alle Eintragungen zulässig sind oder für jede Eintragung wird in einem speziellen Feld definiert, ob die zulässig ist. Auch hier wird dabei ein in der realen Welt passierender Vorgang definiert.

5.5 Zusammenfassung

Dieses Kapitel hat anhand ausgewählter Beispiele die Übersetzung eines Gesetzes in eine Algebra gezeigt. Der erste Schritt war die Bestimmung der vorkommenden Elemente und ihrer Zusammenhänge, sowie eine grobe Abgrenzung des zu modellierenden Rechtsbereiches. In einem zweiten Schritt wurden die Paragraphen in Teile des Modells übersetzt, wodurch Klassen, Operationen und Axiome entstanden und Typen in allgemeiner Form definiert wurden. Parallel dazu wurde eine Realisierung des Modells gebaut, mit deren Hilfe die Axiome auf ihre Korrektheit überprüft werden konnten. Abschließend wurden noch Aspekte diskutiert, bei denen die Modellierung an ihre Grenzen stößt. Hier sind es speziell die organisatorischen Inhalte von Gesetzen, die sich einer Modellierung widersetzen.

6 Formalisierung des Allgemeinen Grundbuchsgesetzes 1955

Die bisherigen Kapitel haben Möglichkeiten aufgezeigt, Gesetze in algebraische Modelle zu übersetzen. Dieses Kapitel prüft anhand eines realen Beispiels, inwieweit dies tatsächlich möglich ist. Zusätzlich soll untersucht werden, ob die in Abschnitt 5.4 gezeigten Grenzen der Modellierung tatsächlich gegeben sind. Als Beispiel wurde das österreichische Allgemeine Grundbuchsgesetz verwendet.

6.1 Aus anderen Gesetzen übernommene Strukturen

Wie in Kapitel 5.1 beschrieben, müssen manche Teile des Modells von anderen Gesetzen übernommen werden. Die Voraussetzungen für das Grundbuchsgesetz sind vor allem im ABGB zu finden, das ABGB definiert die dinglichen Rechte (§§ 307 – 308, § 309 ff). Ebenfalls im ABGB steht der Grund für die Existenz des Grundbuches. § 431 ABGB behandelt den Eigentumserwerb durch Übergabe für unbewegliche Sachen und Bauwerke und lautet:

Zur Übertragung des Eigentumes unbeweglicher Sachen muß das Erwerbungsgeschäft in die dazu bestimmten öffentlichen Bücher eingetragen werden. Diese Eintragung nennt man Einverleibung (Intabulation).

Die öffentlichen Bücher sind dabei nicht nur die Grundbücher, sondern auch das Eisenbahnbuch, das Bergbuch und das Wasserbuch. Alle diese Bücher haben eigene gesetzliche Bestimmungen und dienen dazu, unbewegliche Sachen bestimmter Kategorien zu verwalten. Die vorliegende Arbeit beschränkt sich auf das umfangreichste Buch, nämlich das Grundbuch.

Zwei Teilaspekte sind notwendig, bevor zum nächsten Teil des Modells übergegangen werden kann. Zunächst ist zu klären, welche Daten gespeichert werden müssen und wie somit die Datentypen aussehen. Dann sind die Operationen notwendig und die Klassen zu denen sie gehören. So viele Operationen wie möglich müssen durch Axiome beschrieben werden.

6.1.1 Datentypen

Alle Eintragungen ins Grundbuch benötigen Dokumente, die als Nachweis für den Inhalt der Eintragung dienen. Die Dokumente müssen unterschiedliche Daten enthalten und bestimmten Merkmalen genügen, um für eine Eintragung verwendet werden zu können. Das Dokument muss eindeutig feststellen, was ins Grundbuch eingetragen werden soll. Das ist der Inhalt des

Dokumentes. Es ist die Identifizierung der betroffenen Liegenschaft und der Inhalt der Eintragung notwendig. Zusätzlich muss das Dokument auf eine bestimmte Art entstanden sein um etwaigen Betrug vorzubeugen. Außerdem sind die Unterschriften aller beteiligten Personen notwendig und das Datum an dem diese geleistet wurden. Das Modell muss diese Daten und Merkmale abbilden. Im Detail handelt es sich dabei um:

- *Typdefinition:* Es gibt unterschiedliche Arten von Dokumenten. Ein Dokument kann eine von einem Notar beglaubigte Urkunde, eine von zwei Zeugen beglaubigte Urkunde, ein Gerichtsurteil oder eine Genehmigung von einer Behörde sein.
- *Liste der betroffenen Liegenschaften:* Ein Dokument bezieht sich üblicherweise auf eine einzige Liegenschaft. In speziellen Fällen kann es aber vorkommen, dass mehrere Liegenschaften betroffen sind. Das ist beispielsweise dann der Fall, wenn auf einem Grundstück ein Wegerecht errichtet wird und der Begünstigte ein anderes Grundstück (bzw. dessen jeweiliger Eigentümer) ist.
- *Liste der Unterschriften:* Das Dokument muss verschiedene Unterschriften enthalten. Diese Unterschriften dienen als Nachweis der Zustimmung zum Inhalt des Dokumentes. Daher müssen sowohl die Berechtigten, als auch die Verpflichteten das Dokument unterzeichnen. Außerdem sind noch zusätzliche Unterschriften, wie beispielsweise die von Zeugen oder vom Notar, notwendig.
- *Inhalt des Dokumentes:* Die für das Grundbuch relevanten Dokumente haben rechtliche Aspekte zum Inhalt. Daher beschreibt dieser Punkt üblicherweise eine Änderung im rechtlichen Status der Liegenschaft. Das kann nun die Eintragung eines neuen Rechtes (z.B. Pfandrecht oder Wegerecht) betreffen, das kann die Löschung eines Rechtes sein, es kann aber auch die Übertragung eines Rechtes auf eine oder mehrere andere Personen sein (z.B. Eigentumsänderung).
- *Datum der Unterzeichnung:* Das Dokument muss das Datum ausweisen, an dem es unterzeichnet wurde.

Zusätzlich muss ein für die Eintragung ins Grundbuch geeignetes Dokument bestimmten physikalischen Anforderungen genügen. Diese Anforderungen werden zwar in § 27 GBG55 definiert und sind somit eigentlich nicht Bestandteil der Strukturen, die aus anderen Gesetzen übernommen werden, haben aber einen starken inhaltlichen Bezug.

Der Inhalt des Dokumentes, also das dokumentierte Recht, benötigt weitere Daten. Für das Recht ist die Trennung zwischen Berechtigten und Verpflichteten wichtig. Auf Ebene der Dokumente waren hier nur Unterschriften aller Personen, die dem Vertrag zugestimmt haben, vorhanden. Auf der Ebene des Rechtes ist es wichtig die Verpflichteten zu kennen. Nur dann kann geprüft werden, ob diese Personen das gegenständliche Recht auch wirklich garantieren können und dürfen. Zusätzlich muss auch der Rechteinhalt abgebildet werden. Dies geschieht am einfachsten über einen Summentyp (RTyp), der die jeweils notwendigen Daten für die unterschiedlichen Arten von Rechtsinhalten beschreibt. Das ist allerdings nur deshalb möglich, weil im Grundbuch nur eine kleine, genau definierte Anzahl von Einträgen erlaubt ist.

In Haskell sieht der Summentyp folgendermaßen aus:

```
data RTyp =
    Eigentum Anteil RONr |
    Pfandrecht Betrag Betrag RONr |
    Simultanhypothek Betrag Betrag [Nr] RONr |
    PfandR2Simult (Recht RTyp) [Nr] RONr |
    Afterpfandrecht (Recht RTyp) RONr |
    Dienstbarkeit Beschreibung RONr |
    BeschrDienstbarkeit Beschreibung Abgrenzung RONr |
    Reallast Beschreibung RONr |
    Wiederkaufsrecht Betrag RONr |
    Vorkaufsrecht RONr |
    Bestandrecht Beschreibung RONr |
    DinglRecht Beschreibung RONr |
    Vorrang (Recht RTyp) (Recht RTyp) |
    Rangordnung RONr |
    Rechtfertigung (Recht RTyp) |
    Anm Beschreibung |
    AnmZwangsversteigerung |
    AnmAbweisung (Recht RTyp) |
    LoeschVB Datum Datum |
    Loeschung (Recht RTyp)
```

Der Datentyp RTyp beschreibt die für das Grundbuch notwendigen Rechtsinhalte. Es handelt sich dabei nicht nur um Rechte im herkömmlichen Sinn wie beispielsweise das Pfandrecht oder das Eigentumsrecht, sondern auch um andere rechtlich relevante Eintragungsinhalte. So ergibt sich beispielsweise aus der Anmerkung der Abweisung eines Rechtes zwar kein Rechts-

anspruch, es ergibt sich aber die Rechtsfolge, dass das abgewiesene Recht auch später nicht mehr eingetragen werden kann.

Der abstrakte Datentyp `Recht t` hat einen Parameter `t`, der später durch den verwendeten Datentyp für die Rechtsinhalte ersetzt wird. Im vorliegenden Modell wird der Parameter `t` als Platzhalter für `RTyp` verwendet. Es ist aber problemlos möglich, andere Dokumentdefinitionen einzubauen.

Außer dem Rechtstyp müssen noch die Listen mit den Berechtigten und den Begünstigten vorhanden sein:

```
data Recht t = R t [Person] [Person]
```

Der Datentyp für das Dokument schließlich hat zwei Parameter, die den Datentyp für den Inhalt des Dokumentes (das Recht) enthalten. Die übrigen Felder enthalten die Unterzeichner, die Nummern der Liegenschaften, den Dokumenttyp und die Eigenschaften des Dokumentes. Dabei ist der Dokumenttyp (`DokTyp`) die Art der Entstehung der Urkunde. Das letzte Feld gibt einige Eigenschaften des Dokumentes wieder, beispielsweise die Anzahl der Seiten oder ob es sich um ein Original oder eine Abschrift handelt.

```
data Dokument r t = D (r t) [Person][Nr] DokTyp
                    Eigenschaften
```

6.1.2 Klassen und Operationen samt Axiomen

Für die Datentypen werden Operationen benötigt, welche die Funktionalität festlegen. Wie in Abschnitt 3.2.5 beschrieben, werden die Operationen in Klassen zusammengefasst. Es sind Klassen für Datumsangaben, Dokumenttypen, Rechtstypen, Dokumenteigenschaften, Rechte und Dokumente notwendig.

Abbildung 6 zeigt die Abhängigkeiten zwischen den Klassen.

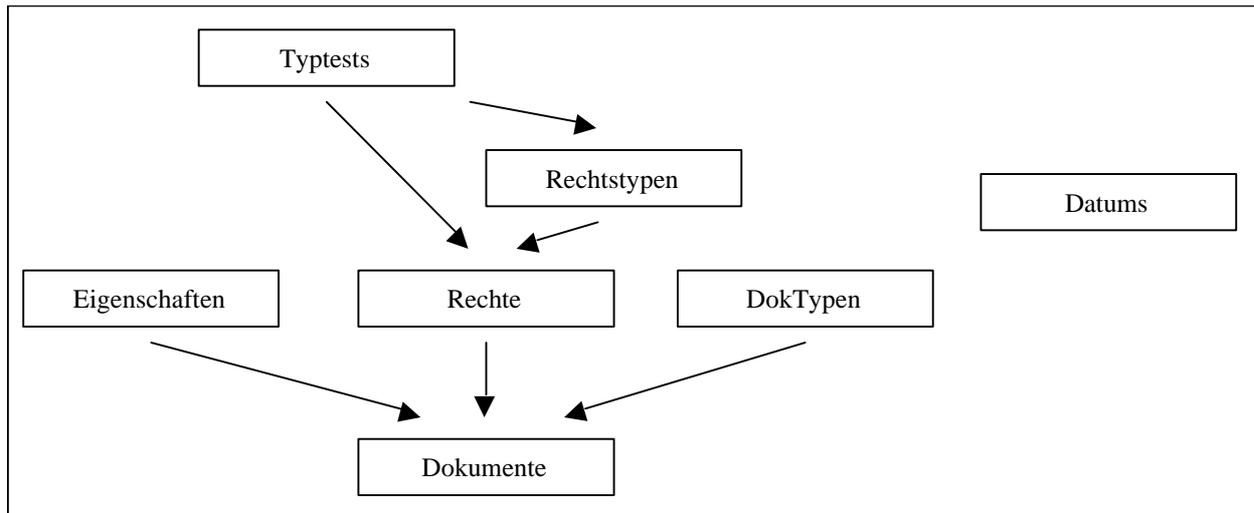


Abbildung 6: Klassenstruktur Dokumente

6.1.2.1 Rechte

Bei den Axiomen muss zwischen den einzelnen Rechtstypen unterschieden werden können. Dazu sind Funktionen notwendig, die prüfen, ob ein bestimmter Typ von Recht vorliegt. Diese Prüffunktionen beginnen alle mit ‚ist‘ und haben alle als Ergebnis einen booleschen Wert. Die Operationen sind alle in der Klasse `TypTest` zusammengefasst.

```

class TypTest r where
  istEigentum, istPfandrecht, istSimultan, istAfterpfand,
  istDienstbarkeit, istBDienstbarkeit, istReallast,
  istWiederkaufsrecht, istVorkaufsrecht, istBestandrecht,
  istDinglRecht, istVorrang, istRechtfertigung,
  istRangordnung, istAnmerkung, istAbweisung,
  istZwangsversteigerung, istLoeschung :: r -> Bool
  
```

Um die bei den einzelnen Rechtstypen vorkommenden Daten auslesen zu können benötigt man Zugriffsfunktionen. Hier muss speziell bei der Implementierung beachtet werden, dass viele Operationen nur für bestimmte Arten von Rechten sinnvoll sind. Beispielsweise gibt es bei einem Vorkaufsrecht keinen Betrag. Den gibt es nur bei einer Eigentumsübertragung, einem Pfandrecht oder einer Simultanhypothek sowie bei einem Wiederkaufsrecht. Die Prüfung, ob ein Rechtstyp einen Verweis auf ein anderes Recht enthält, kann als Axiom definiert werden, wenn man die vorkommenden Rechtstypen samt Parametern kennt. Dies erfolgt unter Zuhilfenahme der Operationen der Klasse `TypTest`.

```

class (TypTest t) => Rechtstypen t where
  anteil          :: t -> Anteil
  typBetrag       :: t -> Betrag
  typBeschr       :: t -> Beschreibung
  typRONummer     :: t -> RONr
  typVonDt        :: t -> Datum
  typBisDt        :: t -> Datum
  typHaupteinlage :: t -> Nr
  typNebeneinlage :: t -> [Nr]

  typEnthRecht   :: t -> Bool
  typEnthRecht r =
    (istAfterpfand r)    || (istVorrang r)    ||
    (istRechtfertigung r) || (istLoeschung r) ||
    (istAbweisung r)    || (istSimultan r)

```

Die bisher definierten Klassen sind die Voraussetzung für die Klasse Rechte. Abkürzungen für häufig benötigte Daten sind sinnvoll. Ein Beispiel dafür ist der Test, ob sich das Recht auf ein anderes Recht bezieht. Das ist beispielsweise bei einer Rechtfertigung oder einer Löschung der Fall. Falls es sich um ein Eigentumsrecht handelt, ist die Funktion `eigentuemer` wichtig, die eine Liste mit den neuen Eigentümern samt den jeweiligen Anteilen liefert. Interessant ist weiters noch die Operation `Rangsort`. Diese Operation hat als Eingangswert eine Liste von Rechten. Die Rechte werden nach dem Rang sortiert wobei Vorrangeinräumungen und Rangordnungen berücksichtigt werden. Die ursprünglich nach dem Datum der Eintragung sortierte Liste ist somit nach Anwendung der Operation nach der Bedeutung der Rechte sortiert.

```

class (Rechtstypen t, TypTest (r t), Eq t, Eq (r t)) =>
  Rechte r t where
  rechtstyp      :: r t -> t
  berechnigte    :: r t -> [Person]
  belastete      :: r t -> [Person]
  betrRecht      :: r t -> r t
  betrRecht2     :: r t -> r t

  eigentumsanteil :: r t -> Anteil
  eigentumsanteil = anteil.rechtstyp

```

```

eigentuemer      :: r t -> [(Person,Anteil)]
eigentuemer r = if istEigentum r then
                  zip (berechtigte r)
                      (replicate ((length.berechtigte) r)
                                   (eigentumsanteil r))
                  else []

anteilVon        :: Person -> r t -> Anteil
anteilVon p r =
  if istEigentum r
  then (foldl (+) 0.map snd.filter t.eigentuemer) r
  else zero
  where t (e,a) = e == p

enthRecht        :: r t -> Bool
enthRecht       = typEnthRecht.rechtstyp

betrag          :: r t -> Betrag
betrag          = typBetrag.rechtstyp

beschreibung     :: r t -> Beschreibung
beschreibung     = typBeschr.rechtstyp

roNummer        :: r t -> RONr
roNummer        = typRONummer.rechtstyp

rangsort        :: [r t] -> [r t]
rangsort rs = filter (istEigentum.rechtstyp) srs ++
               filter (not.istEigentum.rechtstyp) srs where
  srs = rsort rs rs
  rsort [] res = res
  rsort (r:rs) res =
    if (istVorrang.rechtstyp) r
    then rsort rs
         ((delete r.insertBefore r1 r2.delete r2) res)
    else
      if (istRangordnung.rechtstyp) r
      then
        if r3 /= r
        then rsort rs
             ((delete r.insertBefore r r3.delete r3) res)

```

```

    else rsort rs res
  else rsort rs res
where
  r1 = betrRecht2 r
  r2 = betrRecht r
  r3 = if length r3' == 0 then r else head r3'
  r3' = filter ((roNummer r ==).roNummer) (res \\ [r])

```

6.1.2.2 Dokumente

Für den Umgang mit Dokumenten sind außerdem Klassen für die Dokumentstypen und die physischen Eigenschaften der Dokumente notwendig. Die Dokumentstypen enthalten die Art der Erstellung des Dokumentes und die Liste der dabei anwesenden Zeugen. Für beide Felder sind Observer notwendig. Zusätzlich gibt es eine Prüffunktion, welche testet, ob die Art der Erstellung zulässig ist für eine ins Grundbuch einzutragende Urkunde. Für die Felder der Dokumenteigenschaften gibt es ebenfalls Observer. Die Felder des Datentyps wurden nach den folgenden Stellen im Grundbuchgesetz festgelegt:

§ 27. (1) Die Urkunden, auf Grund deren eine bücherliche Eintragung geschehen soll, müssen frei von solchen sichtbaren Mängeln sein, durch die ihre Glaubwürdigkeit geschwächt wird, und, wenn sie aus mehreren Bogen bestehen, so geheftet sein, dass kein Bogen unterschoben werden kann.

(2) ...

§ 87. (1) Die Urkunden, auf Grund deren eine Eintragung erfolgen soll, sind im Original beizulegen.

(2) ...

In diesen beiden Paragraphen werden Eigenschaften für eine Urkunde, auf Grund dessen ein Recht ins Grundbuch eingetragen werden soll, festgelegt. Urkunden sind Dokumente, die nach bestimmten Regeln entstanden sind. § 27 Abs.1 verlangt, dass Urkunden geheftet sein müssen, wenn sie aus mehreren Bögen (Blättern) bestehen. Es sind daher zwei Felder notwendig, von denen eines die Zahl der Blätter enthält und das andere das Vorhandensein einer Heftung zeigt. Die Modellierung der sichtbaren Mängel geschieht über einen Booleschen Wert. Die Entscheidung, ob eine Urkunde Mängel aufweist wird nicht modelliert. Die Interpretation des Begriffes ‚sichtbare Mängel‘ wird somit vermieden. Was allerdings modelliert werden muss ist

die Auswirkung sichtbarer Mängel. Dies geschieht in Abschnitt 6.2. § 87 Abs.1 sagt aus, dass die Urkunde im Original beizulegen ist. Das ist die letzte der modellierten Eigenschaften.

```
class DokTypen t where
  erstellung :: t -> Art
  zeuge      :: t -> [Zeuge]

  spezErfOK :: t -> Bool
  spezErfOK dt = elem (erstellung dt) erlaubteUrk

class Eigenschaft e where
  geheftet, maengel, beglaubigt, originalOK :: e -> Bool
  seiten :: e -> Int

  eigOK :: e -> Bool
  eigOK e =
    (((seiten e)>1 && (geheftet e)) || (seiten e==1)) &&
    (not.maengel) e && (originalOK e)
```

Die Definition der Operationen für die Dokumente baut auf den bisher definierten Klassen auf. Für die Dokumente benötigt das Modell fünf unterschiedliche Konstruktoren, von denen jeder einen speziellen Typ von Dokument erstellt. Diese Dokumente sind die im Grundbuchgericht intern erzeugten Dokumente. Es handelt sich dabei um die Rangordnung und um Anmerkungen diverser Vorgänge (Löschung, Abweisung und Simultanhypothek). Die Klasse enthält keine Konstruktoren für Dokumente, die außerhalb des Grundbuchgerichtes erstellt werden. Solche Dokumente werden nur für Testzwecke benötigt, die verwendeten Datentypen stehen hierbei fest. Somit können die Konstruktoren der Datentypen verwendet werden.

Die Klasse enthält auch eine Operation, welche bei extern erstellten Dokumenten prüft, ob alle notwendigen Personen das Dokument unterzeichnet und damit ihre Zustimmung zum Inhalt ausgedrückt haben. Der Test besteht aus zwei Teilen. Berechtigte und Verpflichtete müssen unterschrieben haben. Dazu wird eine Liste der Berechtigten und der Verpflichteten erstellt. Eine Verschneidung mit der Liste der Unterzeichner liefert alle Berechtigten und Verpflichteten, welche das Dokument unterzeichnet haben. Dieses Ergebnis muss mit der Liste der Berechtigten und Verpflichteten übereinstimmen. Der zweite Teil des Tests prüft, ob die Eigentümer der Liegenschaft unterzeichnet haben. Die dazu notwendige Liste der Eigentümer muss der Operation als Parameter übergeben werden. Nur wenn beide Tests positiv ausfallen, ist die Eintragung des Dokumentes erlaubt.

```

class (Rechte r t) => Dokumente d r t where
  recht                :: d r t -> r t
  unterzeichner       :: d r t -> [Person]
  dokumentTyp         :: d r t -> DokTyp
  grundbuchseinlagen :: d r t -> [Nr]
  eigenschaften       :: d r t -> Eigenschaften

  makeRO              :: [Person] -> Nr -> [d r t] -> d r t
  makeLoeschVB       :: Datum -> Datum -> [Person] -> Nr ->
                      d r t
  makeAbweisung      :: r t -> [Nr] -> [Person] -> d r t
  makeLoeschung      :: r t -> [Nr] -> [Person] -> d r t
  makeAnmSimultan    :: [Person] -> [Nr] -> d r t

  eintragErlaubt     :: d r t -> [Person] -> Bool
  eintragErlaubt d es =
    (sort (intersect (gegen ++ fuer) (unterzeichner d))
     == sort (gegen ++ fuer)) &&
    (sort (intersect es (unterzeichner d)) == sort es)
  where fuer = (berechtigte.recht) d
        gegen = (belastete .recht) d

  dokumentOK         :: d r t -> Bool
  dokumentOK         = eigOK.eigenschaften

  dienende           :: (Rechte r t) => d r t -> [Person]
  dienende           = belastete.recht

  spezErfordernisse :: d r t -> Bool
  spezErfordernisse = spezErfOK.dokumentTyp

```

6.1.2.3 Datumsangaben

Die Klasse `Datums` steht außerhalb der Struktur, da Datumsangaben zwar vorkommen, die Operationen der Klasse `Datums` aber noch keine Verwendung finden. Die für ein Datum notwendigen Operationen sind das Lesen der Einzelteile und die Definition des korrekten Zählvorganges. Dies geschieht durch die Operation `incTag`, die den darauffolgenden Tag für jedes beliebige Datum bestimmt. Dabei ist `months` eine Liste, welche die Anzahl der Tage der einzelnen Monate enthält. Mit Hilfe der Funktion `incTage` können größere Zeiträume übersprungen werden, indem einfach die Anzahl der weiterzuzählenden Tage angegeben wird.

```

class Datums a where
  newdatum      :: Int -> Int -> Int -> a
  tag,monat,jahr :: a -> Int

  incTag        :: a -> a
  incTag d = if tag d + 1 <= (months !! (monat d -1))
              then newdatum (tag d + 1) (monat d) (jahr d)
              else if monat d + 1 <= 12
                    then newdatum 1 (monat d + 1) (jahr d)
                    else newdatum 1 1 (jahr d + 1)

  incTage       :: a -> Int -> a
  incTage d 0 = d
  incTage d x = incTage (incTag d) (x-1)

```

6.1.3 Instanziierung der Klassen

Die Instanziierung der Klassen erfolgt durch Definition der Operationen, die nicht durch Axiome beschrieben werden. Im allgemeinen verwendet man nur noch Pattern-Matching um die Observer zu definieren. Die Konstruktoren werden ebenfalls durch Einsetzen des verwendeten Datentyps definiert. Als Beispiel soll hier die Instanz der Klasse `Datums` für den Datentyp `Datum` dienen:

```

instance Datums Datum where
  newdatum t m j      = Datum t m j
  tag (Datum t m j)   = t
  monat (Datum t m j) = m
  jahr (Datum t m j)  = j

```

Manchmal ist es notwendig, die Gleichheit von Datensätzen und die Ordnungsrelation explizit zu definieren. Das ist vor allem dann notwendig, wenn die Gleichheit nicht durch den Inhalt sämtlicher Felder des Datensatzes beschrieben wird, sondern beispielsweise durch eindeutige Kennungen. Als Beispiel für diese Vorgangsweise soll die Ordnungsrelation für den Datentyp `Datum` dienen:

```

instance Ord Datum where
  compare (Datum t1 m1 j1) (Datum t2 m2 j2)
    | (t1 == t2) && (m1 == m2) && (j1 == j2) = EQ
    | (j1 > j2) = GT
    | (j1 < j2) = LT

```

```

| (m1 > m2) = GT
| (m1 < m2) = LT
| (t1 > t2) = GT
| (t1 < t2) = LT
| otherwise = error "no other case fits"

```

Die Instanzierung der Operationen der Klasse `TypTest` erfolgt durch Pattern-Matching, wobei es nicht auf den Inhalt der Felder ankommt, sondern auf den zu verwendenden Datentypkonstruktor. Daher gibt es für jede der Funktionen zwei Zeilen. Die erste Zeile definiert, bei welchem Datentypkonstruktor die Operation den Wert `True` liefern soll, bei allen übrigen Parametern liefert die Operation den Wert `False`.

```

instance TypTest RTyp where
  istEigentum (Eigentum _ _ _) = True
  istEigentum _                 = False
  istPfandrecht (Pfandrecht _ _ _) = True
  istPfandrecht _                 = False
  ...

```

Bei der Instanzierung der Klasse `RTyp` muss auf zwei Aspekte Rücksicht genommen werden. Zunächst kann eine Abfrage bei mehreren Rechtstypen verwendet werden. Es gibt beispielsweise mehrere Rechte, für die ein Betrag benötigt wird, nämlich das Pfandrecht, das Wiederkaufsrecht und die Eigentumsübertragung. Außerdem ist es wichtig, für jeden Rechtstyp ein gültiges Ergebnis zu liefern. Das ist vor allem dann wichtig, wenn eine Liste von Rechten nach bestimmten Eigenschaften wie etwa dem Alleineigentum (der Anteil beträgt dann 1) durchsucht wird.

```

instance Rechtstypen RTyp where
  anteil (Eigentum a _ _) = a
  anteil _                 = zero
  typBetrag (Pfandrecht b p _) = b
  typBetrag (Wiederkaufsrecht b _) = b
  typBetrag (Eigentum _ b _) = b
  typBetrag _                 = 0
  ...

```

Für die Dokumente ist ebenfalls eine Implementierung der Observer notwendig. Zusätzlich müssen die Konstruktoren definiert werden. Jeder dieser Konstruktoren liefert ein Dokument, das vom Grundbuch erzeugt wird. Erkennbar ist dieser Umstand daran, dass beim Dokumenttyp

die Konstante `gericht` verwendet wird und als ‚Zeuge‘ das Grundbuch selbst fungiert. Das unterscheidet diese Dokumente von anderen Gerichtsurteilen, bei denen nicht das Grundbuch, sondern Richter und Beisitzer den korrekten Inhalt bezeugen.

```
instance Dokumente Dokument Recht RTyp where
  ...
  makeRO ps nr ds =
    D (R (Rangordnung
          (((1+).length.filter (istRangordnung.recht)) ds))
        ps ps) ps [nr]
      (DokTyp gericht ["Grundbuch"])
      (Eig 1 False False False True Original)
  makeAbweisung r nrs es =
    D (R (AnmAbweisung r) es []) es nrs
      (DokTyp gericht ["Grundbuch"])
      (Eig 1 False False False True Original)
  ...
```

6.2 Objektstruktur des GBG55

Die ersten Paragraphen des Grundbuches beschreiben die Teile aus denen sich das Grundbuch zusammensetzt. Diese Beschreibung dient zur Modellierung der Objektstruktur. Diese Paragraphen lauten:

§ 1. Das Grundbuch besteht aus dem Hauptbuch und der Urkundensammlung.

§ 2. (1) Das Hauptbuch wird aus den Grundbuchseinlagen gebildet.

(2) Die Grundbuchseinlagen sind bestimmt zur Eintragung:

- 1. der Grundbuchskörper und ihrer Änderungen;*
- 2. der sich auf die Grundbuchskörper beziehenden dinglichen Rechte und ihrer Änderungen.*

§ 6. (1) Von jeder Urkunde, auf Grund deren eine bücherliche Eintragung vorgenommen wird, ist bei dem Grundbuch eine beglaubigte Abschrift zurückzubehalten.

(2) Diese Abschriften bilden die Urkundensammlung.

§ 1 beschreibt das Grundbuch durch Angabe von zwei Elementen, aus denen das Grundbuch aufgebaut ist. Hier muss man die Annahme treffen, dass nach den Worten ‚besteht aus‘ alle

Elemente des Grundbuches angeführt sind. Ähnliches gilt für § 2 Abs. 1, in dem angenommen werden muss, dass das Hauptbuch nur aus den Grundbuchseinlagen gebildet wird und es keine sonstigen Teile gibt. Bei § 2 Abs. 2 hingegen ist klar, dass es sich bei den nachfolgenden Punkten um eine komplette Auflistung aller möglichen Eintragungen handelt, da die Formulierung ‚sind bestimmt zur Eintragung‘ keinen anderen Schluss zulässt. § 6 zeigt die Verbindung des Grundbuchgesetzes zu den Dokumenten. Eintragungen ins Grundbuch können nur aufgrund von Dokumenten erfolgen. Beglaubigte Abschriften dieser Dokumente bilden die Urkundensammlung. Abbildung 7 zeigt die beschriebene Struktur unter Einbindung der in Abschnitt 6.1 definierten Objekte.

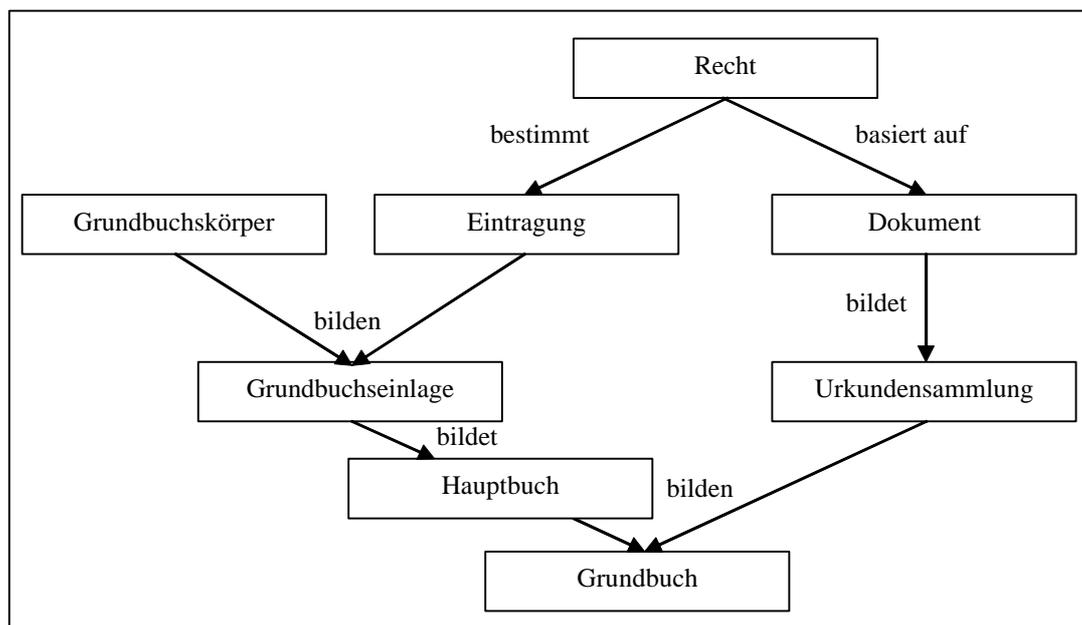


Abbildung 7: Die Objektstruktur des Grundbuchgesetzes

Ausgehend von der Definition der Datentypen beim Grundbuch sieht man, dass das Grundbuch aus Hauptbuch und Urkundensammlung besteht. Die Urkundensammlung ist eine Liste von Dokumenten, kann also mit $[d \ r \ t]$ beschrieben werden. Dabei ist d der Parameter für den Datentyp des Dokumentes, und $r \ t$ sind die bereits aus Abschnitt 6.1.2.2 bekannten Parameter des Dokument-Datentyps. Das Hauptbuch muss Rechte speichern und benötigt somit den Datentyp $\text{Hauptbuch } r \ t$. Das Hauptbuch besteht aus einer Liste von Grundbuchseinlagen. Die Einlage dient zur Eintragung des Grundbuchkörpers und der dinglichen Rechte (jeweils samt Änderungen). Diese beiden Teile bilden somit die Grundbuchseinlage. Der Grundbuchkörper muss die Liste der enthaltenen Grundstücke und die Änderungen in dieser Liste speichern. Dinglichen Rechte umfassen die Daten der Eintragung. Zusätzlich hat eine Grundbuchseinlage allerdings noch einen Namen (§ 85 Abs.1: *Die Grundbuchseinlagen, in*

denen eine Eintragung geschehen soll, sind mit der nämlichen Bezeichnung anzuführen, unter der sie im Grundbuch erscheinen.). Im Modell wird dieser Name als `Nr` bezeichnet und ist als `String` modelliert, da eine Bezeichnung eine beliebige Kette von Zeichen sein kann.

```

data Grundbuch d r t = G (Hauptbuch r t) [d r t]

data Hauptbuch r t = H [Einlage r t]

data Einlage r t = E Nr Koerper [Eintragung r t]

data Koerper = K [Nr] [Aenderung]

type Nr = String

data Aenderung = Abschreibung Nr | Zuschreibung Nr

data Eintragung r t =
  Einverleibung (r t) Datum Loesch |
  Vormerkung (r t) Datum Frist Rechtfertigung Abweisung
  Loesch |
  Anmerkung (r t) Datum Loesch

```

Die drei Möglichkeiten, die es bei der Eintragung gibt, werden von § 8 definiert:

§ 8. Die grundbücherlichen Eintragungen sind:

1. *Einverleibungen (unbedingte Rechtserwerbungen oder Löschungen – Intabulationen oder Extabulationen), die ohne weitere Rechtfertigung oder*
2. *Vormerkungen (bedingte Rechtserwerbungen oder Löschungen – Pränotationen), die nur unter der Bedingung ihrer nachfolgenden Rechtfertigung die Erwerbung, Übertragung, Beschränkung oder Erlöschung bürgerlicher Rechte bewirken, oder*
3. *bloße Anmerkungen.*

In den Anmerkungen zu diesem Paragraphen werden auch noch die Ab- und Zuschreibung, die Ersichtlichmachung und die Löschung erwähnt. Die Ab- und Zuschreibung ist allerdings keine Eintragung eines Rechtes, sondern ändert den Umfang des Grundbuchkörpers. Die Ersichtlichmachung ist vom technischen Standpunkt vergleichbar mit der Anmerkung, hat aber eine geringere rechtliche Bedeutung. Sie benötigt ebenfalls ein Dokument. Die Ersichtlichmachung kann daher unter Verwendung der Operationen der Anmerkung modelliert werden. Die Löschung hingegen ist ein Vorgang, der gleichberechtigt neben der Eintragung eines neuen Rechtes steht. Es handelt sich dabei um die ‚Umkehrung‘ der Eintragung eines Rechtes und

kann somit als Einverleibung modelliert werden, bei der die Löschung eines Rechtes eingetragen wird. Somit können die drei zusätzlichen Arten der Eintragung, die in den Anmerkungen erwähnt werden, entweder auf einen der drei Eintragungstypen Einverleibung, Vormerkung und Anmerkung zurückgeführt werden, oder sind für den Datentyp der Eintragung nicht relevant.

Für jeden dieser Datentypen sind Operationen notwendig, die in Klassen zusammengefasst werden. Abbildung 8 zeigt die Abhängigkeiten zwischen den unterschiedlichen Klassen. Die Klassen `Dokument` und `Recht`, die in Abschnitt 6.1 besprochen wurden, werden als vorhanden vorausgesetzt. Zusätzlich gibt es noch die Klassen `GBTeile` und `Abschriften`. Die Klasse `GBTeile` enthält nur die Konstruktoren und die Observer für das Grundbuch. Die Klasse `Abschriften` umfasst alle Operationen, die Daten aus dem Grundbuch auslesen. Dazu werden die Operationen von `GBTeile` verwendet. Die Klasse `Grundbücher` enthält dann nur mehr die Operationen, welche für die Eintragung, Löschung und Änderung von Daten im Grundbuch benötigt werden. Die genaue Definition dieser Klassen kann in Anhang D gefunden werden. Die wichtigsten Teile des Modells werden in den folgenden Abschnitten anhand verschiedener Vorgänge im Grundbuch beschrieben.

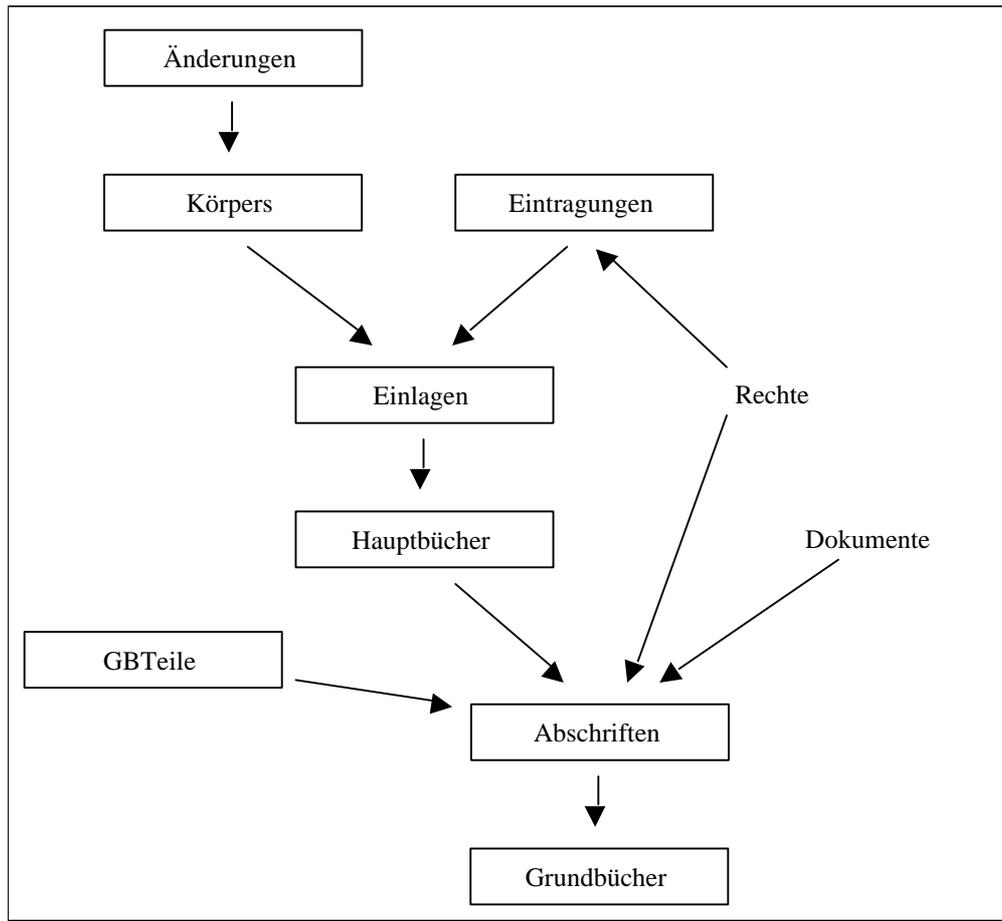


Abbildung 8: Klassen des Grundbuchgesetzes

6.3 Eintragungen

Das Grundbuchgesetz sieht drei unterschiedliche Arten der Eintragung vor (§ 8). Zunächst gibt es die Einverleibung, die eine unbedingte Rechtserwerbung darstellt. Im Gegensatz dazu ist die Vormerkung eine bedingte Rechtserwerbung. Hier muss vor dem tatsächlichen Erwerb des Rechtes noch eine Bedingung erfüllt werden. Schließlich gibt es noch die Anmerkung, mit der sonstige relevante Daten eingetragen werden können.

Die Eintragung wird ohne Einvernehmung der Parteien und ohne Zwischenerledigungen durchgeführt (§ 95). Daher gibt es während der Durchführung der Eintragung keine Interaktion. Hier tritt das aus der Datenbanktheorie bekannte Prinzip der ‚atomare Transaktionen‘ auf – eine Aktion wird entweder komplett durchgeführt oder gar nicht durchgeführt. Auf das Grundbuch umgelegt hieße das: Eine Eintragung wird entweder durchgeführt oder abgewiesen.

Eine Eintragung benötigt einige Angaben. Zunächst muss feststehen, welche Art der Eintragung vorgenommen werden soll. Außerdem wird für die Eintragung eine Urkunde

benötigt, welche das einzutragende Recht dokumentiert. Schließlich ist auch noch das Datum der Eintragung von Interesse, da es im Grundbuch festgehalten wird.

Die Art der Eintragung wird über den Datentyp `Eintragungstyp` definiert. Der Datentyp hat zusätzlich zu den drei bereits genannten Arten der Eintragung (Einverleibung, Vormerkung und Anmerkung) noch einen vierten mit Namen `AEv`. Laut § 85 Abs. 3 beinhaltet die Einverleibung auch stillschweigend eine Vormerkung. Ein Antrag auf Einverleibung kann folglich in einen Antrag auf Vormerkung umgewandelt werden, wenn das eingereichte Dokument nicht allen entsprechenden Anforderungen genügt. Ist eine solche Vorgangsweise vom Antragsteller nicht erwünscht, so hat er die Vormerkung explizit auszuschließen. Diese Art der Eintragung ergibt einen vierten Eintragungstyp, obwohl es sich eigentlich nur um eine Abart der Einverleibung handelt.

```

data Eintragungstyp = Ev | Vo | An | AEv

eintragenGB :: (Rechte r t, Ord (r t)) =>
  g d r t -> Eintragungstyp -> d r t -> Datum -> g d r t
eintragenGB g Ev d dt =
  if (istLoeschung.recht) d then
    if (beglaubigt (eigenschaften d)) && (dokumentOK d) &&
      (belastete.recht) d == (berechtigte.betrRecht.recht) d
    then
      if (spezErfordernisse d)
      then FALL 1
      else FALL 2
    else FALL 3
  else
    if (beglaubigt (eigenschaften d)) && (dokumentOK d) &&
      (eintragErlaubt d (nub (foldl (++) [])
        (map (eigentuemervon g) (liegenschaften d)))) then
      if (spezErfordernisse d) then FALL 1
      else FALL 2
    else FALL 3

```

```

eintragenGB g t d dt =
  if (beglaubigt.eigenschaften) d && (dokumentOK d) &&
    (eintragErlaubt d (nub (foldl (++) [])
      (map (eigentuemervon g) (liegenschaften d))))))
  then FALL 4
  else FALL 3

```

Im angeführten Code sind nur die Entscheidungspunkte enthalten. Die resultierenden Tätigkeiten werden hier nur als Fallunterscheidungen angegeben. Eine genaue Beschreibung der unterschiedlichen Fälle folgt in den Abschnitten 6.3.2 bis 6.3.4. Die einzelnen Fälle sind:

- Fall 1: Es wird eine Einverleibung durchgeführt
- Fall 2: Es wird eine Vormerkung durchgeführt obwohl um eine Einverleibung angesucht wurde
- Fall 3: Der Antrag wird abgewiesen und die Abweisung wird angemerkt
- Fall 4: Es wird eine Eintragung in der gewünschten Form durchgeführt – das kann eine Vormerkung oder eine Anmerkung sein.

Die Unterscheidung zwischen diesen Fällen erfolgt durch die in §§ 21, 26, 27 und 31-34 definierten Regeln. Zunächst entscheiden die §§ 21, 26 und 27 über die grundsätzliche Rechtmäßigkeit der Eintragung. § 21 sagt, dass Eintragungen nur gegen den Eigentümer der betroffenen Liegenschaft oder den Berechtigten des betroffenen Rechtes zulässig ist. Die §§ 26 und 27 legen Eigenschaften für das Dokument fest. Diese Tests entscheiden, ob eine Eintragung durchgeführt wird (Fälle 1 und 2), oder ob der Antrag abgelehnt wird (Fall 3). Da bei der Löschung eines Rechtes (im Gegensatz zu allen übrigen Eintragungen) die Zustimmung der Berechtigten notwendig ist, wird dieser Fall extra geregelt. Die §§ 31-34 geben an, welchen speziellen Eigenschaften das Dokument noch für eine Eintragung ins Grundbuch genügen muss. Diese Erfordernisse können aber auch nachträglich erfüllt werden. Sind also nicht alle dieser Erfordernisse erfüllt, so wird eine beantragte Einverleibung nicht abgelehnt, sondern in eine Vormerkung umgewandelt und diese Vormerkung wird dann durchgeführt. Diese Umwandlung ist nur bei einer Einverleibung möglich. Daher ist der Antrag der Einverleibung im Code zunächst separat behandelt und anschließend wird der allgemeine Fall beschrieben.

6.3.1 Generelle Vorgangsweise

Bei einer Eintragung müssen zwei Dinge erledigt werden: Das Recht muss im Hauptbuch eingetragen und das Dokument in die Urkundensammlung aufgenommen werden:

§ 4. Die Erwerbung, Übertragung, Beschränkung und Aufhebung der bürgerlichen Rechte (§ 9) wird nur durch ihre Eintragungen in das Hauptbuch erwirkt.

§ 6. (1) Von jeder Urkunde, auf Grund deren eine bürgerliche Eintragung vorgenommen wird, ist bei dem Grundbuch eine beglaubigte Abschrift zurückzubehalten.

(2) Diese Abschriften bilden die Urkundensammlung.

Unter der Annahme, dass am Dokument selbst nichts ergänzt werden muss (dass also das Eintragsdatum und ähnliche Ergänzungen bereits vorgenommen wurden), sieht dieser Vorgang formal folgendermaßen aus:

```
grundbuecher (eintragenHB (hauptbuch g) (rechtseintrag t)
              (grundbuchseinlagen d))
              (d:urkundensammlung g)
```

Es wird also die Operation `eintragenHB` aufgerufen, welche die Eintragung des Rechtes ins Hauptbuch übernimmt und zusätzlich wird das Dokument zur Urkundensammlung hinzugefügt. Notwendig sind für die Operation `eintragenHB` das Hauptbuch selbst, der vorzunehmende Eintrag und die Liste der Grundbuchseinlagen, bei denen der Eintrag vorzunehmen ist. Das Ergebnis der Operation ist das um die notwendigen Eintragungen ergänzte Hauptbuch. Der vorzunehmende Eintrag wird dabei von einer eigenen Operation bestimmt, welche als lokale Funktion auf alle Parameter der übergeordneten Operation zugreifen kann. Als Kriterium für die Art des Eintrages wird der Eintragstyp als Parameter übergeben. Die Erstellung der Eintragung kann einfach gehalten werden, da explizit angegeben werden muss, was im Grundbuch eingetragen werden soll (§ 85 Abs. 2):

§ 85. (1) ...

(2) Im Begehren ist genau anzugeben, was im Grundbuch eingetragen werden soll.

(3) ...

Der Einfachheit halber wurde im Modell angenommen, dass jedes Dokument nur ein Recht enthalten kann und dass sich die Eintragung auf dieses Recht bezieht. Ein Dokument darf mehrere Rechte beinhalten. Dann ist beim Antrag genau anzugeben, welche Rechte eingetragen werden sollen. Der folgende Code ist eine lokale Funktion von `eintragenGB`. Die Parameter `d` und `dt` stammen vom Aufruf von `eintragenGB` und beinhalten das eingereichte Dokument und das aktuelle Datum.

```

rechtseintrag Ev = Einverleibung (recht d) dt False
rechtseintrag AEv = Einverleibung (recht d) dt False
rechtseintrag Vo  = Vormerkung (recht d) dt (incTage dt 14)
                                False False False
rechtseintrag An  = Anmerkung (recht d) dt False

```

Auf der Ebene der Hauptbücher wird geprüft, welche Grundbuchseinlagen vom Eintrag betroffen sind und in diesen Einlagen wird dann die Eintragung vorgenommen. Das Hauptbuch ist eine Liste von Grundbuchseinlagen. § 85 Abs. 1 verlangt die namentliche Nennung alle Grundbuchseinlagen, in denen die Eintragung erfolgen soll:

§ 85. (1) Die Grundbuchseinlagen, in denen eine Eintragung geschehen soll, sind mit der nämlichen Bezeichnung anzuführen, unter der sie im Grundbuch erscheinen.

(2) ...

(3) ...

Daher muss die Bezeichnung jeder Grundbuchseinlage mit der Bezeichnung übereinstimmen, die als Parameter übergeben wurde. Für die entsprechenden Grundbuchseinlagen wird dann die Operation `eintragen` aufgerufen. Anschließend gibt die Operation das geänderte Hauptbuch zurück.

```

eintragenHB :: h r t -> Eintragung r t -> [Nr] -> h r t
eintragenHB h et ezs =
    hauptbuecher (map eintr' (gbEinlagen h))
where
    eintr' einlage = if elem (einlagezahl einlage) ezs
                    then eintragen einlage et else einlage

```

Die Eintragung wird dann in der Grundbuchseinlage zunächst noch einmal überprüft. Dabei wird geprüft, ob die Eintragung bereits einmal angemerkt, dann aber das Recht aberkannt wurde. Es ist nämlich nicht möglich, eine Eintragung, die bereits abgewiesen wurde, noch

einmal zu beantragen (§§ 47, 48). Daher muss die Abweisung einer Eintragung auch in der Grundbuchseinlage angemerkt werden.

§ 47. Ist die Vormerkung deshalb gelöscht worden, weil dem Kläger das vorgemerkte Recht endgültig aberkannt oder die Vormerkung nicht für gerechtfertigt erklärt worden ist oder weil der, der sie erwirkt hat, unbedingt darauf verzichtet hat, so ist jede in der Folge auf Grund der nämlichen Urkunde abermals angesuchte Vormerkung desselben Rechtes entweder von Amts wegen abzuweisen oder, wenn dies unterblieben und abermals eine Vormerkung vorgenommen worden ist, diese Vormerkung wieder zu löschen, sobald der Gegner anzeigt, daß sie schon einmal gelöscht worden ist.

§ 48. (1) Ist dagegen die Vormerkung nur aus dem Grunde gelöscht worden, weil die Rechtfertigungsklage nicht in gehöriger Zeit angebracht worden ist, so kann zwar abermals eine Vormerkung angesucht werden; diese äußert jedoch ihre rechtliche Wirksamkeit erst von dem Zeitpunkte der Überreichung des neuen Gesuches.

(2) Auch steht dem Eigentümer der Liegenschaft oder des bürgerlichen Rechtes frei, auf Feststellung des Nichtbestehens des vorgemerkt gewesenen Rechtes zu klagen und im Fall eines günstigen Erkenntnisses durch dessen Anmerkung im Grundbuch einer wiederholten Bewilligung der Vormerkung vorzubeugen.

Soll die Einverleibung oder Vormerkung eines Rechtes oder eine Anmerkung erfolgen, so wird die Eintragung einfach zu den bereits vorhandenen Eintragungen hinzugefügt. Wenn allerdings eine Eintragung gelöscht werden soll ist der Vorgang umfangreicher. Zunächst muss der betroffene Eintrag gefunden und gelöscht werden. Das erfolgt mit Hilfe der Operation `eintragungLoeschen`, welche einen Eintrag als nicht mehr gültig markiert. Anschließend muss aber auch der Eintrag der erfolgten Löschung zur Liste der Eintragungen hinzugefügt werden.

Das Grundbuchsgesetz sagt nicht, wie eine Löschung zu geschehen hat. Die Modellierung erfolgt daher nach dem Prinzip, dass alle Änderungen im Datenbestand nachvollziehbar sein müssen. Daher darf die Löschung nicht durch Entfernen des Eintrages vorgenommen werden.

```

eintragen :: e r t -> Eintragung r t -> e r t
eintragen einlage e =
  if (istVormerkung e) && eintragExists &&
    (eintragGeloescht vormerk) &&
    ((not.eintragAbgewiesen) vormerk)
  then einlage
  else
    if ((istLoeschung.eintragRecht) e) && (istEinverleibung e)
    then einlagen (einlagezahl einlage) (koerper einlage)
      (e:(loesch (recht2Eintrag (betrRecht (eintragRecht e))
        (eintrliste einlage))
        (eintrliste einlage)))
      else einlagen (einlagezahl einlage) (koerper einlage)
        (e:eintrliste einlage)
where
  vormerk = (eintrliste einlage) !! untag
            (elemIndex (eintragRecht e) rs)
  rs      = map eintragRecht (eintrliste einlage)
  eintragExists = elem (eintragRecht e) rs
  untag (Just a) = a
  loesch sz [] = []
  loesch sz (z:zs) = if sz == z then eintragungLoeschen z:zs
                    else z:loesch sz zs

```

6.3.2 Einverleibung

Die Anforderungen an die für die Einverleibung geeigneten Dokumente wird von den §§ 31-34 beschrieben. Diese Paragraphen spezifizieren die Urkunden, die für eine Eintragung ins Grundbuch in Frage kommen. Dazu zählen beispielsweise notariell beglaubigte Urkunden oder Genehmigungen einer Behörde. Im Allgemeinen ist es so, dass eine Urkunde vertrauenswürdig sein muss und dass sichergestellt sein muss, dass alle Unterzeichner dem Inhalt des Dokumentes zustimmen.

Bei der Eintragung einer Einverleibung wird genau nach der im Abschnitt 6.3.1 beschriebenen Methode vorgegangen. Zu beachten ist dabei nur, dass das Dokument der Eintragung den oben erwähnten Anforderungen genügen muss.

6.3.3 Vormerkung

Die Vormerkung dient der Eintragung eines Rechtes, obwohl das Dokument noch nicht alle für die Einverleibung notwendigen Voraussetzungen erfüllt. Die Eintragung ist nur unter der Bedingung gültig, dass diese Erfordernisse noch erfüllt werden. Daher spricht man bei der Vormerkung auch von der bedingten Rechtserwerbung (§ 8). Die Mängel, die das Dokument haben darf, um eine Vormerkung noch zu ermöglichen, sind genau definiert. § 35 sagt, dass eine Vormerkung nur dann erfolgen darf, wenn das Dokument zwar den allgemeinen Erfordernissen (§§ 26, 27) nicht aber allen speziellen Erfordernissen (§§ 31 - 34) genügt.

Die Vormerkung benötigt für den tatsächlichen Rechtserwerb die Erfüllung einer Bedingung. Das Erfüllen der Bedingung heißt Rechtfertigung. Für die Rechtfertigung wird vom Grundbuchsgericht eine Frist vorgegeben, welche explizit angegeben wird:

§ 42. (1) Muß die Rechtfertigung im Prozeßwege geschehen, so ist die Klage binnen 14 Tagen nach dem Tage der Zustellung des Vormerkungsbeschlusses von dem Vormerkungswerber bei dem zuständigen Gerichte zu erheben.

(2) ...

§ 43. (1) Die Frist zur Erhebung der Rechtfertigungsklage ist in dem Vormerkungsbeschuß auszudrücken. Sie kann aus erheblichen Gründen verlängert werden.

(2) ...

Über die Länge der Frist entscheidet das Grundbuchsgericht. Nur wenn die Rechtfertigung auf dem Prozessweg zu erfolgen hat, ist die Frist mit 14 Tagen festgelegt. In allen anderen Fällen ist es möglich die Frist zu verlängern, wenn ‚erhebliche‘ Gründe vorliegen. Da weder festgelegt ist, was unter ‚erheblich‘ zu verstehen ist, noch bis zu welchem Limit verlängert werden darf, ist das Grundbuchsgericht in seiner Entscheidung nur wenig eingengt. Bei der Modellierung wurde wie in Abschnitt 5.4.3 beschrieben vorgegangen. Die Frist wurde generell auf 14 Tage festgelegt.

```
rechtseintrag Vo = Vormerkung (recht d) dt (incTage dt 14)
False False False
```

Innerhalb der Frist kann die Rechtfertigung erfolgen. Es hat also der Antragsteller einen Antrag auf Rechtfertigung eingebracht und die Eintragung wird (wenn alle Bedingungen erfüllt wurden) um die erfolgte Rechtfertigung ergänzt. Auf die automatische Löschung von Vor-

merkungen, welche nach einem vorgemerkten Eigentumserwerb durchgeführt wird, geht Abschnitt 6.5.1 näher ein.

Falls keine Rechtfertigung erfolgte, hat der Verpflichtete die Möglichkeit die Löschung der Vormerkung zu beantragen. Dies geschieht durch Markieren als gelöschte Eintragung. Diese Markierung darf allerdings nicht verwechselt werden mit einer abgewiesenen Vormerkung, da im Fall der abgewiesenen Eintragung kein neuerliches Ansuchen auf Eintragung möglich ist, während es nach Löschung wegen Fristversäumnis erlaubt ist.

§ 48. (1) Ist dagegen die Vormerkung nur aus dem Grunde gelöscht worden, weil die Rechtfertigungsklage nicht in gehöriger Zeit angebracht worden ist, so kann zwar abermals eine Vormerkung angesucht werden; diese äußert jedoch ihre rechtliche Wirksamkeit erst von dem Zeitpunkte der Überreichung des neuen Gesuches.

(2)...

```

rechtfertigen :: g d r t -> Nr -> d r t -> Datum -> g d r t
rechtfertigen g ez d dt =
  grundbuecher (rechtfHB (hauptbuch g) ez (recht d) dt)
    (d:urkundensammlung g)

rechtfHB :: h r t -> Nr -> r t -> Datum -> h r t
rechtfHB h ez r d = hauptbuecher (map rf' (gbEinlagen h))
  where rf' et = if (einlagezahl et) == ez
    then rechtfRecht et r dv else et

rechtfRecht :: e r t -> r t -> Datum -> e r t
rechtfRecht e r d = einlagen (einlagezahl e) (koerper e)
  (voLoeschen (map rf' (eintrliste e)))

where
  rf' et =
    if (istVormerkung et) && ((not.gerechtfertigt) et) &&
      ((eintragFrist et) >= d)
    then doRechtfertigen et else et
voLoeschen es =
  if istEigentum r
  then voDel es (belastete r)
    ((eintragDatum.(recht2Eintrag r)) es)
  else es

```

```

doRechtfertigen :: e r t -> e r t
doRechtfertigen e =
  if (istEinverleibung e) || (istEintrAnm e) then e
  else vormerkung (eintragRecht e) (eintragDatum e)
                 (eintragFrist e) (True) (eintragAbgewiesen e)
                 (eintragGeloescht e)

```

6.3.4 Anmerkung

Anmerkungen im Grundbuch sind nur in bestimmten Fällen zulässig. Die wichtigsten Situationen sind die Anmerkung der persönlichen Verhältnisse und die Anmerkung der Rangordnung. Die Anmerkung der persönlichen Verhältnisse ist dann von Bedeutung, wenn der Eigentümer nicht mehr frei über die Liegenschaft verfügen darf, also beispielsweise bei einem eingeleiteten Konkurs. Die Anmerkung der Rangordnung dient dazu, Sicherheit über den Rang zu haben, in dem eine Eintragung stehen wird. Wenn man beispielsweise bei einer Bank einen Kredit aufnimmt und als Sicherstellung die Eintragung eines Pfandrechtes im Grundbuch zulässt, so verlangt die Bank eine Rangordnung. Damit ist für die Bank sichergestellt, dass das Pfandrecht nach der Eintragung den Rang besitzt unter dem momentan die Rangordnung eingetragen ist.

§ 53. (1) Der Eigentümer ist berechtigt, die bücherliche Anmerkung für eine beabsichtigte Veräußerung oder Verpfändung zu verlangen, um die bücherliche Rangordnung vom Zeitpunkte dieses Ansuchens für die infolge dieser Veräußerung oder Verpfändung einzutragenden Rechte zu begründen. Hierbei macht es keinen Unterschied, ob die Verpfändung für eine Schuld, deren Betrag anzugeben ist, oder für einen Höchstbetrag (§ 14 Abs. 2) erfolgt und ob die Urkunde, auf Grund deren die aus der Veräußerung oder Verpfändung sich ergebenden Rechte eingetragen werden sollen, vor oder nach dem Ansuchen um die Anmerkung errichtet worden ist. Auf Antrag ist in die Anmerkung der beabsichtigten Verpfändung die Bedingung aufzunehmen, daß die Eintragung eines Pfandrechtes im Range der Anmerkung nur für dieselbe Forderung zulässig ist, für die entweder im Zeitpunkt des Einlangens des Ansuchens um Eintragung des Pfandrechts bereits im Range einer anderen Anmerkung der beabsichtigten

Verpfändung, der eine Bedingung nicht beigesetzt ist, die Eintragung eines anderen Pfandrechtes bewilligt worden ist oder gleichzeitig mit der Bewilligung der Eintragung des Pfandrechtes im Range einer anderen Anmerkung der beabsichtigten Verpfändung, der eine Bedingung nicht beigesetzt ist, die Eintragung eines anderen Pfandrechtes bewilligt wird.

(2) *Mit gleicher Rechtsfolge kann ein Hypothekargläubiger die Anmerkung der beabsichtigten Abtretung oder Löschung seiner Forderung verlangen.*

(3) *Die Anmerkungen solcher Gesuche können jedoch nur dann bewilligt werden, wenn nach dem Grundbuchsstand die Einverleibung des einzutragenden Rechtes oder die Löschung des bestehenden Rechtes zulässig wäre und wenn die Unterschrift der Gesuche gerichtlich oder notariell beglaubigt ist. Die Bestimmungen des § 31 Abs. 3 bis 5 sind anzuwenden.*

§ 54. *Von dem Beschluß, mit dem das Gesuch bewilligt wird, darf nur eine Ausfertigung erteilt werden; diese ist mit der Bestätigung der vollzogenen Anmerkung zu versehen.*

```
rangordnung :: Datum -> Nr -> g d r t -> (RONr,g d r t)
rangordnung dt nr g = ((roNummer.recht) ro,gb)
  where
    ro = makeRO (eigentuemervon g nr) nr (urkundensammlung g)
    gb = eintragenGB g An ro dt
```

Die Rangordnung wird auf Antrag des Eigentümers vom Grundbuch ausgestellt und im Grundbuch eingetragen (§ 53). Es existiert nur ein einziges Exemplar, welches dem Eigentümer ausgehändigt wird (§ 54). Daher ist das Ergebnis der Funktion nicht nur ein geändertes Grundbuch sondern auch ein Rangordnungsdokument. Die Gültigkeit einer Rangordnung läuft nach einem Jahr ab (§ 55). Danach wird sie automatisch gelöscht (siehe Abschnitt 6.5.2)

Die Rangordnung kann bei einem Grundbuchsgesuch mit übergeben werden. Das hat zur Folge, dass die Einverleibung des Rechtes mit dem Datum erfolgt, zu dem die Rangordnung ausgestellt wurde. Wenn es sich bei dem einzutragenden Recht um ein Eigentumsrecht handelt, ist es beispielsweise möglich, alle Rechte zu löschen, die nach der Rangordnung eingetragen

wurden. Sollte es sich bei dem einzutragenden Recht um ein Pfandrecht handeln ist sichergestellt, dass im Falle einer Veräußerung der Liegenschaft der Berechtigte in der Reihe der Entschädigungen im Rang der Rangordnung steht. Das ist immer dann von Bedeutung, wenn bei der Veräußerung nicht genügend Geld eingenommen wird, um alle Pfandrechte abzugelten, da dann nach der Rangordnung vorgegangen wird.

6.4 Abfragen

Abfragen werden nur an wenige Stellen im Grundbuchgesetz behandelt. Eine davon definiert allerdings das Öffentlichkeitsprinzip:

- § 7. (1) *Das Grundbuch ist öffentlich.*
 (2) *Jedermann kann das Grundbuch in Gegenwart eines Grundbuchsbeamten einsehen und Abschriften oder Auszüge daraus erheben; der Grundbuchsführer hat sie zu erteilen.*

Dieser Paragraph sagt aus, dass das Grundbuch eine öffentliche Einrichtung ist und jedermann berechtigt, Einsicht zu nehmen und Abschriften zu erheben. Es gibt also Operationen, mit deren Hilfe die Daten aus dem Grundbuch gelesen werden können. Eine dieser Operationen könnte beispielsweise `eintragungenVEinl` heißen und alle in einer Grundbuchseinlage vorhandenen Eintragungen lesen. Als ersten Schritt muss eine solche Operation prüfen, ob die Abfrage überhaupt erlaubt ist. Das geschieht durch Aufruf der Operation `abfrageErlaubt`, welche `True` zurückgibt, wenn die Abfrage erlaubt ist und `False`, wenn sie nicht erlaubt ist. § 7 definiert das Ergebnis der Operation als Konstante mit dem Wert `True` und ermöglicht dadurch den uneingeschränkten Zugriff für Jedermann.

```
abfrageErlaubt :: g d r t -> Person -> Bool
abfrageErlaubt g _ = True

eintragungenVEinl :: g d r t -> Nr -> Person ->
                  [Eintragung r t]
eintragungenVEinl g ez p =
  if (abfrageErlaubt g p)
  then (map chg.sort.eintrliste.rd.gbEinlagen.hauptbuch) g
  else error "Abfrage nicht erlaubt"
where
  rd = head.filter ((ez ==).einlagezahl)
```

```

chg ein = if (istVormerkung ein) && (gerechtfertigt ein)
  then einverleibung (eintragRecht ein)
    (eintragDatum ein)
    (eintragGeloescht ein)
  else ein

```

Das Lesen der Eintragungen geschieht durch den Ausdruck `(map chg.sort.eintrliste.rd.gbEinlagen.hauptbuch) g`, welcher mehrere Schritte hintereinander durchführt:

- Lesen der Daten des Hauptbuches
- Lesen der Daten der Grundbuchseinlagen
- Bestimmen der gewünschten Grundbuchseinlage
- Lesen der Einträge
- Sortieren entsprechend der Rangordnung
- Umwandeln von gerechtfertigten Vormerkungen in Einverleibungen (da die rechtliche Wirkung nach der Rechtfertigung gleich ist)

6.5 Automatische Abläufe

Der Grundsatz im Grundbuch ist, dass Eintragungen nur auf Antrag geschehen. Diese Regel ist in § 76 festgehalten:

§ 76. Das Grundbuchsgericht ordnet, außer den in diesem oder in einem anderen Gesetz bestimmten Fällen, Eintragungen nicht von Amts wegen, sondern nur auf Ansuchen von Parteien oder Behörden an.

Es wird aber gleichzeitig ermöglicht, dass Ausnahmen geschaffen werden. Folgende Tätigkeiten werden von Amts wegen durchgeführt:

- § 49: Löschen von vorgemerkten Rechten, die gegen ein vorgemerktes Eigentumsrecht eingetragen wurden, das aber abgewiesen wurde (bzw. gegen den alten Eigentümer, wenn das Eigentumsrecht gerechtfertigt wurde).

§ 49. (1) Wenn gegen den, der als Eigentümer einer Liegenschaft einverleibt ist, die Vormerkung des Eigentumsrechtes bewirkt worden ist, können sowohl gegen den einverlebten als gegen den vorge-

merkten Eigentümer weitere Eintragungen zwar bewilligt werden, doch hängt deren rechtlicher Bestand davon ab, ob die Vormerkung des Eigentumsrechtes gerechtfertigt wird oder nicht.

(2) Wird die Vormerkung gerechtfertigt, so sind bei Eintragung der Rechtfertigung zugleich alle Eintragungen von Amts wegen zu löschen, die gegen den einverleibten Eigentümer nach dem Einlangen desjenigen Einschreitens erwirkt worden sind, auf das das Eigentumsrecht vorgemerkt worden ist.

(3) Wird dagegen die Vormerkung des Eigentumsrechtes gelöscht, so sind zugleich alle in bezug auf diese Vormerkung vorgenommenen Eintragungen von Amts wegen zu löschen.

(4) ...

- § 57 Abs.2: Löschen von Rangordnungen, die nicht innerhalb eines Jahres verwendet wurden

§ 57. (1) ...

(2) Wird das Eintragungsgesuch nicht vor dem Ende der festgesetzten Frist angebracht oder wird der Betrag, für den die Anmerkung der Rangordnung erfolgt ist, bis zum Ende dieser Frist nicht erschöpft, so wird die Anmerkung unwirksam und ist von Amts wegen zu löschen.

(3) ...

- § 130: Löschen von Eintragungen, deren Inhalt von Gesetz wegen nicht Gegenstand einer bürgerlichen Eintragung sein kann (Löschen unzulässiger Eintragungen)

§ 130. Ergibt sich aus einer Eintragung, daß ihr Inhalt nach dem Gesetz nicht Gegenstand einer grundbücherlichen Eintragung sein kann, so ist sie von Amts wegen als unzulässig zu löschen. Die Vorschriften des ersten bis dritten und fünften Hauptstückes, insbesondere über die Verständigung der Beteiligten und den Rekurs, sind entsprechend anzuwenden.

- § 131: Löschen gegenstandsloser Eintragungen

§ 131.(1) Ist eine Eintragung gegenstandslos, so kann sie das Grundbuchsgericht gemäß den §§ 132 bis 135 von Amts wegen löschen.

(2)...

Die automatischen Abläufe sind zwar eigentlich Aktionen, die im Grundbuch stattfinden, die eigentlichen Arbeitsschritte finden aber auf Ebene der Eintragungen statt. Manche der Aktionen benötigen das aktuelle Datum, welches daher als Parameter zu übergeben ist. Daher muss das Datum bei Aufruf der übergeordneten Funktion `automation` angegeben werden.

```
class (Rechte r t) => Eintragungen e r t where
  automation :: Datum -> [e r t] -> [e r t]
  automation dt =
    autoLoeschen.illegaleEintragungen.delRO dt.delVO dt
```

Zusätzlich gibt es noch die automatische Löschung von Grundbucheinlagen. Dies geschieht dann, wenn sämtliche Grundstücke von der Einlage abgeschrieben wurden, wenn also die Grundstückseinlage ihren Zweck der Repräsentation von Grundstücken verliert.

6.5.1 Löschen von Vormerkungen

§ 49 regelt die Vorgangsweise bei einem vorgemerkten Eigentumsrecht. Das hier gelöste Problem besteht darin, dass Rechte nur gegen den Eigentümer eingetragen werden dürfen. Wenn nun ein Eigentumsrecht vorgemerkt ist, so ist der Begünstigte noch nicht Eigentümer der Liegenschaft. Das wird er erst, wenn die Vormerkung gerechtfertigt wird. Man darf aber nicht davon ausgehen, dass sie in jedem Fall gerechtfertigt wird. Es muss daher möglich sein, sowohl gegen den ursprünglichen Eigentümer der Liegenschaft, als auch gegen den vorgemerkten Eigentümer Rechte vorzumerken. Die Rechtfertigung dieser nachfolgenden Rechte erfolgt nun entweder automatisch mit der Rechtfertigung des Eigentumsrechtes oder mit der nicht erfolgten Rechtfertigung desselben. Wenn nun das Eigentumsrecht nicht gerechtfertigt wird, müssen die vorgemerkten Rechte zu diesem Eigentumsrecht gelöscht werden. Bei erfolgter Rechtfertigung müssen die gegen den alten Eigentümer vorgemerkten Rechte gelöscht werden. Folgendes Beispiel soll den Vorgang erläutern:

Herr Müller ist Eigentümer der Grundbucheinlage 1. Er will sein Grundstück an Herrn Maier verkaufen und verfasst einen Kaufvertrag. Herr Maier will das Eigentumsrecht einverleiben, darf es aber nur vormerken (da einer der in den §§ 31-34 GBG55 angegebenen Punkte nicht erfüllt ist). Es ist nun möglich, sowohl gegen Herrn Müller als auch gegen Herrn

Maier Eintragungen vorzumerken. Das könnten beispielsweise Pfandrechte sein. Die Grundbuchseinlage würde dann so aussehen:

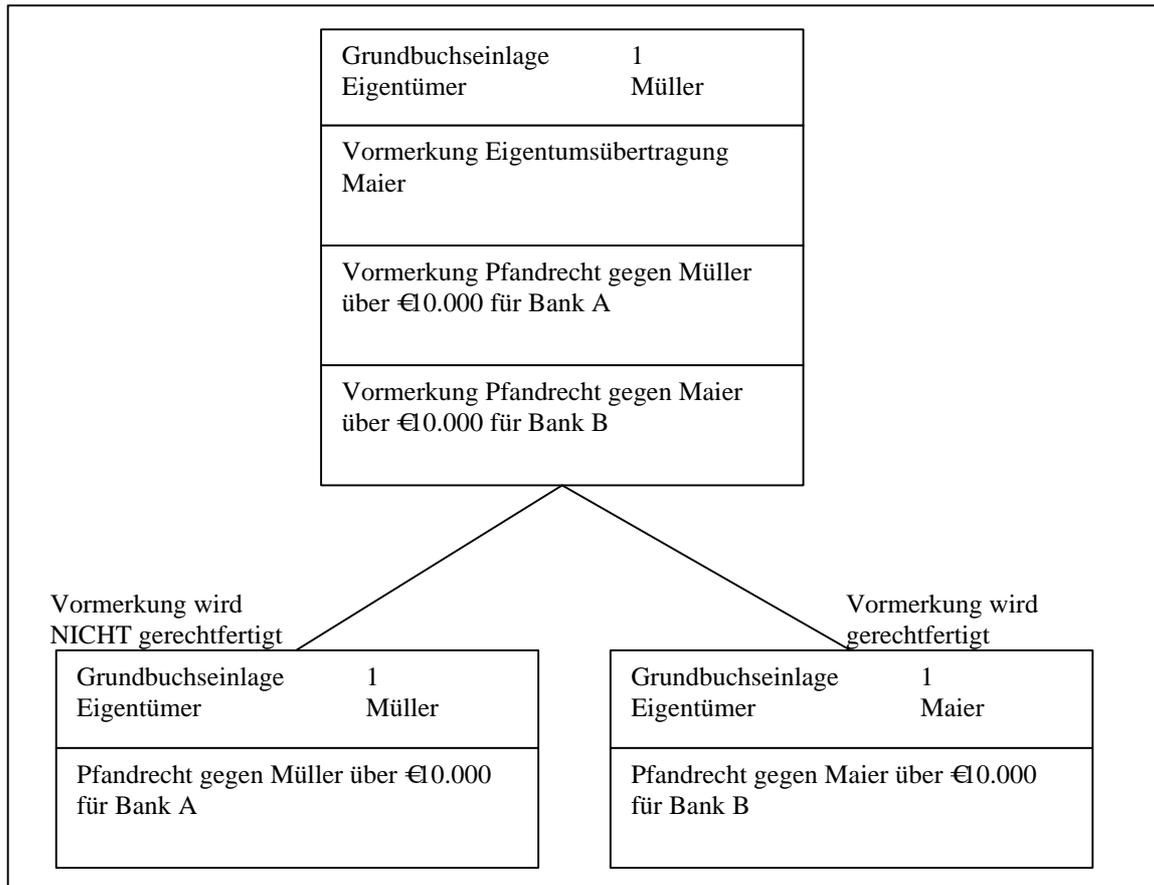


Abbildung 9: Automatisches Löschen von Vormerkungen

Wenn die Vormerkung der Eigentumsübertragung nicht gerechtfertigt wird, bleibt Herr Müller Eigentümer der Liegenschaft. Sowohl die Vormerkung der Eigentumsübertragung, als auch die Vormerkung des Pfandrechtes gegen Herrn Maier werden automatisch gelöscht. Gleichzeitig wird die Vormerkung des Pfandrechtes gegen Herrn Müller gerechtfertigt. Wenn die Vormerkung der Eigentumsübertragung gerechtfertigt wird, so wird Herr Maier Eigentümer und das Pfandrecht gegen Herrn Müller wird gelöscht. Gleichzeitig wird das Pfandrecht gegen Herrn Maier gerechtfertigt.

Es handelt sich also um eine Löschung von Vormerkungen, die nach einem bestimmten Datum gegen eine bestimmte Person eingetragen wurden. Das Datum ist dabei das Datum der Vormerkung des Eigentumsrechtes und die Person ist davon abhängig, ob die Rechtfertigung erfolgt ist, oder nicht.

voDel :: [e r t] -> [Person] -> Datum -> [e r t]

```

voDel [] _ _ = []
voDel (e:es) ps dt =
  if (istVormerkung e) && ((eintragDatum e) > dt) &&
    (((sort.belastete.eintragRecht) e) == sort ps)
  then voDel es ps dt else e:voDel es ps dt

```

6.5.2 Löschen von Rangordnungen

Die Eigentümer von Liegenschaften können sich eine Rangordnung ausfertigen lassen (vergleiche Abschnitt 6.3.4). Diese Rangordnung wird nur in einem Exemplar ausgestellt und gilt exakt ein Jahr. Nach einem Jahr verliert sie ihre Gültigkeit, wenn sie in der Zwischenzeit nicht verwendet wurde. Wenn die Rangordnung ihre Gültigkeit verliert wird sie von Amts wegen gelöscht (§ 57 Abs. 2). Eine Operation, welche die Löschung automatisch durchführt, muss also für jede vorhandene Anmerkungen der Rangordnung überprüfen, ob die Rangordnung verwendet wurde bzw. (falls sie nicht verwendet wurde), ob sie abgelaufen ist. Die Operation führt dazu vier Tests durch:

- Ist die bearbeitete Eintragung eine Anmerkung?
- Ist die bearbeitete Anmerkung eine Rangordnung?
- Wurde die Nummer der Rangordnung bei der Eintragung eines Rechtes verwendet?
- Ist seit der Eintragung ein Jahr vergangen?

Jeder dieser Tests ist ein Abbruchkriterium für die Löschung. Nur wenn alle vier Bedingungen erfüllt sind, wird die Anmerkung der Rangordnung gelöscht.

```

delRO :: Datum -> [e r t] -> [e r t]
delRO dt el = del' el where
  del' (e:es) =
    if istEintrAnm e && (istRangordnung.eintragRecht) e &&
      notElem ((roNummer.eintragRecht) e)
      ((map roNummer.filter(not.istRangordnung).
        map eintragRecht) el) &&
      incTage (eintragDatum e) 365 > dt
    then eintragungLoeschen e:del' es
    else e:del' es

```

6.5.3 Löschen illegaler Eintragungen

§ 130 regelt die Löschung von unzulässigen Eintragungen. Er lautet:

§ 130. Ergibt sich aus einer Eintragung, daß ihr Inhalt nach dem Gesetz nicht Gegenstand einer grundbücherlichen Eintragung sein kann, so ist sie von Amts wegen als unzulässig zu löschen. Die Vorschriften des ersten bis dritten und fünften Hauptstückes, insbesondere über die Verständigung der Beteiligten und den Rekurs, sind entsprechend anzuwenden.

Die Problematik dieser Regelung wurde bereits in Abschnitt 5.4.3 behandelt. Im Modell wurde angenommen, dass die Eintragungen alle zulässig sind. Die Operation `eintragRechtmaessig` liefert also unabhängig vom geprüften Eintrag den Wert `True`. Der Löschvorgang selbst wird von der Operation `illegaleEintragungen` ausgeführt. Die Operation ruft für jede Eintragung `eintragRechtmaessig` auf. Je nach Rückgabewert wird die Eintragung gelöscht oder unverändert gelassen.

```
eintragungRechtmaessig :: e r t -> Bool
eintragungRechtmaessig _ = True

illegaleEintragungen :: [e r t] -> [e r t]
illegaleEintragungen = map f
  where f e = if eintragungRechtmaessig e then e
             else eintragLoeschen e
```

6.5.4 Löschen gegenstandsloser Eintragungen

§§ 131 -135 regeln gegenstandslose Eintragungen:

§ 131.(1) Ist eine Eintragung gegenstandslos, so kann sie das Grundbuchgericht gemäß den §§ 132 bis 135 von Amts wegen löschen.

(2) Eine Eintragung ist gegenstandslos, soweit das ihren Gegenstand bildende Recht oder das Recht, auf das sie sich bezieht,

a) nicht besteht oder aus tatsächlichen Gründen dauernd nicht ausgeübt werden kann,

b) verjährt ist,

c) für den Berechtigten einen lediglich wirtschaftlichen Wert darstellt, der 1.500 S, bei wiederkehrenden Leistungen

500 S jährlich, nicht übersteigt, sofern die Eintragung des Rechtes vor dem 1. Mai 1945 erfolgt ist.

(3) Im Falle des Abs. 2 lit. c bedarf es zur Löschung eines Pfandrechtes nicht der Zustimmung des Eigentümers, dem das Verfügungsrecht nach § 469 ABGB. zusteht.

(4) Abs. 2 lit. c gilt auch für Pfandrechte, bei denen gemäß Artikel 3 der Grundbuchsnovelle, BGBl. Nr. 4/1930, ein Antrag auf Aufrechterhaltung angemerkt ist.

§ 132.(1) Das Grundbuchsgericht soll das Verfahren zur Löschung gegenstandsloser Eintragungen einleiten, wenn besondere äußere Umstände (zum Beispiel Umschreibung der Grundbuchseinlage wegen Unübersichtlichkeit, Teilveräußerung oder Neubelastung des Grundstückes, Anregung seitens eines Beteiligten) hinreichenden Anlaß dazu geben und Grund zu der Annahme besteht, daß die Eintragung gegenstandslos ist.

(2) Das Grundbuchsgericht entscheidet nach freiem Ermessen, ob das Lösungsverfahren einzuleiten und durchzuführen ist; diese Entscheidung ist unanfechtbar.

§ 133.(1) Voraussetzung für die Löschung ist, dass

a) die Gegenstandslosigkeit der Eintragung offenkundig oder durch öffentliche oder gerichtlich oder notariell beglaubigte Urkunden nachgewiesen ist, oder daß, falls dies nicht zutrifft,

b) dem Betroffenen eine Lösungsankündigung unter kurzer Bekanntgabe des Grundes zugestellt ist und er nicht binnen einer vom Grundbuchsgericht zugleich zu bestimmenden Frist Widerspruch erhoben hat, oder daß, falls auch nach lit. b nicht verfahren werden kann, insbesondere wenn Widerspruch erhoben ist,

c) durch einen mit Gründen versehenen Beschluß rechtskräftig festgestellt ist, daß die Eintragung gegenstandslos ist.

(2) Kann die Löschung nicht sogleich angeordnet werden, so ist die Einleitung des Verfahrens im Grundbuch anzumer-

ken. Die Anmerkung hat die Wirkung, dass spätere Eintragungen die Löschung nicht hindern. Sie ist zu löschen, wenn die gegenstandslose Eintragung gelöscht oder von der Fortsetzung des Verfahrens Abstand genommen wird. Einer Verständigung der Beteiligten von der Anordnung dieser Anmerkung und ihrer Löschung bedarf es nicht. Gegen diese Anordnungen ist ein Rechtsmittel nicht zulässig.

§ 134. Für das Verfahren gelten sinngemäß die Vorschriften des dritten Hauptstückes. Die Vorschriften über das Verfahren außer Streitsachen sind, soweit erforderlich, ergänzend heranzuziehen. Dabei gilt folgendes:

- a) Eine Verweisung der Beteiligten auf den Rechtsweg oder das Verwaltungsverfahren (§ 2 Z. 7 des Kaiserlichen Patentgesetzes vom 9. August 1854, RGBl. Nr. 208) findet nicht statt;*
- b) die Löschungsankündigung (§ 133 Abs. 1 lit. b) kann nicht durch öffentliche Bekanntmachung zugestellt werden;*
- c) ist die Person des Beteiligten, dem zugestellt werden soll, unbekannt, so sind die Vorschriften über die Zustellung durch öffentliche Bekanntmachung sinngemäß anzuwenden;*
- d) die Rechtsmittel gegen die Entscheidungen des Grundbuchgerichtes, mit denen die Löschung gegenstandslos gewordener Eintragungen angeordnet wird, richten sich nach dem siebenten Abschnitt des dritten Hauptstückes; im übrigen gelten für die Anfechtung von Entscheidungen die Vorschriften über das Verfahren außer Streitsachen. Gegen die Löschungsankündigung (§ 133 Abs. 1 lit. b) ist kein Rechtsmittel gegeben.*

§ 135. Ist ein Beteiligter durch eine nach den §§ 131 ff. bewilligte Löschung in seinem bürgerlichen Rechte verletzt, so kann er im Prozeßwege die Wiederherstellung des vorigen bürgerlichen Standes begehren. Die Vorschriften der §§ 61 ff. sind sinngemäß anzuwenden.

Am wichtigsten ist dabei § 131. Dieser Paragraph definiert, was eine gegenstandslose Eintragung ist und was mit ihr passiert. Er erlaubt dem Grundbuchsgericht beispielsweise, auch

ohne Antrag tätig zu werden. § 132 präzisiert dann noch einmal, woran man erkennt, dass eine Eintragung gegenstandslos ist und streicht heraus, dass das Grundbuch nach freiem Ermessen entscheiden darf, ob das Verfahren eingeleitet wird, oder nicht. § 133 fasst die verfahrensmäßigen Voraussetzung für die Durchführung der Löschung zusammen. § 134 liefert Vorschriften für die Verfahrensweise und verweist für nähere Details auf das Verfahren außer Streitsachen, das ergänzend herangezogen werden soll.

Probleme bereitet wieder die Definition einer gegenstandslosen Eintragung. Im Gesetz sind folgende Fälle für gegenstandslose Rechte definiert:

- Recht besteht nicht oder kann nicht ausgeübt werden
- Recht ist verjährt
- Recht stellt einen wirtschaftlichen Wert von unter 1.500,- ATS (oder 500,- ATS bei wiederkehrenden Leistungen) dar wenn die Eintragung des Rechtes vor dem 1.Mai 1945 erfolgte.

In einem Modell des Grundbuches alleine kann nicht geprüft werden, ob ein Recht besteht oder nicht. Die Existenz kann nur in der Realität nachgewiesen werden, nicht jedoch im Grundbuch. Daher muss für diesen Nachweis sowohl Grundbuch als auch Realität modelliert werden. Einen Ansatz dafür liefert (Bittner 1998).

Für den zweiten Punkt muss es möglich sein, zu jedem Recht auch eine Verjährungsfrist anzugeben. Falls es sich um ein Recht ohne Verjährungsfrist handelt, so wird einfach der Wert ~~keine~~ Verjaehrung eingetragen. Die eigentliche Überprüfung besteht dann aus zwei Teilen. Zunächst wird überprüft, ob der spezielle Wert bei der Verjährungsfrist eingetragen ist. Steht dort dieser Wert, so ist das Recht nicht verjährt. Andernfalls wird das angegebene mit dem aktuellen Datum verglichen. Sollte das aktuelle Datum nach dem angegebenen Datum sein, so ist das Recht verjährt.

Der dritte Punkt verursacht ähnliche Probleme wie der erste. Auch hier treten Auswirkungen in der Realität als Entscheidungsmerkmal auf. Der wirtschaftliche Wert kann nur von einem Fachmann in der Realität bestimmt werden. Daher ist es nur möglich, anhand des Betrages, der für das Recht eingetragen ist, zu urteilen. Man würde auf diese Art beispielsweise Pfandrechte finden, die nur für geringe Werte stehen. Eine Alternative wäre die Eintragung des Wertes für jedes Recht. Wie dieser Wert bestimmt wird ist für das Modell nicht relevant, es können aber die Auswirkungen gezeigt werden.

```

gegenstandslos :: e r t -> Bool
gegenstandslos e = ((betrag.eintragRecht) e /= 0) &&
                    ((betrag.eintragRecht) e < 1500) &&
                    ((eintragDatum e) < (newdatum 1 5 1945))

```

Das automatische Löschen funktioniert wie beim Löschen illegaler Eintragungen. Der einzige Unterschied ist die Abfrage auf Gegenstandslosigkeit der Eintragung.

```

autoLoeschen :: [e r t] -> [e r t]
autoLoeschen es = map f es
  where f e = if gegenstandslos e then eintragLoeschen e
              else e

```

6.5.5 Löschen leerer Grundbuchkörper

§ 3 Abs. 3 bestimmt, dass Grundbuchkörper gelöscht werden sollen, wenn sie keine Grundstücke mehr enthalten:

- § 3. (1) ...
 (2) ...
 (3) *Wenn alle in einer Grundbuchseinlage eingetragenen Liegenschaften abgeschrieben worden sind (§ 11) oder wenn sie aufgehört haben, ein Gegenstand des Grundbuches zu sein, ist die Einlage zu löschen.*

Es sind zwei Operationen notwendig. Eine Operation prüft ob eine Grundbuchseinlage gelöscht werden muss und die andere Operation führt diese Löschung durch. Die Funktion `toDelK` übernimmt die erste Aufgabe. Es wird einfach die Liste der Grundstücke ausgelesen und die Länge der Liste bestimmt. Das Ergebnis wird dann mit Null verglichen. Die zweite Ausgaben übernimmt die Operation `autoDelEinl`. Diese Operation löscht alle Grundbuchseinlagen, für die `toDelK` den Wert `True` liefert. Im Gegensatz zur Löschung der Rechte wird hier der Datensatz wirklich gelöscht um zu zeigen, dass auch so vorgegangen werden kann.

```

toDelK :: k -> Bool
toDelK = ((0==).length.grundstuecke)

autoDelEinl :: h r t -> h r t
autoDelEinl h = hauptbuecher (filter (not.toDelK.koerper)
                                  (gbEinlagen h))

```

6.6 Zusammenfassung

Es hat sich gezeigt, dass eine algebraische Modellierung des Grundbuchgesetzes möglich ist. Die Definitionen des Gesetzestextes konnten in abstrakte Definitionen übergeführt werden. Das Ergebnis der Modellierung ist eine algebraische Beschreibung des Gesetzesinhaltes. Die Gliederung des Gesetzes in einzelne Paragraphen ist bei der Modellierung teilweise verloren gegangen, da oft mehrere Paragraphen ein Axiom definieren oder einzelne Paragraphen viele Axiome beeinflusst haben.

Schwierigkeiten haben sich vor allem dort ergeben, wo Ermessensspielraum gelassen wurde. Dieser Aspekt definiert Interaktion zwischen den Vorschriften und dem Anwender des Gesetzes. Der Anwender ist aber Teil der realen Welt und die Arbeit soll nur das Gesetz selbst beschreiben. Eine einfache Lösung ist es daher, die Entscheidung vorzudefinieren. Das kann durch konstante Prüffunktionen geschehen, die unabhängig vom Datensatz immer dieselbe Entscheidung liefern oder dadurch, dass zu jedem Element auch die zu treffende Entscheidung gespeichert wird.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die Arbeit beschrieb eine Methode, mit der man Gesetze in Algebren umwandeln kann. Kapitel 2 zeigte, welche Methoden bisher verwendet werden, um Gesetze im Computer zu verarbeiten. Kapitel 3 beschrieb anschließend algebraische Modelle, ihre Struktur und ihre Entwicklung mit Hilfe der Programmiersprache Haskell. Kapitel 4 zeigte Parallelen und Unterschiede zwischen Gesetzen und algebraischen Modellen auf. Davon ausgehend beschrieb Kapitel 5 eine Methodik für die Umwandlung eines Gesetzes in eine Algebra. Kapitel 6 zeigte anhand von Beispielen aus dem Allgemeinen Grundbuchgesetz, dass die Umwandlung möglich ist.

Gesetze und algebraische Modelle sind in Ihrem prinzipiellen Aufbau ähnlich. Sowohl Gesetze als auch Algebren beschreiben Modelle in abstrakter Form. Gesetze definieren allgemeine Rechtsfolgen für abstrakt definierte Situationen. Für das so beschriebene System werden die Teile festgelegt und die erlaubten Arten der Interaktion vorgegeben. Algebren definieren ebenfalls in abstrakter Weise Systeme durch Vorgabe der anwendbaren Funktionen und Angabe von Axiomen für diese Funktionen.

Es hat sich allerdings auch gezeigt, dass es Unterschiede zwischen Gesetzen und Algebren gibt. Die bei algebraischen Modellen benutzten Begriffe sind immer eindeutig definiert. In Gesetzestexten hingegen werden manchmal absichtlich unscharfe Formulierungen verwendet um bei der Anwendung der Gesetze einen Spielraum zu haben. In der Praxis der Rechtsprechung ist dieser Spielraum wichtig um Gesetze nicht ständig ändern zu müssen. Wenn ein Paragraph beispielsweise Grundflächen behandelt, deren Wert geringer ist als die Kosten eines standardmäßigen Verfahrens, so kann man von ‚geringwertigen Flächen‘ sprechen. Die exaktere Definition ‚Flächen mit einem Wert unter €1.000,-‘ hat den Nachteil, dass sie laufend an die Inflation angepasst werden müsste. Das bedeutet, dass man das Gesetz jedes Jahr ändert, was vermeidbaren Arbeitsaufwand bedeutet. Daher sind in manchen Fällen Begriffe in Gesetzen wesentlich weniger streng, als es für ein algebraisches Modell notwendig wäre.

Zusätzlich gibt es in Gesetzen auch Regelungen die für ein algebraisches Modell nur dann notwendig wären, wenn man den gesamten Verwaltungsapparat mitmodelliert. § 140 GBG⁵⁵ legt fest, dass das Bundesministerium für Justiz mit der Vollziehung des Gesetzes betraut wird. Ein algebraisches Modell benötigt diese Information nicht. Wichtig wird sie erst, wenn man das System der Ministerien mitmodelliert. Alle administrativen Paragraphen konnten bei der

Modellierung eines einzelnen Gesetzes weggelassen werden. Erst beim Einbau dieses Systems in einen größeren Kontext werden diese Paragraphen wichtig und müssen ebenfalls beachtet werden.

Wir haben die Methodik gesehen, mit der man aus einem Gesetz ein mathematisches Modell schaffen kann. Da ein für das Modell relevanter Paragraph auf eine von drei Arten etwas zum Modell beitragen kann (Definition der Objekte, Signatur von Funktionen, Axiome für Funktionen), wird das Modell auch in drei Stufen gebaut. In einem ersten Schritt werden die behandelten Objekte erfasst und zueinander in Relation gebracht. Danach werden für jedes Objekt die notwendigen Funktionen ermittelt und die Signaturen zu den Objekten hinzugefügt. Schließlich werden die Funktionen durch Axiome realisiert. Parallel mit der Entwicklung des Modells muss eine einfache Realisierung des Modells geschaffen werden, um die Axiome sofort überprüfen zu können. Nach Abschluss der dritten Stufe hat man ein komplettes Modell.

7.2 Diskussion der Ergebnisse

Es hat sich gezeigt, dass beim Österreichischen Grundbuchsgesetz eine Formalisierung möglich ist. Das Ergebnis ist ein algebraisches Modell des Gesetzes, bei dem die Arbeitsweise durch die Zusammenfassung zu kleinen Blöcken (den Klassen) leicht nachvollziehbar ist. Da das Modell in Haskell geschrieben wurde, ist es auch ausführbar und damit überprüfbar. Man kann dieses Modell somit auch als Kern eines Grundbuchsprogramms verwenden.

Die Umwandlung des Grundbuchsgesetzes war bei den meisten Paragraphen einfach zu realisieren. Die Paragraphen, bei denen eine Umwandlung nicht möglich war, enthielten entweder administrative Informationen, oder Informationen, die für das Modell keinen Einfluss hatten. Das resultierende Modell hat gegenüber dem reinen Gesetzestext einige Vorteile:

- Funktionen, die einen bestimmten Teil des Modells beschreiben, stehen nahe beisammen. Im Gesetzestext kann es vorkommen, dass eine Vorgangsweise von mehreren Paragraphen beschrieben wird, welche an den unterschiedlichsten Stellen des Gesetzes stehen. Das algebraische Modell fasst diese Paragraphen zusammen und beschreibt den Vorgang an einer einzigen Stelle.
- Der allgemeine Fall der Bearbeitung und die Ausnahmeregelungen werden nacheinander beschrieben. Im Gesetz werden zunächst für alle Teile des Modells die allgemeinen Vorschriften behandelt und dann sämtliche Spezialfälle beschrieben. Das algebraische Modell beschreibt für jede Funktion allgemeine Vorschrift und Ausnahmen direkt

untereinander. Daher werden die Zusammenhänge zwischen allgemeiner und spezieller Lösung deutlicher und die Unterschiede in der Bearbeitung klarer.

- Das algebraische Modell besteht aus vielen kleinen Teilen, die leicht zu verstehen sind. Das algebraische Modell gliedert sich in leicht begreifbare Teile wie die Grundbucheinlage und behandelt alle Aspekte dieser Teile an einer Stelle. Daher bleiben die Beschreibungen einfach und leicht nachvollziehbar. Im Gesetzestext ist diese Zusammenfassung nicht gegeben, wodurch es für den Leser schwieriger ist, die Zusammenhänge im Gesamtkontext zu sehen.

Die im Modell verwendeten Namen sind austauschbar. Die Namensvergabe hält sich an die im Grundbuchgesetz verwendeten Bezeichnungen um einen Vergleich mit dem Gesetzestext zu vereinfachen. Es ist aber ohne Schwierigkeiten möglich, allgemeine Bezeichnungen einzuführen. Die Definition der Bedeutung erfolgt durch Angabe der Operationen. Daher ist es einfacher, dieses Modell mit einem anderen Grundbuchmodell zu vergleichen weil die Elemente mit gleicher Funktion gefunden werden können. Wenn man die Gesetzestexte für einen solchen Vergleich heranzieht, ist es nicht so einfach möglich, die Funktionalität so einfach zu erkennen wodurch der Vergleich erschwert wird.

7.3 Ausblick

Zukünftige Arbeiten werden sich stärker im Bereich dieser Interaktionen engagieren und eine Verbindung zu der Dissertation von Steffen Bittner (Bittner 2001) herstellen. Speziell der Bereich der Interaktion mit Gerichten wirft noch einige Fragen auf. Es wäre interessant zu wissen, wie weit gerichtsspezifische Gesetze (wie beispielsweise die Geschäftsordnung für Gerichte) für eine Modellierung der Interaktion herangezogen werden müssen. Ein möglicher Ansatz wäre hier, dass das Gericht als gekapseltes Objekt betrachtet wird und immer nur so detailliert wie notwendig modelliert wird. Dieser Ansatz wurde in der vorliegenden Arbeit zwar verwendet, jedoch ohne den Hintergrund genau zu durchleuchten. Es wurde beispielsweise nie überprüft, ob weitere Gesetze wie beispielsweise das Zustellgesetz diese Interaktion beeinflussen. Somit ist auch die Frage ungeklärt, wo man aufhören kann, wenn man einen Ausschnitt der Rechtsordnung modellieren will.

Auch der Vergleich von Rechtsmodellen benötigt weitere Studien. Der Vergleich algebraischer Modelle von gesetzlichen Situationen scheint im Moment möglich zu sein. Ein solcher Vergleich wurde aber im Rahmen dieser Arbeit allerdings nicht durchgeführt. Da es sich

aber um eine interessante Anwendung handelt, sollte eine Untersuchung dieser Thematik stattfinden um zu überprüfen, ob die Hypothese haltbar ist.

Ferner wurde auch die Methodik der Nachführung des Modells nicht untersucht. Die Arbeit beleuchtet nur die Schaffung eines algebraischen Modells aus einem Gesetzestext um zu zeigen, dass das Prinzip umsetzbar ist. Da sich Gesetze allerdings ändern können wäre es interessant zu wissen, wie man das Modell im Falle einer Gesetzesänderung aktualisieren kann.

Ebenfalls interessant wäre die Übertragung eines algebraischen Modells in eine textliche Beschreibung. Man könnte zunächst versuchen das Modell zu verbessern und daraus einen Gesetzestext ableiten. Das könnte insofern von Bedeutung sein, weil man im Modell etwaige Schwachstellen der aktuellen Regelung erkennen und beseitigen kann. Wenn man aus dem verbesserten Modell dann ein Gesetz produzieren könnte, hätte das dann ein verbessertes Gesetz zur Folge wovon wiederum die Allgemeinheit profitieren würde.

Referenzen

- Abadi, M. and L. Cardelli (1996). A Theory of Objects. New York, Springer-Verlag.
- Athen, M. and J. Bruhn (1980). Rechnen und Mathematik. München, Deutschland, Springer-Verlag.
- Bird, R. (1998). Introduction to Functional Programming Using Haskell. Hemel Hempstead, UK, Prentice Hall Europe.
- Bittner, S. (1998). Die Modellierung eines Grundbuchsystems im Situationskalkül. Institut für Informatik. Leipzig, Universität.
- Bittner, S. (2001). An agent-based model of reality in a cadastre. Department of Geoinformation. Vienna, Technical University Vienna.
- Breu, R. (1991). Algebraic Specification Techniques in Object Oriented Programming Environments. Lecture Notes in Computer Science Vol.562. Berlin Heidelberg, Springer-Verlag. **562**: 228.
- Carnap, R. (1943). Formalization of Logic. Cambridge, England, Harvard University Press.
- Dittrich, R., P. Angst, et al. (1979). Das österreichische Grundbuchsrecht. Vienna, Manzsche Verlags- und Universitätsbuchhandlung.
- Dittrich, R. and H. Tades (1997). Das Allgemeine Bürgerliche Gesetzbuch. Wien, Manzsche Verlags- und Universitätsbuchhandlung.
- Egenhofer, M. and A. Frank (1992). "Object Oriented Modeling for GIS." Journal of the Urban and Regional Information Systems URISA **4**(2): 3-19.
- Ehrich, H.-D., M. Gogolla, et al. (1989). Algebraische Spezifikation abstrakter Datentypen. Stuttgart, B.G. Teubner.
- Frank, A. U. (1999). One step up the abstraction ladder: Combining algebras - from functional pieces to a whole. Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science (Int. Conference COSIT'99, Stade, Germany). C. Freksa and D. M. Mark. Berlin, Springer-Verlag. **1661**: 95-107.
- Frank, A. U. and W. Kuhn (1995). Specifying Open GIS with Functional Languages. Advances in Spatial Databases (4th Int. Symposium on Large Spatial Databases, SSD'95, in Portland, USA). M. J. Egenhofer and J. R. Herring, Springer-Verlag. **951**: 184-195.

- Frank, A. U. and W. Kuhn (1999). A Specification Language for Interoperable GIS. Interoperating Geographic Information Systems. M. F. Goodchild, M. J. Egenhofer, R. Fegeas and C. Kottman. Norwell, MA, Kluwer: 123-132.
- Guttag, J. V. and J. J. Horning (1978). "The Algebraic Specification of Abstract Data Types." Acta Informatica **10**(1): 27-52.
- Guttag, J. V., E. Horowitz, et al. (1978). "Abstract Data Types and Software Validation." Comm. ACM **21**(12): 1048-1064.
- Horebeek, I. V. and J. Lewi (1989). Algebraic Specifications in Software Engineering. Berlin Heidelberg, Springer-Verlag.
- Hudak, P., J. Peterson, et al. (1997). A Gentle Introduction to Haskell, Yale University.
- Kaiser, H., R. Mlitz, et al. (1985). Algebra für Informatiker. Wien, Österreich, Springer-Verlag Wien New York.
- Krejci, H. (1995). Privatrecht. Wien, Manzsche Verlags- und Universitätsbuchhandlung.
- Kühne, J. (1985). Verfassungs- und Verwaltungsrecht. Wien, Österreich, Hochschülerschaft Technische Universität Wien.
- Lachmayer, F. (1977). Grundzüge einer Normentheorie. Berlin, Duncker & Humblot.
- Lego, K. (1968). Geschichte des österreichischen Grundkatasters. Vienna, Bundesamt f. Eich- und Vermessungswesen.
- Lin, F.-T. (1998). "Many sorted algebraic data models for GIS." IJGIS **12**(8): 765-788.
- Loeckx, J., H.-D. Ehrich, et al. (1996). Specification of Abstract Data Types. Chichester, UK and Stuttgart, John Wiley and B.G. Teubner.
- Marent, K.-H. and G. Preisl (1994). Grundbuchsrecht. Wien, Linde.
- McCarthy, J. and P. J. Hayes (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence **4**. B. Meltzer and D. Michie. Edinburgh, Edinburgh University Press: 463-502.
- McCarty, L. T. (1989). A Language for Legal Discourse. Second International Conference on Artificial Intelligence and Law, Vancouver, Canada.
- Morscher, E. (1988). Was sind, war tun und was können Formalisierungen im Recht. Salzburg, Forschungsinstitut Philosophie/Technik/Wirtschaft.

- North, D. C. (1997). Institutions, Institutional Change and Economic Performance. Cambridge, Cambridge University Press.
- Peyton Jones, S., J. Hughes, et al. (1999). Haskell 98: A Non-strict, Purely Functional Language.
- Reinhardt, F. and H. Soeder (1991). dtv-Atlas zur Mathematik: Grundlagen, Algebra und Geometrie (Band 1). Muenchen, dtv.
- Sergot, M. J., F. Sadri, et al. (1986). "The British Nationality Act as a Logic Program." Communications of the ACM **29**(5): 370-386.
- Stamper, R. K. (1991). "The Role of Semantics in Legal Expert Systems and Legal Reasoning." Ratio Juris **4**(2): 219-244.
- Thompson, S. (1996). Haskell - The Craft of Functional Programming. Harlow, UK, Addison-Wesley.
- Valente, A. (1995). Legal Knowledge Engineering; A Modelling Approach. Amsterdam, Netherlands, IOS Press.
- van Kralingen, R. W. (1995). Frame-Based conceptual Models of Statute Law. The Hague, Netherlands, Kluwer International.
- Visser, P. R. S. (1995). Knowledge Specification for Multiple Legal Tasks; A Case Study for the Interaction Problem in the Legal Domain. The Hague, Netherlands, Kluwer International, Law.
- Winter, S. and S. Nittel (to appear). Formal Information Modelling for Standardization in the Spatial Domain, OpenGIS Consortium.

Anhang A: Suchergebnis für 'Grundstück' und 'betreten'

Zwischenstaatliche Abkommen:

Amtssitz - Org. der Vereinten Nationen für Industrielle Entwicklung
Amtssitz - Vereinten Nationen in Wien
Amtssitz - Vorbereitende Kommission für CTBTO
Europäische Menschenrechtskonvention

Bundesgesetze:

Abfallwirtschaftsgesetz
Allgemeines Sozialversicherungsgesetz
Berggesetz
Forstgesetz
Geltungsbereich der Konvention der Menschenrechte
Gentechnikgesetz 2x
Gewerberecht
Mineralrohstoffgesetz 2x
Sicherheitspolizeigesetz 2x
Sonderabfallgesetz
Strukturanpassungsgesetz
Telekommunikationsgesetz
Vermessungsgesetz 2x
Zollgesetz
Zollrechts-Durchführungsgesetz

Bundesverordnungen:

Gewerbeordnung 2x
Kulturpflanzen-Ausgleichszahlungsverordnung 1994
Kulturpflanzen-Ausgleichszahlungsverordnung 1997
Kulturpflanzenflächenzahlungsverordnung

Landesgesetze:

Baugesetz (V)
Bauordnung Wien
Bekämpfung der Ratten (W)
Burgenländisches Elektrizitätswesengesetz

Burgenländisches Jagdgesetz	
Flurverfassungs-Landesgesetz (W)	
Flurverfassungs-Landesgesetz (K)	
Geschützter Landschaftsteil Wienerberg (W)	
Güter- und Seilwege-Landesgesetz (T)	
Kanalgesetz (St)	
Kärnter Abfallwirtschaftsordnung	2x
Naturschutzgebiet Edlbacher Moor in Edlbach (O)	
Naturschutzgebiet Fuschler-Ache in St.Lorenz und Mondsee (O)	
Naturschutzgebiet Kuhschellenrasen in Gunskirchen (O)	
Naturschutzgebiet Ortler Bucht in Gmunden (O)	
O.ö. Abfallwirtschaftsgesetz 1990	2x
O.ö. Abfallwirtschaftsgesetz 1997	2x
O.ö. Elektrizitätsgesetz	
O.ö. Nationalparkgesetz	
O.ö. Natur- und Landschaftsschutzgesetz	
O.ö. Statistikgesetz	
O.ö. Umweltschutzgesetz	
Salzburger Flurverfassungs-Landesgesetz	2x
Steiermärkisches Abfallwirtschaftsgesetz	
Steiermärkisches Baugesetz	
Steiermärkisches Baumschutzgesetz	
Steiermärkisches Elektrizitätswirtschaftsgesetz	
Steiermärkisches Fischereigesetz	
Steiermärkisches Güter- und Seilwege-Landesgesetz	
Steiermärkisches Jagdgesetz	
Steiermärkisches Landesabgabenordnung	
Steiermärkisches Naturschutzgesetz	
Steiermärkisches Raumordnungsgesetz	
Steiermärkisches Starkstromwegegesetz	
Steiermärkisches Zusammenlegungsgesetz	
Tiroler Bauordnung	
Tiroler Flurverfassungslandesgesetz	
Tiroler Jagdgesetz	

Tiroler Verkehrsaufschließungsabgabengesetz
Ursprunger Moor-Naturschutzgebietsverordnung
Wiener Abfallwirtschaftsgesetz
Wiener Abgabenordnung
Wiener Baumschutzgesetz
Wiener Elektrizitätswirtschaftsgesetz
Wiener Jagdgesetz
Wiener Marktordnung
Wiener Nationalparkgesetz
Wiener Naturschutzgesetz
Wiener Pflanzenschutzmittelgesetz
Wiener Starkstromweegegesetz
Wiener Statistikgesetz

Anhang B: Haskell

1 Was ist eine funktionale Programmiersprache?

Eine funktionale Programmiersprache ist eine Programmiersprache, in der es nur Funktionen gibt. Andere Programmiersprachen haben Konstante, Prozeduren und Funktionen. Funktionale Programmiersprachen ist alles eine Funktion. Eine Funktion ist ein Ausdruck, der ausgewertet wird und nicht eine Abfolge von Befehlen, die ausgeführt wird. Diese Vereinfachung führt zu einem sehr einfachen mathematischen Konzept für funktionale Programmiersprachen.

Beim Ablauf von Programmen werden nur Ausdrücke ausgewertet. Bei prozeduralen Programmiersprachen gibt es außer der Auswertung von Ausdrücken auch die Abarbeitung von imperativen Befehlen. Eine Funktionale Programmiersprache kann durch Ersetzen von Teilausdrücken den gegebenen Ausdruck so lange vereinfachen bis keine weitere Vereinfachung mehr möglich ist.

Funktionale Programmiersprachen verwenden das Konzept der Substitution. Jeder Ausdruck hat einen bestimmten Wert, der sich während der Ausführung des Programms nicht ändert. Es gibt keine ‚side effects‘. Es gibt nur die Auswertung von Funktionen, nicht aber die Änderung von Variableninhalten.

2 Der Interpreter

Die Hauptcharakteristik einer funktionalen Programmiersprache ist, dass alles als Funktion angesehen wird. Daher ist auch die Ausführung eines Programms (das ja somit auch nur eine Funktion ist) ein Funktionsaufruf. Daher ist es einfach, Interpreter für funktionale Programmiersprachen zu schreiben. Für Haskell gibt es mit HUGS, eine komfortable Schnittstelle zu einem Haskell Interpreter.

HUGS kann unter <http://haskell.org/hugs> gefunden werden. HUGS ist für Windows und Linux/Unix erhältlich und läuft auf den meisten Hardware-Plattformen.

2.1 Vordefinierte Datentypen

Haskell bietet einige vordefinierte Datentypen:

- *Bool*: Boolesche Ausdrücke, also True und False
- *Int*: Ganze Zahlen im Bereich von -2147483647 bis 2147483648

- *Integer*: Ganze Zahlen (Bereichsbeschränkung nur durch den verfügbaren Speicher)
- *Float*: Fließkommazahlen
- *Char*: Einzelnes Zeichen

Für die vordefinierten Datentypen sind auch die Standardfunktionen definiert. Die Standardfunktionen umfassen arithmetische Operationen (+, -, *, /), logische Vergleiche (==, <, <=, etc.), trigonometrische Funktionen, usw.

2.2 Ausdrücke

Funktionen und die notwendigen Argumente bilden zusammen Ausdrücke. Standardmässig werden die Parameter in Präfix-Notation geschrieben. Dabei werden die Parameter nach dem Funktionsnamen geschrieben. Bei den mathematischen Funktionen wird aber die Infix-Notation verwendet, bei der der Funktionsname zwischen den ersten beiden Parametern steht:

```
5 + 1
3.1265 / 0.45
3 == (7 - 4)
cos (pi / 4)
```

Zusätzlich gibt es eine Verzweigung, die von einer Bedingung abhängt:

```
if BEDINGUNG then ZWEIG_1 else ZWEIG_2
```

Die Parameter der Funktionen werden nicht durch Klammern an den Funktionsnamen gebunden oder durch Kommata voneinander getrennt. Klammern dienen nur dazu, die Reihenfolge der Berechnung zu bestimmen. Eine Funktion f mit den drei Parametern a , b und c hätte in der Mathematik folgenden Aufruf: $f(a,b,c)$. In Haskell ist der Aufruf `f a b c`. Die Anzahl der Parameter, die für die Funktion notwendig sind, wird dabei von der Signatur der Funktion (siehe Abschnitt 3) bestimmt.

2.3 Vordefinierte Operationen

Es gibt eine Reihe von vordefinierten Operationen, wie beispielsweise `if – then – else`. Alle vorhandenen Operationen können im Haskell-Prelude (siehe Appendix C) gefunden werden. Den Operationen zugeordnet sind unterschiedliche Gewichtungen für die Ausführung. Die Operationen mit dem höchsten Gewicht werden zuerst ausgeführt, die mit dem geringsten

Gewicht am Schluss. Dadurch ergibt sich eine Reihenfolge der Ausführung. Will man die durch die Gewichtung gegebene Reihenfolge ändern kann man Klammersetzung verwenden.

2.4 Lazy evaluation

Die Auswertung von Ausdrücken erfolgt ‚lazy‘. Ein Ausdruck wird nur so weit ausgewertet, bis das Ergebnis feststeht. Bei einer AND-Verknüpfung von Booleschen Ausdrücken ist es beispielsweise nicht relevant, welchen Wert der zweite Ausdruck annimmt, wenn der erste bereits `False` ist. Somit ist folgender Code in Haskell ausführbar, obwohl bei der Auswertung des zweiten Ausdruckes eine Division durch Null erfolgen würde:

```
False and (1 == (5 / 0))
```

Ein weiterer Vorteil dieser Vorgangsweise ist, dass unendlich lange Datenstrukturen verwendet werden können. Man kann beispielsweise mit einer Liste der natürlichen Zahlen arbeiten. Die Liste wird dabei immer so lange berechnet, wie sie gerade notwendig ist. Es ist beispielsweise problemlos möglich, die ersten 100 Zahlen zu addieren. Man sollte allerdings vermeiden, die gesamte Liste bearbeiten zu wollen (z.B. ausdrucken), da dem System irgendwann der Speicher ausgehen wird.

3 Funktionsdefinition

Der Interpreter kann nur Ausdrücke auswerten, die in einer Zeile geschrieben werden können. Das schränkt die Definitionsmöglichkeiten für den Interpreter ein. Komplexere Funktionen müssen in einem separaten Textfile definiert und in den Interpreter geladen werden. Die wichtigsten Basisfunktionen sind bereits in einer solchen Datei, dem ‚Prelude‘ definiert. Diese Datei wird automatisch beim Start von HUGS geladen und die Befehle stehen somit sofort zur Verfügung.

Die Definition von Funktionen besteht aus zwei Teilen. Zunächst müssen die Datentypen der Argumente und des Ergebnisses bestimmt und dann die Vorgangsweise bei der Auswertung definiert werden.

3.1 Funktionssignatur

Der erste Teil heißt Signatur der Funktion. Sie setzt sich zusammen aus Name der Funktion, Trennzeichen, Datentypen der Argumente getrennt durch ‚->‘ und Datentyp des Ergebnisses. Hier einige Beispiele für einfache Signaturen:

```

increment :: Int -> Int

sin :: Float -> Float

maximum :: Int -> Int -> Int

```

Funktionsnamen beginnen immer mit Kleinbuchstaben. Danach ist jede Kombination von Buchstaben, Ziffern ‚_‘ und ‚‘ erlaubt. Dabei wird zwischen Klein- und Großschreibung unterschieden.

Es gibt auch die Möglichkeit, dass eine Funktion als Argument für eine andere Funktion verwendet wird. In dem Fall wird die Signatur der übergebenen Funktion in Klammern gesetzt und als Argument in die Liste eingefügt. Das angeführte Beispiel ist die Funktion `map`, die eine Operation auf jedes Element einer Liste anwendet. Es werden keine expliziten Datentypen angegeben weil die Funktion mit jedem beliebigen Datentyp funktioniert. Wichtig ist nur, dass der Typ der Listenelemente mit dem Typ des Arguments der übergebenen Liste übereinstimmt.

```
map :: (a -> b) -> [a] -> [b]
```

3.2 Berechnungsanweisung

Der zweite Teil der Funktionsdefinition enthält die Anweisungen zur Berechnung des Ergebnisses. Eine solche Definition besteht aus dem Namen der Funktion gefolgt von Platzhaltern für die Argumente, einem Gleichheitszeichen und einer Formel zur Berechnung.

```
increment x = x + 1
```

Innerhalb einer Berechnungsanweisung können lokale Funktionen definiert werden indem zunächst das Schlüsselwort `where` geschrieben wird und anschließend die Definitionen stehen. Wichtig ist dabei die Einrückung. Die Definition einer Funktion endet dann, wenn eine Zeile im Text wieder in derselben Spalte beginnt, in der der Name der Funktion geschrieben wurde. Eine korrekte Version einer Funktionsdefinition mit lokaler Funktion wäre daher:

```

distanz x y = sqrt (xx + yy)
  where xx = x * x
        yy = y * y

```

Folgende Versionen sind nicht richtig:

```

distanz x y = sqrt (xx + yy)
where xx = x * x
      yy = y * y

```

```
distanz x y = sqrt (xx + yy)
  where
xx = x * x
yy = y * y
```

Es gibt immer mehrere Schreibweisen für eine Funktion. Auch die folgenden Schreibweisen sind richtig:

```
distanz x y = sqrt (x * x + y * y)
distanz x y = sqrt (sqr x + sqr y)
  where
    sqr z = z * z
```

4 Kommentare

Haskell kennt zwei Arten von Kommentaren. Entweder geht ein Kommentar bis zum Ende einer Zeile oder er steht zwischen zwei Symbolen:

- `--` startet einen Kommentar, der am Ende einer Zeile aufhört. Alles was rechts davon steht gehört zum Kommentar
- `{- und -}` begrenzen einen Kommentar, der beliebig lang sein kann. Diese Art Kommentar kann in eine Befehlszeile eingefügt sein aber auch über mehrere Zeilen gehen.

5 Funktionskomposition

So wie in der Mathematik können auch in Haskell Funktionen miteinander verknüpft werden. In der Mathematik schreibt man die Verknüpfung der beiden Funktionen f und g als $f(g(x))$ oder $f \circ g(x)$. In Haskell gibt es die Operation `, . '`, die dasselbe bewirkt:

```
f :: b -> c
g :: a -> b
verkn = f.g
```

Der Typ der Funktion `verkn` ist dabei `a -> c`.

6 Definition von Datentypen – Summen-/Produkttypen, Listen, parametrisierte Datentypen

Komplexere Datentypen werden mit Hilfe von einfacheren Datentypen erzeugt. Die Namen für Datentypen folgen dabei denselben Konventionen wie die Namen für Funktionen, wobei allerdings der erste Buchstabe ein Großbuchstabe sein muss. Die Definition besteht aus vier Teilen. Zunächst steht das Schlüsselwort `data` gefolgt vom Namen, dann kommt ein `=` und den Abschluss bildet die eigentliche Definition. Für die Definition gibt es unterschiedliche Möglichkeiten.

6.1 Neue Datentypen

Haskell ermöglicht die Definition neuer Datentypen. Beispielsweise kann man Datentypen für die Wochentage definieren:

```
data Tag1 = Montag
data Tag2 = Dienstag
...
```

Dabei sind `Montag` und `Dienstag` ‚Constructor‘-Funktionen. Diese Funktionen bauen den Ergebnistyp auf.

6.2 Produkttyp

Wenn eine ‚Constructor‘-Funktion Argumente besitzt so ist das Ergebnis ein Produkttyp. Die Argumente müssen dabei einen bereits existierenden Typ haben. Beispiele für Produkttypen sind:

```
data Vektor2D = V2D Float Float
data Vektor3D = V2D Float Float Float
data Datum = Datum Int Int Int
```

6.3 Summentyp

Ein Summentyp ist ein Datentyp, der einen von mehreren Typen annehmen kann. Die einzelnen Typen werden dabei durch vertikale Striche voneinander getrennt:

```
data Wochentage = Montag | Dienstag | Mittwoch | ...
data RTyp = Eigentum Anteil RONr |
```

```
Pfandrecht Betrag Betrag RONr |
Simultanhypothek Betrag Betrag [Nr] RONr | ...
```

6.4 Liste

Listen sind Datentypen die rekursiv aufgebaut sind. Der Startpunkt ist eine leere Liste und eine Liste baut sich dann durch wiederholtes hinzufügen eines Elementes am Anfang der Liste auf. Die Funktion, die dafür zur Verfügung steht ist `,:'`. Listen von Elementen werden in Haskell durch eckige Klammern gekennzeichnet und die Elemente werden durch Kommas voneinander getrennt. Beispiele für Listen sind:

```
[1,2,3] :: [Int]
[1.4,1.5] :: [Float]
[True], [True, True, True] :: [Bool]
```

Ein Sonderfall ist dabei die Liste von Einzelzeichen (`Char`). Eine solche Liste wird als Zeichenkette (`String`) bezeichnet und durch `"` begrenzt.

```
"Navratil" :: [Char]
```

Um eine Liste zu erweitern steht der Konstruktor `,:'` zur Verfügung:

```
'N' : "avratil"           ergibt  "Navratil"
1 : [2,3,4]              ergibt  [1,2,3,4]
True : [False, True]     ergibt  [True, False, True]
```

6.5 Parametrisierte Typen

Datentypen können mit Typparametern parametrisiert werden. Diese Parameter müssen bei der Verwendung des Datentyps für einen bestimmten Datentyp instanziiert werden. Der folgende Datentyp beispielsweise kann als Basis für 2D-Vektoroperationen dienen, wobei für die Koordinaten später unterschiedliche Datentypen verwendet werden können:

```
data Vector2D coord = Vector2D coord coord
```

7 Klassendefinition

Die einfachste Art der Klassendefinitionen sieht folgendermaßen aus. Die Definition wird durch das Schlüsselwort `class` eingeleitet. Anschließend folgt der Name der Klasse, der mit einem Großbuchstaben beginnen muss. Danach kommt eine Typvariable und das Schlüsselwort `where`. Anschließend folgen die Signaturen der Funktionen der Klasse. Wichtig ist dabei, dass

die Typvariable bei jeder Funktion verwendet werden muss. Das Ende der Definition ist dadurch gekennzeichnet, dass eine Zeile wieder bei derselben Spalte beginnt wie die Zeile mit dem Schlüsselwort `class`.

```
class Testklasse a where
  op1 :: a -> a
  op2 :: Int -> a
  ...
```

Klassen können auch von mehreren Typvariablen abhängig sein. Diese werden dann zwischen dem Namen der Klasse und `where` aufgelistet, wobei die Trennung durch Leerzeichen erfolgt. Zusätzlich können Klassen andere Klassen voraussetzen. Die vorausgesetzten Klassen werden direkt nach dem Klassennamen geschrieben und von den Typvariablen der Klasse durch `=>` getrennt. Schließlich können die Funktionen der Klasse auch durch Axiome definiert werden.

```
class (Dokumente d r, Hauptbuecher Hauptbuch r) =>
  Grundbuecher g d r where
  grundbuecher      :: Hauptbuch r -> [d r] -> g d r
  hauptbuch        :: g d r -> Hauptbuch r
  urkundensammlung :: g d r -> [d r]

  rechtfertigen :: g d r -> Nr -> d r -> Datum -> g d r
  rechtfertigen g ez d dt =
    grundbuecher (rechtfHB (hauptbuch g) ez (recht d) dt)
      (d:urkundensammlung g)
```

Klassen müssen für die Verwendung mit einem bestimmten Datentyp instanziiert werden. Dazu dient das Schlüsselwort `instance`. Nach dem Schlüsselwort steht der Name der Klasse, die verwendeten Datentypen, das Schlüsselwort `where` und anschließend alle zu definierenden Funktionen.

```
instance Grundbuecher Grundbuch Dokument Recht where
  grundbuecher h u      = G h u
  hauptbuch      (G h u) = h
  urkundensammlung (G h u) = u
```

Anhang C: Haskell Prelude

```
{-----  
-----  
||  || ||  ||  ||  ||  ||__  Hugs 98: The Nottingham and Yale Haskell system  
||__|| ||__|| ||__||  __||  Copyright (c) 1994-1999  
||---||          __||          World Wide Web: http://haskell.org/hugs  
||  ||          Report bugs to: hugs-bugs@haskell.org  
||  || Version: February 1999-----  
-----
```

This is the Hugs 98 Standard Prelude, based very closely on the Standard Prelude for Haskell 98.

WARNING: This file is an integral part of the Hugs source code. Changes to the definitions in this file without corresponding modifications in other parts of the program may cause the interpreter to fail unexpectedly. Under normal circumstances, you should not attempt to modify this file in any way!

```
-----  
The Hugs 98 system is Copyright (c) Mark P Jones, Alastair Reid, the  
Yale Haskell Group, and the Oregon Graduate Institute of Science and  
Technology, 1994-1999, All rights reserved. It is distributed as  
free software under the license in the file "License", which is  
included in the distribution.  
-----}
```

```
module Prelude (  
-- module PreludeList,  
    map, (++), concat, filter,  
    head, last, tail, init, null, length, (!!),  
    foldl, foldl1, scanl, scanl1, foldr, foldr1, scanr, scanr1,  
    iterate, repeat, replicate, cycle,  
    take, drop, splitAt, takeWhile, dropWhile, span, break,  
    lines, words, unlines, unwords, reverse, and, or,  
    any, all, elem, notElem, lookup,  
    sum, product, maximum, minimum, concatMap,  
    zip, zip3, zipWith, zipWith3, unzip, unzip3,  
-- module PreludeText,  
    ReadS, ShowS,  
    Read(readsPrec, readList),  
    Show(show, showsPrec, showList),  
    reads, shows, read, lex,
```

```

    showChar, showString, readParen, showParen,
-- module PreludeIO,
    FilePath, IOError, ioError, userError, catch,
    putChar, putStr, putStrLn, print,
    getChar, getLine, getContents, interact,
    readFile, writeFile, appendFile, readIO, readLn,
-- module Ix,
    Ix(range, index, inRange, rangeSize),
-- module Char,
    isAscii, isControl, isPrint, isSpace, isUpper, isLower,
    isAlpha, isDigit, isOctDigit, isHexDigit, isAlphaNum,
    digitToInt, intToDigit,
    toUpper, toLower,
    ord, chr,
    readLitChar, showLitChar, lexLitChar,
-- module Numeric
    showSigned, showInt,
    readSigned, readInt,
    readDec, readOct, readHex, readSigned,
    readFloat, lexDigits,
-- module Ratio,
    Ratio, Rational, (%), numerator, denominator, approxRational,
-- Non-standard exports
    IO(..), IOResult(..), primExitWith, Addr, Word, StablePtr, ForeignObj,
    basicIORun, IOFinished(..),
    threadToIOResult,

    Bool(False, True),
    Maybe(Nothing, Just),
    Either(Left, Right),
    Ordering(LT, EQ, GT),
    Char, String, Int, Integer, Float, Double, IO,
-- List type: []((:), [])
    (:),
-- Tuple types: (,), (,,), etc.
-- Trivial type: ()
-- Functions: (->)
    Rec, EmptyRec, EmptyRow, -- non-standard, should only be exported if TREX
    Eq(==, (/=)),
    Ord(compare, (<), (<=), (>=), (>), max, min),
    Enum(succ, pred, toEnum, fromEnum, enumFrom, enumFromThen,
        enumFromTo, enumFromThenTo),
    Bounded(minBound, maxBound),

```

```

-- Num(+), (-), (*), negate, abs, signum, fromInteger),
   Num(+), (-), (*), negate, abs, signum, fromInteger, fromInt),
   Real(toRational),
-- Integral(quot, rem, div, mod, quotRem, divMod, toInteger),
   Integral(quot, rem, div, mod, quotRem, divMod, even, odd, toInteger, toInt),
-- Fractional(/), recip, fromRational),
   Fractional(/), recip, fromRational, fromDouble),
   Floating(pi, exp, log, sqrt, (**), logBase, sin, cos, tan,
             asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh),
   RealFrac(properFraction, truncate, round, ceiling, floor),
   RealFloat(floatRadix, floatDigits, floatRange, decodeFloat,
             encodeFloat, exponent, significand, scaleFloat, isNaN,
             isInfinite, isDenormalized, isIEEE, isNegativeZero, atan2),
   Monad(>>=), (>>), return, fail),
   Functor(fmap),
   mapM, mapM_, sequence, sequence_, (= <<),
   maybe, either,
   (&&), (||), not, otherwise,
   subtract, even, odd, gcd, lcm, (^), (**),
   fromIntegral, realToFrac,
   fst, snd, curry, uncurry, id, const, (.), flip, ($), until,
   asTypeOf, error, undefined,
   seq, ($)
) where

-- Standard value bindings {Prelude} -----

infixr 9  .
infixl 9  !!
infixr 8  ^, ^^, **
infixl 7  *, /, `quot`, `rem`, `div`, `mod`, :%, %
infixl 6  +, -
--infixr 5  :    -- this fixity declaration is hard-wired into Hugs
infixr 5  ++
infix  4  ==, /=, <, <=, >=, >, `elem`, `notElem`
infixr 3  &&
infixr 2  ||
infixl 1  >>, >>=
infixr 1  = <<
infixr 0  $, $!, `seq`

```

```
-- Equality and Ordered classes -----
```

```
class Eq a where
```

```
  (==), (/=) :: a -> a -> Bool
```

```
  -- Minimal complete definition: (==) or (/=)
```

```
  x == y      = not (x/=y)
```

```
  x /= y      = not (x==y)
```

```
class (Eq a) => Ord a where
```

```
  compare      :: a -> a -> Ordering
```

```
  (<), (<=), (>=), (>) :: a -> a -> Bool
```

```
  max, min     :: a -> a -> a
```

```
  -- Minimal complete definition: (<=) or compare
```

```
  -- using compare can be more efficient for complex types
```

```
  compare x y | x==y      = EQ
```

```
              | x<=y      = LT
```

```
              | otherwise = GT
```

```
  x <= y      = compare x y /= GT
```

```
  x < y       = compare x y == LT
```

```
  x >= y      = compare x y /= LT
```

```
  x > y       = compare x y == GT
```

```
  max x y    | x >= y      = x
```

```
              | otherwise = y
```

```
  min x y    | x <= y      = x
```

```
              | otherwise = y
```

```
class Bounded a where
```

```
  minBound, maxBound :: a
```

```
  -- Minimal complete definition: All
```

```
-- Numeric classes -----
```

```
class (Eq a, Show a) => Num a where
```

```
  (+), (-), (*) :: a -> a -> a
```

```
  negate      :: a -> a
```

```
  abs, signum :: a -> a
```

```
  fromInteger :: Integer -> a
```

```
  fromInt     :: Int -> a
```

```

-- Minimal complete definition: All, except negate or (-)
x - y          = x + negate y
fromInt        = fromIntegral
negate x       = 0 - x

class (Num a, Ord a) => Real a where
  toRational    :: a -> Rational

class (Real a, Enum a) => Integral a where
  quot, rem, div, mod :: a -> a -> a
  quotRem, divMod    :: a -> a -> (a,a)
  even, odd          :: a -> Bool
  toInteger         :: a -> Integer
  toInt             :: a -> Int

-- Minimal complete definition: quotRem and toInteger
n `quot` d      = q where (q,r) = quotRem n d
n `rem` d       = r where (q,r) = quotRem n d
n `div` d       = q where (q,r) = divMod n d
n `mod` d       = r where (q,r) = divMod n d
divMod n d      = if signum r == - signum d then (q-1, r+d) else qr
                  where qr@(q,r) = quotRem n d

even n          = n `rem` 2 == 0
odd             = not . even
toInt          = toInt . toInteger

class (Num a) => Fractional a where
  (/)           :: a -> a -> a
  recip         :: a -> a
  fromRational :: Rational -> a
  fromDouble   :: Double -> a

-- Minimal complete definition: fromRational and (/) or recip
recip x        = 1 / x
fromDouble     = fromRational . toRational
x / y         = x * recip y

```

```

class (Fractional a) => Floating a where
  pi                :: a
  exp, log, sqrt    :: a -> a
  (**), logBase     :: a -> a -> a
  sin, cos, tan     :: a -> a
  asin, acos, atan  :: a -> a
  sinh, cosh, tanh  :: a -> a
  asinh, acosh, atanh :: a -> a

-- Minimal complete definition: pi, exp, log, sin, cos, sinh, cosh,
--                               asinh, acosh, atanh
pi                = 4 * atan 1
x ** y            = exp (log x * y)
logBase x y       = log y / log x
sqrt x            = x ** 0.5
tan x             = sin x / cos x
sinh x            = (exp x - exp (-x)) / 2
cosh x            = (exp x + exp (-x)) / 2
tanh x            = sinh x / cosh x
asinh x           = log (x + sqrt (x*x + 1))
acosh x           = log (x + sqrt (x*x - 1))
atanh x           = (log (1 + x) - log (1 - x)) / 2

class (Real a, Fractional a) => RealFrac a where
  properFraction   :: (Integral b) => a -> (b,a)
  truncate, round  :: (Integral b) => a -> b
  ceiling, floor   :: (Integral b) => a -> b

-- Minimal complete definition: properFraction
truncate x        = m where (m,_) = properFraction x

round x           = let (n,r) = properFraction x
                    m       = if r < 0 then n - 1 else n + 1
                    in case signum (abs r - 0.5) of
                        -1 -> n
                        0  -> if even n then n else m
                        1  -> m

ceiling x         = if r > 0 then n + 1 else n
                  where (n,r) = properFraction x

floor x           = if r < 0 then n - 1 else n
                  where (n,r) = properFraction x

```

```

class (RealFrac a, Floating a) => RealFloat a where
  floatRadix      :: a -> Integer
  floatDigits     :: a -> Int
  floatRange      :: a -> (Int,Int)
  decodeFloat     :: a -> (Integer,Int)
  encodeFloat     :: Integer -> Int -> a
  exponent        :: a -> Int
  significand     :: a -> a
  scaleFloat      :: Int -> a -> a
  isNaN, isInfinite, isDenormalized, isNegativeZero, isIEEE
                :: a -> Bool
  atan2          :: a -> a -> a

-- Minimal complete definition: All, except exponent, significand,
--                               scaleFloat, atan2
exponent x      = if m==0 then 0 else n + floatDigits x
                  where (m,n) = decodeFloat x
significand x   = encodeFloat m (- floatDigits x)
                  where (m,_) = decodeFloat x
scaleFloat k x  = encodeFloat m (n+k)
                  where (m,n) = decodeFloat x

atan2 y x
  | x>0          = atan (y/x)
  | x==0 && y>0  = pi/2
  | x<0 && y>0   = pi + atan (y/x)
  | (x<=0 && y<0) ||
    (x<0 && isNegativeZero y) ||
    (isNegativeZero x && isNegativeZero y)
                = - atan2 (-y) x
  | y==0 && (x<0 || isNegativeZero x)
                = pi      -- must be after the previous test on zero y
  | x==0 && y==0 = y      -- must be after the other double zero tests
  | otherwise    = x + y -- x or y is a NaN, return a NaN (via +)

-- Numeric functions -----
subtract      :: Num a => a -> a -> a
subtract     = flip (-)

gcd           :: Integral a => a -> a -> a
gcd 0 0      = error "Prelude.gcd: gcd 0 0 is undefined"

```

```

gcd x y      = gcd' (abs x) (abs y)
              where gcd' x 0 = x
                  gcd' x y = gcd' y (x `rem` y)

lcm          :: (Integral a) => a -> a -> a
lcm _ 0      = 0
lcm 0 _      = 0
lcm x y      = abs ((x `quot` gcd x y) * y)

(^)          :: (Num a, Integral b) => a -> b -> a
x ^ 0        = 1
x ^ n | n > 0 = f x (n-1) x
              where f _ 0 y = y
                  f x n y = g x n where
                      g x n | even n    = g (x*x) (n`quot`2)
                          | otherwise = f x (n-1) (x*y)
_ ^ _        = error "Prelude.^: negative exponent"

(^^)         :: (Fractional a, Integral b) => a -> b -> a
x ^^ n       = if n >= 0 then x ^ n else recip (x^(-n))

fromIntegral :: (Integral a, Num b) => a -> b
fromIntegral = fromInteger . toInteger

realToFrac   :: (Real a, Fractional b) => a -> b
realToFrac   = fromRational . toRational

-- Index and Enumeration classes -----

class (Ord a) => Ix a where
    range          :: (a,a) -> [a]
    index          :: (a,a) -> a -> Int
    inRange        :: (a,a) -> a -> Bool
    rangeSize      :: (a,a) -> Int

    rangeSize r@(l,u)
        | l > u      = 0
        | otherwise = index r u + 1

```

```

class Enum a where
  succ, pred      :: a -> a
  toEnum         :: Int -> a
  fromEnum       :: a -> Int
  enumFrom       :: a -> [a]           -- [n..]
  enumFromThen   :: a -> a -> [a]     -- [n,m..]
  enumFromTo     :: a -> a -> [a]     -- [n..m]
  enumFromThenTo :: a -> a -> a -> [a] -- [n,n'..m]

  -- Minimal complete definition: toEnum, fromEnum
  succ      = toEnum . (1+)      . fromEnum
  pred     = toEnum . subtract 1 . fromEnum
  enumFrom x      = map toEnum [ fromEnum x .. ]
  enumFromThen x y = map toEnum [ fromEnum x, fromEnum y .. ]
  enumFromTo x y  = map toEnum [ fromEnum x .. fromEnum y ]
  enumFromThenTo x y z = map toEnum [ fromEnum x, fromEnum y .. fromEnum z ]

-- Read and Show classes -----

type ReadS a = String -> [(a,String)]
type ShowS   = String -> String

class Read a where
  readsPrec :: Int -> ReadS a
  readList  :: ReadS [a]

  -- Minimal complete definition: readsPrec
  readList = readParen False (\r -> [pr | ("",s) <- lex r,
                                           pr      <- readl s ])
    where readl s = [([],t) | ("",t) <- lex s] ++
      [(x:xs,u) | (x,t) <- reads s,
                  (xs,u) <- readl' t]
      readl' s = [([],t) | ("",t) <- lex s] ++
        [(x:xs,v) | ("",t) <- lex s,
                    (x,u) <- reads t,
                    (xs,v) <- readl' u]

class Show a where
  show      :: a -> String
  showsPrec :: Int -> a -> ShowS
  showList  :: [a] -> ShowS

  -- Minimal complete definition: show or showsPrec

```

```

show x          = showsPrec 0 x ""
showsPrec _ x s = show x ++ s
showList []     = showString "[]"
showList (x:xs) = showChar '[' . shows x . showl xs
                  where showl []     = showChar ']'
                        showl (x:xs) = showChar ',' . shows x . showl xs

-- Monad classes -----

class Functor f where
  fmap :: (a -> b) -> (f a -> f b)

class Monad m where
  return :: a -> m a
  (>>=)  :: m a -> (a -> m b) -> m b
  (>>)   :: m a -> m b -> m b
  fail   :: String -> m a

  -- Minimal complete definition: (>>=), return
  p >> q = p >>= \ _ -> q
  fail s = error s

sequence      :: Monad m => [m a] -> m [a]
sequence []   = return []
sequence (c:cs) = do x <- c
                    xs <- sequence cs
                    return (x:xs)

sequence_     :: Monad m => [m a] -> m ()
sequence_     = foldr (>>) (return ())

mapM          :: Monad m => (a -> m b) -> [a] -> m [b]
mapM f        = sequence . map f

mapM_         :: Monad m => (a -> m b) -> [a] -> m ()
mapM_ f       = sequence_ . map f

(=<<)         :: Monad m => (a -> m b) -> m a -> m b
f =<< x       = x >>= f

```

```

-- Evaluation and strictness -----

primitive seq          :: a -> b -> b

primitive ($) "strict" :: (a -> b) -> a -> b
-- f $! x              = x `seq` f x

-- Trivial type -----

-- data () = () deriving (Eq, Ord, Ix, Enum, Read, Show, Bounded)

instance Eq () where
    () == () = True

instance Ord () where
    compare () () = EQ

instance Ix () where
    range ((),()) = [()]
    index ((),()) () = 0
    inRange ((),()) () = True

instance Enum () where
    toEnum 0 = ()
    fromEnum () = 0
    enumFrom () = [()]
    enumFromThen () () = [()]

instance Read () where
    readsPrec p = readParen False (\r -> [(() ,t) | ("(",s) <- lex r,
                                                    (")",t) <- lex s ])

instance Show () where
    showsPrec p () = showString "()"

instance Bounded () where
    minBound = ()
    maxBound = ()

-- Boolean type -----

data Bool = False | True
           deriving (Eq, Ord, Ix, Enum, Read, Show, Bounded)

```

```

(&&), (||)  :: Bool -> Bool -> Bool
False && x   = False
True  && x   = x
False || x  = x
True  || x  = True

not         :: Bool -> Bool
not True   = False
not False  = True

otherwise   :: Bool
otherwise   = True

-- Character type -----

data Char          -- builtin datatype of ISO Latin characters
type String = [Char] -- strings are lists of characters

primitive primEqChar  :: Char -> Char -> Bool
primitive primCmpChar :: Char -> Char -> Ordering

instance Eq Char where (==) = primEqChar
instance Ord Char where compare = primCmpChar

primitive primCharToInt :: Char -> Int
primitive primIntToChar :: Int -> Char

instance Enum Char where
    toEnum          = primIntToChar
    fromEnum        = primCharToInt
    enumFrom c      = map toEnum [fromEnum c .. fromEnum (maxBound::Char)]
    enumFromThen c d = map toEnum [fromEnum c, fromEnum d .. fromEnum
    (lastChar::Char)]
    where lastChar = if d < c then minBound else maxBound

instance Ix Char where
    range (c,c')      = [c..c']
    index b@(c,c') ci
        | inRange b ci = fromEnum ci - fromEnum c
        | otherwise     = error "Ix.index: Index out of range."
    inRange (c,c') ci = fromEnum c <= i && i <= fromEnum c'
    where i = fromEnum ci

```

```

instance Read Char where
  readsPrec p      = readParen False
                    (\r -> [(c,t) | ('\':s,t) <- lex r,
                                   (c,"\'") <- readLitChar s ])
  readList = readParen False (\r -> [(l,t) | ('':s, t) <- lex r,
                                       (l,_) <- readl s ])
  where readl ('':s)      = [("",s)]
        readl ('\':s) = readl s
        readl s          = [(c:cs,u) | (c ,t) <- readLitChar s,
                                       (cs,u) <- readl t ]

instance Show Char where
  showsPrec p '\'' = showString "\\\'"
  showsPrec p c    = showChar '\'' . showLitChar c . showChar '\''

  showList cs      = showChar '"' . showl cs
  where showl ""   = showChar '"'
        showl ('':cs) = showString "\\\"" . showl cs
        showl (c:cs)  = showLitChar c . showl cs

instance Bounded Char where
  minBound = '\0'
  maxBound = '\255'

isAscii, isControl, isPrint, isSpace      :: Char -> Bool
isUpper, isLower, isAlpha, isDigit, isAlphaNum :: Char -> Bool

isAscii c          = fromEnum c < 128
isControl c        = c < ' ' || c == '\DEL'
isPrint c          = c >= ' ' && c <= '~'
isSpace c          = c == ' ' || c == '\t' || c == '\n' ||
                    c == '\r' || c == '\f' || c == '\v'
isUpper c          = c >= 'A' && c <= 'Z'
isLower c          = c >= 'a' && c <= 'z'
isAlpha c          = isUpper c || isLower c
isDigit c          = c >= '0' && c <= '9'
isAlphaNum c       = isAlpha c || isDigit c

-- Digit conversion operations
digitToInt :: Char -> Int
digitToInt c
  | isDigit c      = fromEnum c - fromEnum '0'
  | c >= 'a' && c <= 'f' = fromEnum c - fromEnum 'a' + 10

```

```

| c >= 'A' && c <= 'F' = fromEnum c - fromEnum 'A' + 10
| otherwise             = error "Char.digitToInt: not a digit"

intToDigit :: Int -> Char
intToDigit i
| i >= 0 && i <= 9  = toEnum (fromEnum '0' + i)
| i >= 10 && i <= 15 = toEnum (fromEnum 'a' + i - 10)
| otherwise         = error "Char.intToDigit: not a digit"

toUpper, toLower :: Char -> Char
toUpper c | isLower c = toEnum (fromEnum c - fromEnum 'a' + fromEnum 'A')
          | otherwise  = c
toLower c | isUpper c = toEnum (fromEnum c - fromEnum 'A' + fromEnum 'a')
          | otherwise  = c

ord :: Char -> Int
ord  = fromEnum

chr :: Int -> Char
chr  = toEnum

-- Maybe type -----

data Maybe a = Nothing | Just a
              deriving (Eq, Ord, Read, Show)

maybe :: b -> (a -> b) -> Maybe a -> b
maybe n f Nothing  = n
maybe n f (Just x) = f x

instance Functor Maybe where
    fmap f Nothing  = Nothing
    fmap f (Just x) = Just (f x)

instance Monad Maybe where
    Just x >>= k = k x
    Nothing >>= k = Nothing
    return      = Just
    fail s      = Nothing

```

```

-- Either type -----
data Either a b = Left a | Right b
    deriving (Eq, Ord, Read, Show)

either          :: (a -> c) -> (b -> c) -> Either a b -> c
either l r (Left x)  = l x
either l r (Right y) = r y

-- Ordering type -----
data Ordering = LT | EQ | GT
    deriving (Eq, Ord, Ix, Enum, Read, Show, Bounded)

-- Lists -----
-- data [a] = [] | a : [a] deriving (Eq, Ord)

instance Eq a => Eq [a] where
    []      == []      = True
    (x:xs) == (y:ys) = x==y && xs==ys
    _      == _      = False

instance Ord a => Ord [a] where
    compare []      (_:_) = LT
    compare []      []    = EQ
    compare (_:_) []    = GT
    compare (x:xs) (y:ys) = primCompAux x y (compare xs ys)

instance Functor [] where
    fmap = map

instance Monad [] where
    (x:xs) >>= f = f x ++ (xs >>= f)
    []      >>= f = []
    return x      = [x]
    fail s        = []

instance Read a => Read [a] where
    readsPrec p = readList

instance Show a => Show [a] where
    showsPrec p = showList

```

```

-- Tuples -----
-- data (a,b) = (a,b) deriving (Eq, Ord, Ix, Read, Show)
-- etc..

-- Standard Integral types -----

data Int      -- builtin datatype of fixed size integers
data Integer  -- builtin datatype of arbitrary size integers

primitive primEqInt      :: Int -> Int -> Bool
primitive primCmpInt     :: Int -> Int -> Ordering
primitive primEqInteger  :: Integer -> Integer -> Bool
primitive primCmpInteger :: Integer -> Integer -> Ordering

instance Eq Int      where (==) = primEqInt
instance Eq Integer where (==) = primEqInteger
instance Ord Int     where compare = primCmpInt
instance Ord Integer where compare = primCmpInteger

primitive primPlusInt,
           primMinusInt,
           primMulInt      :: Int -> Int -> Int
primitive primNegInt      :: Int -> Int
primitive primIntegerToInt :: Integer -> Int

instance Num Int where
    (+)      = primPlusInt
    (-)      = primMinusInt
    negate   = primNegInt
    (*)      = primMulInt
    abs      = absReal
    signum   = signumReal
    fromInteger = primIntegerToInt
    fromInt x = x

primitive primMinInt, primMaxInt :: Int

instance Bounded Int where
    minBound = primMinInt
    maxBound = primMaxInt

```

```

primitive primPlusInteger,
    primMinusInteger,
    primMulInteger    :: Integer -> Integer -> Integer
primitive primNegInteger    :: Integer -> Integer
primitive primIntToInteger :: Int -> Integer

instance Num Integer where
    (+)      = primPlusInteger
    (-)      = primMinusInteger
    negate   = primNegInteger
    (*)      = primMulInteger
    abs      = absReal
    signum   = signumReal
    fromInteger x = x
    fromInt   = primIntToInteger

absReal x | x >= 0 = x
          | otherwise = -x

signumReal x | x == 0 = 0
             | x > 0 = 1
             | otherwise = -1

instance Real Int where
    toRational x = toInteger x % 1

instance Real Integer where
    toRational x = x % 1

primitive primDivInt,
    primQuotInt,
    primRemInt,
    primModInt    :: Int -> Int -> Int
primitive primQrmInt    :: Int -> Int -> (Int,Int)
primitive primEvenInt :: Int -> Bool

instance Integral Int where
    div      = primDivInt
    quot     = primQuotInt
    rem      = primRemInt
    mod      = primModInt
    quotRem  = primQrmInt
    even     = primEvenInt

```

```

toInteger = primIntToInteger
toInt x   = x

```

```

primitive primQrmInteger :: Integer -> Integer -> (Integer,Integer)
primitive primEvenInteger :: Integer -> Bool

```

```

instance Integral Integer where
    quotRem    = primQrmInteger
    even       = primEvenInteger
    toInteger x = x
    toInt      = primIntegerToInt

```

```

instance Ix Int where
    range (m,n)          = [m..n]
    index b@(m,n) i
        | inRange b i = i - m
        | otherwise   = error "index: Index out of range"
    inRange (m,n) i     = m <= i && i <= n

```

```

instance Ix Integer where
    range (m,n)          = [m..n]
    index b@(m,n) i
        | inRange b i = fromInteger (i - m)
        | otherwise   = error "index: Index out of range"
    inRange (m,n) i     = m <= i && i <= n

```

```

instance Enum Int where
    toEnum          = id
    fromEnum        = id
    enumFrom        = numericEnumFrom
    enumFromTo      = numericEnumFromTo
    enumFromThen    = numericEnumFromThen
    enumFromThenTo  = numericEnumFromThenTo

```

```

instance Enum Integer where
    toEnum          = primIntToInteger
    fromEnum        = primIntegerToInt
    enumFrom        = numericEnumFrom
    enumFromTo      = numericEnumFromTo
    enumFromThen    = numericEnumFromThen
    enumFromThenTo  = numericEnumFromThenTo

```

```

numericEnumFrom      :: Real a => a -> [a]
numericEnumFromThen  :: Real a => a -> a -> [a]
numericEnumFromTo    :: Real a => a -> a -> [a]
numericEnumFromThenTo :: Real a => a -> a -> a -> [a]
numericEnumFrom n      = n : (numericEnumFrom $! (n+1))
numericEnumFromThen n m = iterate ((m-n)+) n
numericEnumFromTo n m  = takeWhile (<= m) (numericEnumFrom n)
numericEnumFromThenTo n n' m = takeWhile p (numericEnumFromThen n n')
    where p | n' >= n    = (<= m)
          | otherwise = (>= m)

primitive primShowsInt :: Int -> Int -> ShowS

instance Read Int where
    readsPrec p = readSigned readDec

instance Show Int where
    showsPrec    = primShowsInt

primitive primShowsInteger :: Int -> Integer -> ShowS

instance Read Integer where
    readsPrec p = readSigned readDec

instance Show Integer where
    showsPrec    = primShowsInteger

-- Standard Floating types -----

data Float    -- builtin datatype of single precision floating point numbers
data Double  -- builtin datatype of double precision floating point numbers

primitive primEqFloat    :: Float -> Float -> Bool
primitive primCmpFloat  :: Float -> Float -> Ordering
primitive primEqDouble  :: Double -> Double -> Bool
primitive primCmpDouble :: Double -> Double -> Ordering

instance Eq Float where (==) = primEqFloat
instance Eq Double where (==) = primEqDouble

instance Ord Float where compare = primCmpFloat
instance Ord Double where compare = primCmpDouble

```

```

primitive primPlusFloat,
    primMinusFloat,
    primMulFloat      :: Float -> Float -> Float
primitive primNegFloat    :: Float -> Float
primitive primIntToFloat  :: Int -> Float
primitive primIntegerToFloat :: Integer -> Float

instance Num Float where
    (+)      = primPlusFloat
    (-)      = primMinusFloat
    negate   = primNegFloat
    (*)      = primMulFloat
    abs      = absReal
    signum   = signumReal
    fromInteger = primIntegerToFloat
    fromInt   = primIntToFloat

primitive primPlusDouble,
    primMinusDouble,
    primMulDouble      :: Double -> Double -> Double
primitive primNegDouble    :: Double -> Double
primitive primIntToDouble  :: Int -> Double
primitive primIntegerToDouble :: Integer -> Double

instance Num Double where
    (+)      = primPlusDouble
    (-)      = primMinusDouble
    negate   = primNegDouble
    (*)      = primMulDouble
    abs      = absReal
    signum   = signumReal
    fromInteger = primIntegerToDouble
    fromInt   = primIntToDouble

instance Real Float where
    toRational = floatToRational

instance Real Double where
    toRational = doubleToRational

-- Calls to these functions are optimised when passed as arguments to
-- fromRational.
floatToRational :: Float -> Rational

```

```

doubleToRational :: Double -> Rational
floatToRational x = realFloatToRational x
doubleToRational x = realFloatToRational x

realFloatToRational x = (m%1)*(b%1)^n
    where (m,n) = decodeFloat x
          b     = floatRadix x

primitive primDivFloat      :: Float -> Float -> Float
primitive doubleToFloat    :: Double -> Float

instance Fractional Float where
    (/)          = primDivFloat
    fromRational = primRationalToFloat
    fromDouble   = doubleToFloat

primitive primDivDouble :: Double -> Double -> Double

instance Fractional Double where
    (/)          = primDivDouble
    fromRational = primRationalToDouble
    fromDouble x = x

-- These primitives are equivalent to (and are defined using)
-- rationalTo{Float,Double}. The difference is that they test to see
-- if their argument is of the form (fromDouble x) - which allows a much
-- more efficient implementation.
primitive primRationalToFloat :: Rational -> Float
primitive primRationalToDouble :: Rational -> Double

-- These functions are used by Hugs - don't change their types.
rationalToFloat :: Rational -> Float
rationalToDouble :: Rational -> Double
rationalToFloat = rationalToRealFloat
rationalToDouble = rationalToRealFloat

```

```

rationalToRealFloat x = x'
  where x'      = f e
        f e    = if e' == e then y else f e'
                where y      = encodeFloat (round (x * (1%b)^e)) e
                          (_ ,e') = decodeFloat y
        (_ ,e) = decodeFloat (fromInteger (numerator x) `asTypeOf` x'
                              / fromInteger (denominator x))
        b      = floatRadix x'

primitive primSinFloat,  primAsinFloat, primCosFloat,
          primAcosFloat, primTanFloat,  primAtanFloat,
          primLogFloat,  primExpFloat,  primSqrtFloat :: Float -> Float

instance Floating Float where
  exp  = primExpFloat
  log  = primLogFloat
  sqrt = primSqrtFloat
  sin  = primSinFloat
  cos  = primCosFloat
  tan  = primTanFloat
  asin = primAsinFloat
  acos = primAcosFloat
  atan = primAtanFloat

primitive primSinDouble, primAsinDouble, primCosDouble,
          primAcosDouble, primTanDouble,  primAtanDouble,
          primLogDouble,  primExpDouble,  primSqrtDouble :: Double -> Double

instance Floating Double where
  exp  = primExpDouble
  log  = primLogDouble
  sqrt = primSqrtDouble
  sin  = primSinDouble
  cos  = primCosDouble
  tan  = primTanDouble
  asin = primAsinDouble
  acos = primAcosDouble
  atan = primAtanDouble

instance RealFrac Float where
  properFraction = floatProperFraction

```

```

instance RealFrac Double where
    properFraction = floatProperFraction

floatProperFraction x
| n >= 0      = (fromInteger m * fromInteger b ^ n, 0)
| otherwise   = (fromInteger w, encodeFloat r n)
    where (m,n) = decodeFloat x
          b     = floatRadix x
          (w,r) = quotRem m (b^(-n))

primitive primFloatRadix  :: Integer
primitive primFloatDigits :: Int
primitive primFloatMinExp,
    primFloatMaxExp :: Int
primitive primFloatEncode :: Integer -> Int -> Float
primitive primFloatDecode :: Float -> (Integer, Int)

instance RealFloat Float where
    floatRadix _ = primFloatRadix
    floatDigits _ = primFloatDigits
    floatRange _ = (primFloatMinExp, primFloatMaxExp)
    encodeFloat = primFloatEncode
    decodeFloat = primFloatDecode
    isNaN _ = False
    isInfinite _ = False
    isDenormalized _ = False
    isNegativeZero _ = False
    isIEEE _ = False

primitive primDoubleRadix  :: Integer
primitive primDoubleDigits :: Int
primitive primDoubleMinExp,
    primDoubleMaxExp :: Int
primitive primDoubleEncode :: Integer -> Int -> Double
primitive primDoubleDecode :: Double -> (Integer, Int)

instance RealFloat Double where
    floatRadix _ = primDoubleRadix
    floatDigits _ = primDoubleDigits
    floatRange _ = (primDoubleMinExp, primDoubleMaxExp)
    encodeFloat = primDoubleEncode
    decodeFloat = primDoubleDecode
    isNaN _ = False

```

```

    isInfinite _ = False
    isDenormalized _ = False
    isNegativeZero _ = False
    isIEEE _ = False

instance Enum Float where
    toEnum      = primIntToFloat
    fromEnum    = truncate
    enumFrom    = numericEnumFrom
    enumFromThen = numericEnumFromThen
    enumFromTo n m = numericEnumFromTo n (m+1/2)
    enumFromThenTo n n' m = numericEnumFromThenTo n n' (m + (n'-n)/2)

instance Enum Double where
    toEnum      = primIntToDouble
    fromEnum    = truncate
    enumFrom    = numericEnumFrom
    enumFromThen = numericEnumFromThen
    enumFromTo n m = numericEnumFromTo n (m+1/2)
    enumFromThenTo n n' m = numericEnumFromThenTo n n' (m + (n'-n)/2)

primitive primShowsFloat :: Int -> Float -> ShowS

instance Read Float where
    readsPrec p = readSigned readFloat

-- Note that showFloat in Numeric isn't used here
instance Show Float where
    showsPrec = primShowsFloat

primitive primShowsDouble :: Int -> Double -> ShowS

instance Read Double where
    readsPrec p = readSigned readFloat

-- Note that showFloat in Numeric isn't used here
instance Show Double where
    showsPrec = primShowsDouble

-- Some standard functions -----

fst      :: (a,b) -> a
fst (x,_) = x

```

```

snd           :: (a,b) -> b
snd (_,y)     = y

curry        :: ((a,b) -> c) -> (a -> b -> c)
curry f x y   = f (x,y)

uncurry      :: (a -> b -> c) -> ((a,b) -> c)
uncurry f p   = f (fst p) (snd p)

id           :: a -> a
id x         = x

const       :: a -> b -> a
const k _   = k

(.)         :: (b -> c) -> (a -> b) -> (a -> c)
(f . g) x   = f (g x)

flip        :: (a -> b -> c) -> b -> a -> c
flip f x y  = f y x

($)        :: (a -> b) -> a -> b
f $ x      = f x

until      :: (a -> Bool) -> (a -> a) -> a -> a
until p f x = if p x then x else until p f (f x)

asTypeOf   :: a -> a -> a
asTypeOf   = const

primitive error :: String -> a

undefined   :: a
undefined | False = undefined

-- Standard functions on rational numbers {PreludeRatio} -----

data Integral a => Ratio a = a ::% a deriving (Eq)
type Rational      = Ratio Integer

(%)               :: Integral a => a -> a -> Ratio a
x % y            = reduce (x * signum y) (abs y)

```

```

reduce          :: Integral a => a -> a -> Ratio a
reduce x y | y == 0      = error "Ratio.%: zero denominator"
              | otherwise = (x `quot` d) :% (y `quot` d)
              where d = gcd x y

numerator, denominator :: Integral a => Ratio a -> a
numerator (x :% y)      = x
denominator (x :% y)    = y

instance Integral a => Ord (Ratio a) where
    compare (x:%y) (x':%y') = compare (x*y') (x'*y)

instance Integral a => Num (Ratio a) where
    (x:%y) + (x':%y') = reduce (x*y' + x'*y) (y*y')
    (x:%y) * (x':%y') = reduce (x*x') (y*y')
    negate (x :% y)    = negate x :% y
    abs (x :% y)        = abs x :% y
    signum (x :% y)     = signum x :% 1
    fromInteger x       = fromInteger x :% 1
    fromInt              = intToRatio

-- Hugs optimises code of the form fromRational (intToRatio x)
intToRatio :: Integral a => Int -> Ratio a
intToRatio x = fromInt x :% 1

instance Integral a => Real (Ratio a) where
    toRational (x:%y) = toInteger x :% toInteger y

instance Integral a => Fractional (Ratio a) where
    (x:%y) / (x':%y') = (x*y') % (y*x')
    recip (x:%y)       = if x < 0 then (-y) :% (-x) else y :% x
    fromRational (x:%y) = fromInteger x :% fromInteger y
    fromDouble      = doubleToRatio

-- Hugs optimises code of the form fromRational (doubleToRatio x)
doubleToRatio :: Integral a => Double -> Ratio a
doubleToRatio x
    | n>=0      = (fromInteger m * fromInteger b ^ n) % 1
    | otherwise = fromInteger m % (fromInteger b ^ (-n))
    where (m,n) = decodeFloat x
          b      = floatRadix x

```

```

instance Integral a => RealFrac (Ratio a) where
    properFraction (x:%y) = (fromIntegral q, r:%y)
                          where (q,r) = quotRem x y

instance Integral a => Enum (Ratio a) where
    toEnum      = fromInt
    fromEnum    = truncate
    enumFrom    = numericEnumFrom
    enumFromThen = numericEnumFromThen

instance (Read a, Integral a) => Read (Ratio a) where
    readsPrec p = readParen (p > 7)
                  (\r -> [(x%y,u) | (x,s) <- reads r,
                                   ("% ",t) <- lex s,
                                   (y,u) <- reads t ])

instance Integral a => Show (Ratio a) where
    showsPrec p (x:%y) = showParen (p > 7)
                          (shows x . showString " % " . shows y)

approxRational      :: RealFrac a => a -> a -> Rational
approxRational x eps = simplest (x-eps) (x+eps)
  where simplest x y | y < x      = simplest y x
                    | x == y      = xr
                    | x > 0       = simplest' n d n' d'
                    | y < 0       = - simplest' (-n') d' (-n) d
                    | otherwise = 0 :% 1
                    where xr@(n:%d) = toRational x
                          (n':%d') = toRational y
    simplest' n d n' d'      -- assumes 0 < n%d < n'%d'
    | r == 0                 = q :% 1
    | q /= q'                = (q+1) :% 1
    | otherwise              = (q*n'+d') :% n'
    where (q,r)              = quotRem n d
          (q',r')            = quotRem n' d'
          (n':%d'')          = simplest' d' r' d r

-- Standard list functions {PreludeList} -----

head      :: [a] -> a
head (x:_) = x

```

```

last          :: [a] -> a
last [x]      = x
last (_:xs)   = last xs

tail          :: [a] -> [a]
tail (_:xs)   = xs

init         :: [a] -> [a]
init [x]      = []
init (x:xs)   = x : init xs

null         :: [a] -> Bool
null []       = True
null (_:_)    = False

(++)         :: [a] -> [a] -> [a]
[] ++ ys      = ys
(x:xs) ++ ys  = x : (xs ++ ys)

map          :: (a -> b) -> [a] -> [b]
map f xs      = [ f x | x <- xs ]

filter       :: (a -> Bool) -> [a] -> [a]
filter p xs   = [ x | x <- xs, p x ]

concat       :: [[a]] -> [a]
concat        = foldr (++) []

length       :: [a] -> Int
length        = foldl' (\n _ -> n + 1) 0

(!!)         :: [b] -> Int -> b
(x:_) !! 0    = x
(_:xs) !! n | n>0 = xs !! (n-1)
(_:_ ) !! _   = error "Prelude.!!: negative index"
[] !! _       = error "Prelude.!!: index too large"

foldl        :: (a -> b -> a) -> a -> [b] -> a
foldl f z []  = z
foldl f z (x:xs) = foldl f (f z x) xs

```

```

foldl'          :: (a -> b -> a) -> a -> [b] -> a
foldl' f a []   = a
foldl' f a (x:xs) = (foldl' f $! f a x) xs

foldl1         :: (a -> a -> a) -> [a] -> a
foldl1 f (x:xs) = foldl f x xs

scanl          :: (a -> b -> a) -> a -> [b] -> [a]
scanl f q xs   = q : (case xs of
                        []    -> []
                        x:xs  -> scanl f (f q x) xs)

scanl1        :: (a -> a -> a) -> [a] -> [a]
scanl1 f (x:xs) = scanl f x xs

foldr         :: (a -> b -> b) -> b -> [a] -> b
foldr f z []   = z
foldr f z (x:xs) = f x (foldr f z xs)

foldr1        :: (a -> a -> a) -> [a] -> a
foldr1 f [x]   = x
foldr1 f (x:xs) = f x (foldr1 f xs)

scanr         :: (a -> b -> b) -> b -> [a] -> [b]
scanr f q0 []  = [q0]
scanr f q0 (x:xs) = f x q : qs
                  where qs@(q:_) = scanr f q0 xs

scanr1        :: (a -> a -> a) -> [a] -> [a]
scanr1 f [x]   = [x]
scanr1 f (x:xs) = f x q : qs
                  where qs@(q:_) = scanr1 f xs

iterate       :: (a -> a) -> a -> [a]
iterate f x    = x : iterate f (f x)

repeat        :: a -> [a]
repeat x       = xs where xs = x:xs

replicate     :: Int -> a -> [a]
replicate n x  = take n (repeat x)

```

```

cycle          :: [a] -> [a]
cycle []       = error "Prelude.cycle: empty list"
cycle xs      = xs' where xs'=xs++xs'

take          :: Int -> [a] -> [a]
take 0 _      = []
take _ []     = []
take n (x:xs) | n>0 = x : take (n-1) xs
take _ _      = error "Prelude.take: negative argument"

drop          :: Int -> [a] -> [a]
drop 0 xs     = xs
drop _ []     = []
drop n (_:xs) | n>0 = drop (n-1) xs
drop _ _      = error "Prelude.drop: negative argument"

splitAt       :: Int -> [a] -> ([a], [a])
splitAt 0 xs  = ([],xs)
splitAt _ []  = ([],[])
splitAt n (x:xs) | n>0 = (x:xs',xs'') where (xs',xs'') = splitAt (n-1) xs
splitAt _ _   = error "Prelude.splitAt: negative argument"

takeWhile     :: (a -> Bool) -> [a] -> [a]
takeWhile p [] = []
takeWhile p (x:xs)
  | p x      = x : takeWhile p xs
  | otherwise = []

dropWhile     :: (a -> Bool) -> [a] -> [a]
dropWhile p [] = []
dropWhile p xs@(x:xs')
  | p x      = dropWhile p xs'
  | otherwise = xs

span, break   :: (a -> Bool) -> [a] -> ([a],[a])
span p []     = ([],[])
span p xs@(x:xs')
  | p x      = (x:ys, zs)
  | otherwise = ([],xs)
              where (ys,zs) = span p xs'

break p       = span (not . p)

```

```

lines    :: String -> [String]
lines ""  = []
lines s   = let (l,s') = break ('\n'==) s
              in l : case s' of []      -> []
                      (_:s'') -> lines s''

words    :: String -> [String]
words s   = case dropWhile isSpace s of
              "" -> []
              s' -> w : words s''
              where (w,s'') = break isSpace s'

unlines  :: [String] -> String
unlines  = concatMap (\l -> l ++ "\n")

unwords  :: [String] -> String
unwords [] = []
unwords ws = foldr1 (\w s -> w ++ ' ':s) ws

reverse  :: [a] -> [a]
reverse  = foldl (flip (:)) []

and, or   :: [Bool] -> Bool
and      = foldr (&&) True
or       = foldr (||) False

any, all  :: (a -> Bool) -> [a] -> Bool
any p    = or . map p
all p    = and . map p

elem, notElem :: Eq a => a -> [a] -> Bool
elem     = any . (==)
notElem  = all . (/=)

lookup    :: Eq a => a -> [(a,b)] -> Maybe b
lookup k []      = Nothing
lookup k ((x,y):xys)
  | k==x        = Just y
  | otherwise   = lookup k xys

sum, product :: Num a => [a] -> a
sum          = foldl' (+) 0
product     = foldl' (*) 1

```

```

maximum, minimum :: Ord a => [a] -> a
maximum           = foldl1 max
minimum          = foldl1 min

concatMap        :: (a -> [b]) -> [a] -> [b]
concatMap f      = concat . map f

zip              :: [a] -> [b] -> [(a,b)]
zip              = zipWith (\a b -> (a,b))

zip3             :: [a] -> [b] -> [c] -> [(a,b,c)]
zip3             = zipWith3 (\a b c -> (a,b,c))

zipWith          :: (a->b->c) -> [a]->[b]->[c]
zipWith z (a:as) (b:bs) = z a b : zipWith z as bs
zipWith _ _ _      = []

zipWith3         :: (a->b->c->d) -> [a]->[b]->[c]->[d]
zipWith3 z (a:as) (b:bs) (c:cs)
                = z a b c : zipWith3 z as bs cs
zipWith3 _ _ _ _  = []

unzip            :: [(a,b)] -> ([a],[b])
unzip            = foldr \(a,b) ~(as,bs) -> (a:as, b:bs) ([], [])

unzip3           :: [(a,b,c)] -> ([a],[b],[c])
unzip3           = foldr \(a,b,c) ~(as,bs,cs) -> (a:as,b:bs,c:cs)
                  ([],[],[])

-- PreludeText -----

reads           :: Read a => ReadS a
reads           = readsPrec 0

shows           :: Show a => a -> ShowS
shows           = showsPrec 0

read            :: Read a => String -> a
read s          = case [x | (x,t) <- reads s, ("","") <- lex t] of
                    [x] -> x
                    []  -> error "Prelude.read: no parse"
                    _   -> error "Prelude.read: ambiguous parse"

```

```

showChar    :: Char -> ShowS
showChar    = (:)

showString  :: String -> ShowS
showString  = (++)

showParen   :: Bool -> ShowS -> ShowS
showParen b p = if b then showChar '(' . p . showChar ')' else p

showField   :: Show a => String -> a -> ShowS
showField m v = showString m . showChar '=' . shows v

readParen   :: Bool -> ReadS a -> ReadS a
readParen b g = if b then mandatory else optional
                where optional r = g r ++ mandatory r
                      mandatory r = [(x,u) | ("(",s) <- lex r,
                                             (x,t)  <- optional s,
                                             (")",u) <- lex t  ]

readField    :: Read a => String -> ReadS a
readField m s0 = [ r | (t, s1) <- lex s0, t == m,
                      ("=",s2) <- lex s1,
                      r      <- reads s2 ]

lex          :: ReadS String
lex ""       = [("", "")]
lex (c:s) | isSpace c = lex (dropWhile isSpace s)
lex ('\':s)  = [('\':ch++"', t) | (ch,'\':t) <- lexLitChar s,
                ch /= "" ]
lex ('':s)   = [('':str, t) | (str,t) <- lexString s]

where
lexString ('':s) = [("\",s)]
lexString s = [(ch++str, u)
               | (ch,t) <- lexStrItem s,
               (str,u) <- lexString t ]

lexStrItem ('\': '&':s) = [("\&",s)]
lexStrItem ('\': c:s) | isSpace c
  = [("",t) | '\':t <- [dropWhile isSpace s]]
lexStrItem s = lexLitChar s

```

```

lex (c:s) | isSingle c = [(c),s]
  | isSym c      = [(c:sym,t)      | (sym,t) <- [span isSym s]]
  | isAlpha c   = [(c:nam,t)      | (nam,t) <- [span isIdChar s]]
  | isDigit c   = [(c:ds++fe,t)    | (ds,s) <- [span isDigit s],
                    (fe,t) <- lexFracExp s    ]
  | otherwise   = []      -- bad character
  where
    isSingle c = c `elem` ",;()[ ]{ }_`"
    isSym c    = c `elem` "!@#$$%&*+./<=>?\\|^|:~"
    isIdChar c = isAlphaNum c || c `elem` "_'"

lexFracExp ('.':s) = [( '.' : ds ++ e, u) | (ds,t) <- lexDigits s,
                    (e,u) <- lexExp t    ]
lexFracExp s      = [("",s)]

lexExp (e:s) | e `elem` "eE"
  = [(e:c:ds,u) | (c:t) <- [s], c `elem` "+-.",
                    (ds,u) <- lexDigits t] ++
  [(e:ds,t)    | (ds,t) <- lexDigits s]
lexExp s = [("",s)]

lexDigits      :: ReadS String
lexDigits      = nonnull isDigit

nonnull        :: (Char -> Bool) -> ReadS String
nonnull p s    = [(cs,t) | (cs@(_:_),t) <- [span p s]]

lexLitChar     :: ReadS String
lexLitChar ('\\':s) = [( '\\':esc, t) | (esc,t) <- lexEsc s]
  where
    lexEsc (c:s)      | c `elem` "abfnrtv\\\\" = [(c),s]
    lexEsc ('^':c:s) | c >= '@' && c <= '_'   = [( '^',c),s]
    lexEsc s@(d:_)   | isDigit d             = lexDigits s
    lexEsc s@(c:_)   | isUpper c
                      = let table = ('\DEL',"DEL") : asciiTab
                        in case [(mne,s') | (c, mne) <- table,
                                ([],s') <- [lexmatch mne s]]
                            of (pr:_) -> [pr]
                               []      -> []

    lexEsc _          = []

lexLitChar (c:s) = [(c),s]
lexLitChar ""    = []

```

```

isOctDigit c = c >= '0' && c <= '7'
isHexDigit c = isDigit c || c >= 'A' && c <= 'F'
              || c >= 'a' && c <= 'f'

lexmatch      :: (Eq a) => [a] -> [a] -> ([a],[a])
lexmatch (x:xs) (y:ys) | x == y = lexmatch xs ys
lexmatch xs      ys           = (xs,ys)

asciiTab = zip ['\NUL'..' ']
          ["NUL", "SOH", "STX", "ETX", "EOT", "ENQ", "ACK", "BEL",
           "BS",  "HT",  "LF",  "VT",  "FF",  "CR",  "SO",  "SI",
           "DLE", "DC1", "DC2", "DC3", "DC4", "NAK", "SYN", "ETB",
           "CAN", "EM",  "SUB", "ESC", "FS",  "GS",  "RS",  "US",
           "SP"]

readLitChar      :: ReadS Char
readLitChar ('\':s) = readEsc s
  where
    readEsc ('a':s) = [('\a',s)]
    readEsc ('b':s) = [('\b',s)]
    readEsc ('f':s) = [('\f',s)]
    readEsc ('n':s) = [('\n',s)]
    readEsc ('r':s) = [('\r',s)]
    readEsc ('t':s) = [('\t',s)]
    readEsc ('v':s) = [('\v',s)]
    readEsc ('\':s) = [('\',s)]
    readEsc ('':s) = [('',s)]
    readEsc ('\':s) = [('\',s)]
    readEsc ('^':c:s) | c >= '@' && c <= '_'
                      = [(toEnum (fromEnum c - fromEnum '@'), s)]
    readEsc s@(d:_) | isDigit d
                    = [(toEnum n, t) | (n,t) <- readDec s]
    readEsc ('o':s) = [(toEnum n, t) | (n,t) <- readOct s]
    readEsc ('x':s) = [(toEnum n, t) | (n,t) <- readHex s]
    readEsc s@(c:_) | isUpper c
                    = let table = ('\DEL',"DEL") : asciiTab
                      in case [(c,s') | (c, mne) <- table,
                                   ([,s') <- [lexmatch mne s]]
                        of (pr:_) -> [pr]
                          []      -> []
    readEsc _ = []
readLitChar (c:s) = [(c,s)]

```

```

showLitChar          :: Char -> ShowS
showLitChar c | c > '\DEL' = showChar '\\' .
                    protectEsc isDigit (shows (fromEnum c))
showLitChar '\DEL'     = showString "\\DEL"
showLitChar '\\'      = showString "\\\"
showLitChar c | c >= ' ' = showChar c
showLitChar '\a'      = showString "\\a"
showLitChar '\b'      = showString "\\b"
showLitChar '\f'      = showString "\\f"
showLitChar '\n'      = showString "\\n"
showLitChar '\r'      = showString "\\r"
showLitChar '\t'      = showString "\\t"
showLitChar '\v'      = showString "\\v"
showLitChar '\SO'     = protectEsc ('H'==) (showString "\\SO")
showLitChar c         = showString ('\\' : snd (asciiTab!!fromEnum c))

protectEsc p f        = f . cont
  where cont s@(c:_) | p c = "\\&" ++ s
        cont s           = s

-- Unsigned readers for various bases
readDec, readOct, readHex :: Integral a => ReadS a
readDec = readInt 10 isDigit (\d -> fromEnum d - fromEnum '0')
readOct = readInt  8 isOctDigit (\d -> fromEnum d - fromEnum '0')
readHex = readInt 16 isHexDigit hex
  where hex d = fromEnum d -
              (if isDigit d
               then fromEnum '0'
               else fromEnum (if isUpper d then 'A' else 'a') - 10)

-- readInt reads a string of digits using an arbitrary base.
-- Leading minus signs must be handled elsewhere.

readInt :: Integral a => a -> (Char -> Bool) -> (Char -> Int) -> ReadS a
readInt radix isDig digToInt s =
  [(foldl1 (\n d -> n * radix + d) (map (fromIntegral . digToInt) ds), r)
   | (ds,r) <- nonnull isDig s ]

```

```

-- showInt is used for positive numbers only
showInt    :: Integral a => a -> ShowS
showInt n r | n < 0 = error "Numeric.showInt: can't show negative numbers"
            | otherwise =
                let (n',d) = quotRem n 10
                    r'      = toEnum (fromEnum '0' + fromIntegral d) : r
                in if n' == 0 then r' else showInt n' r'

readSigned :: Real a => ReadS a -> ReadS a
readSigned readPos = readParen False read'
    where read' r = read'' r ++
        [(-x,t) | ("-",s) <- lex r,
                (x,t)  <- read'' s]
        read'' r = [(n,s) | (str,s) <- lex r,
                            (n,"") <- readPos str]

showSigned    :: Real a => (a -> ShowS) -> Int -> a -> ShowS
showSigned showPos p x = if x < 0 then showParen (p > 6)
                        (showChar '-' . showPos (-x))
                        else showPos x

readFloat     :: RealFloat a => ReadS a
readFloat r   = [(fromRational ((n%1)*10^(k-d)),t) | (n,d,s) <- readFix r,
                                                (k,t)  <- readExp s]
    where readFix r = [(read (ds++ds'), length ds', t)
                      | (ds, s) <- lexDigits r
                        , (ds',t) <- lexFrac s ]

    lexFrac ('.':s) = lexDigits s
    lexFrac s      = [("",s)]

    readExp (e:s) | e `elem` "eE" = readExp' s
    readExp s                       = [(0,s)]

    readExp' ('-':s) = [(-k,t) | (k,t) <- readDec s]
    readExp' ('+':s) = readDec s
    readExp' s       = readDec s

-- Monadic I/O: -----
--data IO a          -- builtin datatype of IO actions
data IOError        -- builtin datatype of IO error codes
type FilePath = String -- file pathnames are represented by strings

```

```

instance Show (IO a) where
    showsPrec p f = showString "<<IO action>>"

primitive primbindIO    "rbindIO" :: IO a -> (a -> IO b) -> IO b
primitive primretIO    "runitIO" :: a -> IO a
primitive catch        "lbindIO" :: IO a -> (IOError -> IO a) -> IO a
primitive ioError      "lunitIO" :: IOError -> IO a
primitive putChar      :: Char -> IO ()
primitive putStr       :: String -> IO ()
primitive getChar      :: IO Char
primitive userError    :: String -> IOError

print    :: Show a => a -> IO ()
print    = putStrLn . show

putStrLn :: String -> IO ()
putStrLn s = do putStr s
                putChar '\n'
getLine   :: IO String
getLine   = do c <- getChar
                if c=='\n' then return ""
                else do cs <- getLine
                        return (c:cs)
-- raises an exception instead of an error
readIO    :: Read a => String -> IO a
readIO s  = case [x | (x,t) <- reads s, ("","") <- lex t] of
                [x] -> return x
                []  -> ioError (userError "PreludeIO.readIO: no parse")
                _   -> ioError (userError
                                "PreludeIO.readIO: ambiguous parse")

readLn    :: Read a => IO a
readLn    = do l <- getLine
                r <- readIO l
                return r

primitive getContents    :: IO String
primitive writeFile     :: FilePath -> String -> IO ()
primitive appendFile    :: FilePath -> String -> IO ()
primitive readFile      :: FilePath -> IO String

interact :: (String -> String) -> IO ()
interact f = getContents >>= (putStr . f)

```

```

instance Functor IO where
    fmap f x = x >>= (return . f)

instance Monad IO where
    (>>=) = primbindIO
    return = primretIO

-- Hooks for primitives: -----
-- Do not mess with these!

data Addr      -- builtin datatype of C pointers
data Word      -- builtin datatype of unsigned ints
data ForeignObj
data StablePtr a

primitive unsafeCoerce "primUnsafeCoerce" :: a -> b

data Obj = Obj

toObj :: a -> Obj
toObj  = unsafeCoerce

fromObj :: Obj -> a
fromObj = unsafeCoerce

newtype IO a = IO ((IOError -> IOResult) -> (a -> IOResult) -> IOResult)

data IOResult
  = Hugs_ExitWith      Int
  | Hugs_Error         IOError
  | Hugs_ForkThread    IOResult IOResult
  | Hugs_DeadThread
  | Hugs_YieldThread   IOResult
  | Hugs_Return        Obj

data IOFinished a
  = Finished_ExitWith Int
  | Finished_Error    IOError
  | Finished_Return   a

hugsPutStr :: String -> IO ()
hugsPutStr = putStr

```

```

hugsIORun  :: IO a -> Either Int a
hugsIORun m =
  case basicIORun (runAndShowError m) of
    Finished_ExitWith i -> Left i
    Finished_Error      _ -> Left 1
    Finished_Return    a -> Right a
  where
    runAndShowError :: IO a -> IO a
    runAndShowError m =
      m `catch` \err -> do
        putChar '\n'
        putStr (ioeGetErrorString err)
        primExitWith 1

basicIORun :: IO a -> IOFinished a
basicIORun (IO m) = loop [m Hugs_Error (Hugs_Return . toObj)]

threadToIOResult :: IO a -> IOResult
threadToIOResult (IO m) = m Hugs_Error (const Hugs_DeadThread)

-- This is the queue of *runnable* threads.
-- There may be blocked processes inside
loop :: [IOResult] -> IOFinished a
loop [] = error "no more threads (deadlock?)"
loop [Hugs_Return a] = Finished_Return (fromObj a)
loop (Hugs_Return a:r) = loop (r ++ [Hugs_Return a])
loop (Hugs_Error e:_) = Finished_Error e
loop (Hugs_ExitWith i:_) = Finished_ExitWith i
loop (Hugs_DeadThread:r) = loop r
loop (Hugs_ForkThread a b:r) = loop (a:b:r)
loop (Hugs_YieldThread a:r) = loop (r ++ [a])
loop _ = undefined

primExitWith :: Int -> IO a
primExitWith c = IO (\ f s -> Hugs_ExitWith c)

primitive ioeGetErrorString "primShowIOError" :: IOError -> String

instance Show IOError where
  showsPrec p x = showString (ioeGetErrorString x)

primCompAux :: Ord a => a -> a -> Ordering -> Ordering
primCompAux x y o = case compare x y of EQ -> o; LT -> LT; GT -> GT

```

```
primPmInt      :: Num a => Int -> a -> Bool
primPmInt n x  = fromInt n == x

primPmInteger  :: Num a => Integer -> a -> Bool
primPmInteger n x = fromInteger n == x

primPmFlt      :: Fractional a => Double -> a -> Bool
primPmFlt n x  = fromDouble n == x

-- The following primitives are only needed if (n+k) patterns are enabled:
primPmNpk      :: Integral a => Int -> a -> Maybe a
primPmNpk n x  = if n' <= x then Just (x-n') else Nothing
                where n' = fromInt n

primPmSub      :: Integral a => Int -> a -> a
primPmSub n x  = x - fromInt n

-- End of Hugs standard prelude -----
```


Anhang D: GBG55

Allgemeines Grundbuchsgesetz 1955

(GBG 1955)

ERSTES HAUPTSTÜCK.

Von den Grundbüchern im allgemeinen.

§ 1. Das Grundbuch besteht aus dem Hauptbuch und der Urkundensammlung.

Anmerkungen

1. Zum Hauptbuch siehe insbesondere § 2 Abs. 1 sowie § 68 Abs. 1 AllgGAG, BGBl. Nr. 2/1930, zur Urkundensammlung § 6 sowie § 13 AllgGAG.
2. Im automationsunterstützten Grundbuch kommt noch das Verzeichnis der gelöschten Eintragungen hinzu (§ 3 Abs. 1 GUG, BGBl. Nr. 550/1980).
3. Die Grundbuchsmappe ist kein Bestandteil des Grundbuchs (§ 3 AllgGAG, BGBl. Nr. 2/1930).

§ 2. (1) Das Hauptbuch wird aus den Grundbuchseinlagen gebildet.

(2) Die Grundbuchseinlagen sind bestimmt zur Eintragung:

1. der Grundbuchskörper und ihrer Änderungen;
2. der sich auf die Grundbuchskörper beziehenden dinglichen Rechte und ihrer Änderungen.

Anmerkungen

1. Zu den Grundbuchseinlagen siehe die §§ 2, 4, 6 bis 12 AllgGAG, BGBl. Nr. 2/1930, zum Grundbuchskörper die §§ 5 und 7 AllgGAG.
2. Für das Baurecht sind eigene Grundbuchseinlagen zu eröffnen (§ 5 BauRG, RGBL. Nr. 86/1912).

§ 3. (1) Jeder Grundbuchskörper ist als ein Ganzes zu behandeln.

(2) Sein Umfang kann nur durch die grundbücherliche Ab- und Zuschreibung von einzelnen Liegenschaften oder Liegenschaftsteilen geändert werden.

(3) Wenn alle in einer Grundbuchseinlage eingetragenen Liegenschaften abgeschrieben worden sind (§ 11) oder wenn sie aufgehört haben, ein Gegenstand des Grundbuches zu sein, ist die Einlage zu löschen.

Anmerkungen

1. Zu Abs. 1: Vgl. die §§ 10 bis 13.
2. Zu Abs. 2: Die näheren Bestimmungen enthält das LiegTeilG, BGBl.

§ 4. Die Erwerbung, Übertragung, Beschränkung und Aufhebung der bürgerlichen Rechte (§ 9) wird nur durch ihre Eintragungen in das Hauptbuch erwirkt.

§ 5. In das Hauptbuch sind die wesentlichen Bestimmungen der bürgerlichen Rechte einzutragen. Lassen sie eine kurze Fassung nicht zu, so ist im Hauptbuch eine Berufung auf die genau

zu bezeichnenden Stellen der Urkunden, die der Eintragung zugrunde liegen, mit der Wirkung zulässig, daß die bezogenen Stellen als im Hauptbuch eingetragen anzusehen sind.

§ 6. (1) Von jeder Urkunde, auf Grund deren eine bücherliche Eintragung vorgenommen wird, ist bei dem Grundbuch eine beglaubigte Abschrift zurückzubehalten.

(2) Diese Abschriften bilden die Urkundensammlung.

Anmerkungen

Vgl. § 577 Geo, BGBl. Nr. 264/1951.

§ 7. (1) Das Grundbuch ist öffentlich.

(2) Jedermann kann das Grundbuch in Gegenwart eines Grundbuchsbeamten einsehen und Abschriften oder Auszüge daraus erheben; der Grundbuchsführer hat sie zu erteilen.

Anmerkungen

1. Vgl. §§ 583 bis 587 Geo, BGBl. Nr. 264/1951.

2. Abweichende Bestimmungen für das automationsunterstützte Grundbuch; § 5 GUG, BGBl. Nr. 550/1980.

ZWEITES HAUPTSTÜCK.

Von den bücherlichen Eintragungen.

ERSTER ABSCHNITT.

Von den Eintragungen im allgemeinen.

1. Arten der Eintragung.

§ 8. Die grundbücherlichen Eintragungen sind:

1. Einverleibungen (unbedingte Rechtserwerbungen oder Löschungen - Intabulationen oder Extabulationen), die ohne weitere Rechtfertigung oder

2. Vormerkungen (bedingte Rechtserwerbungen oder Löschungen - Pränotationen), die nur unter der Bedingung ihrer nachfolgenden Rechtfertigung die Erwerbung, Übertragung, Beschränkung oder Erlöschung bücherlicher Rechte bewirken, oder

3. bloße Anmerkungen.

Anmerkungen

Weitere Eintragungen sind

1. Ab- und Zuschreibungen

2. Ersichtlichmachungen

3. (schlichte) Löschungen.

2. Gegenstand der Einverleibung oder Vormerkung.

§ 9. Im Grundbuch können nur dingliche Rechte und Lasten, ferner das Wiederkaufs- und das Vorkaufsrecht (§§ 1070 und 1073 ABGB.) sowie das Bestandrecht (§ 1095 ABGB.) eingetragen werden.

Anmerkungen

Überdies können die Vorrangseinräumung und das Veräußerungs- und Belastungsverbot nach § 364c ABGB, JGS Nr. 946/1811, einverleibt und vorgemerkt werden.

2 Besondere Bestimmungen in Ansehung

a) des Eigentumsrechtes:

§ 10. Das Miteigentum an den zu einem Grundbuchkörper gehörigen Liegenschaften kann, sofern nicht besondere Vorschriften eine Ausnahme zulassen, nur nach Anteilen, die im Verhältnisse zum Ganzen bestimmt sind, zum Beispiel zur Hälfte, zu einem Drittel, eingetragen werden.

Anmerkungen

Über die Teilung von Gebäuden nach materiellen Anteilen siehe RGBl. Nr. 50/1879, über Bäume als selbständige Vermögensobjekte siehe Art. III und IV, RGBl. Nr. 77/1897.

§ 11. Eintragungen zur Erwerbung des Eigentumes einzelner Bestandteile eines Grundbuchkörpers sind nur nach den Bestimmungen des Liegenschaftsteilungsgesetzes, BGBl. Nr. 3/1930, zulässig.

Anmerkungen

Siehe auch § 74.

b) der Dienstbarkeiten und Reallasten:

§ 12. (1) Bei Dienstbarkeiten und Reallasten muß Inhalt und Umfang des einzutragenden Rechtes möglichst bestimmt angegeben werden; einer Angabe des Geldwertes bedarf es nicht.

(2) Sollen Dienstbarkeiten auf bestimmte räumliche Grenzen beschränkt sein, so müssen diese genau bezeichnet werden.

c) des Pfandrechtes:

§ 13. (1) Das Pfandrecht kann entweder auf einen ganzen Grundbuchkörper oder bei Miteigentum auf den Anteil eines jeden Miteigentümers, dagegen nicht auf einzelne Bestandteile eines Grundbuchkörpers oder auf einen Teil des einem Miteigentümer im Grundbuche zugeschriebenen Anteiles eingetragen werden.

(2) Die Übertragung einer Hypothekarforderung und die Erwerbung des Afterpfandrechtes ist zulässig an der ganzen Forderung sowie an einem verhältnismäßig oder ziffermäßig bestimmten Teile.

Anmerkungen

Sonderbestimmungen für die Eintragung von Simultanhypotheken enthalten die §§ 105 bis 114.

§ 14. (1) Das Pfandrecht kann nur für eine ziffermäßig bestimmte Geldsumme eingetragen werden. Bei einer verzinslichen Forderung muß auch die Höhe der Zinsen eingetragen werden.

(2) Sollen Forderungen, die aus einem gegebenen Kredite, aus einer übernommenen Geschäftsführung oder aus dem Titel der Gewährleistung oder des Schadenersatzes entstehen

können, pfandrechlich sichergestellt werden, so ist in der Urkunde, auf Grund derer die Eintragung vorgenommen werden soll, ein Höchstbetrag anzugeben, bis zu dem der Kredit oder die Haftung reichen soll.

(3) Fehlt die Angabe dieses Betrages in der Urkunde, so muß er in dem Ansuchen ausgedrückt werden.

(4) Hält sich im letzteren Falle der, gegen den die Eintragung erwirkt wird, dadurch beschwert, daß ein zu großer Betrag zur Eintragung angegeben wurde, so kann er innerhalb der ihm zustehenden Rekursfrist seine Verminderung verlangen. Das Gericht, von dem die Eintragung bewilligt worden ist, hat darüber nach Einvernehmung der Parteien zu erkennen und den Betrag nach billigem Ermessen festzusetzen.

Anmerkungen

Zu Abs. 1: Die Eintragung von Wertsicherungsklauseln ist daher unzulässig.

§ 15. (1) Das Pfandrecht kann für dieselbe Forderung ungeteilt auf zwei oder mehrere Grundbuchkörper oder Hypothekarforderungen eingetragen werden (Simultanhypothek).

(2) Der Gläubiger ist in solchen Fällen berechtigt, die Bezahlung der ganzen Forderung aus jeder einzelnen Pfandsache zu verlangen.

Anmerkungen

Siehe auch die §§ 105 bis 114.

§ 16. Das für eine Forderung erworbene Pfandrecht kommt, abgesehen von besonderen Bestimmungen, auch den Prozeß- und Exekutionskosten zu.

Anmerkungen

Vgl. § 216 Abs. 2 EO, RGBl. Nr. 79/1896.

§ 17. Dreijährige Rückstände von Zinsen, die aus einem Vertrag oder aus dem Gesetze gebühren, genießen gleichen Rang mit dem Kapital.

§ 18. Den drei Jahre rückständigen Ansprüchen auf jährliche Renten, Unterhaltsgelder und andere wiederkehrende Zahlungen gebührt der gleiche Rang, der dem Bezugsrechte selbst zukommt.

d) der Bestandrechte:

§ 19. Bei Einverleibung oder Vormerkung von Bestandrechten ist die Angabe einer Summe zur Sicherstellung eines allfälligen Schadenersatzes (§ 1121 ABGB.) nicht notwendig.

3. Gegenstand der Anmerkung.

§ 20. Die grundbücherlichen Anmerkungen können erfolgen:

a) zur Ersichtlichmachung persönlicher Verhältnisse, insbesondere von Beschränkungen der Vermögensverwaltung, mit der Rechtsfolge, daß, wer immer in der betreffenden Grundbucheinlage eine Eintragung erwirkt, sich auf die Unkenntnis dieser Verhältnisse nicht berufen

kann; zum Beispiel die Anmerkung der Minderjährigkeit, der Bestellung eines Sachwalters (§ 248 Abs. 2 AußStrG), der Verlängerung der Minderjährigkeit, des Eintritts der Volljährigkeit, der Konkurseröffnung oder

b) zur Begründung bestimmter, nach den Vorschriften dieses oder eines anderen Gesetzes damit verbundener Rechtswirkungen, wie zum Beispiel die Anmerkung der Rangordnung, der Abschreibung von Grundstücken, der Simultanhaftung, der Aufkündigung einer Hypothekarforderung, der Streitanhängigkeit, der Zwangsverwaltung, der Erteilung des Zuschlages.

Anmerkungen

Siehe insbesondere die §§ 52 bis 73.

4. Bücherlicher Vormann.

§ 21. Eintragungen sind nur wider den zulässig, der zur Zeit des Ansehens als Eigentümer der Liegenschaft oder des Rechtes, in Ansehung deren die Eintragung erfolgen soll, im Grundbuch erscheint oder doch gleichzeitig als solcher einverleibt oder vorgemerkt wird.

Anmerkungen

Ausnahmen siehe §§ 22 bis 24.

§ 22. Ist eine Liegenschaft oder ein bürgerliches Recht auf mehrere Personen nacheinander außerbücherlich übertragen worden, so kann der letzte Übernehmer unter Nachweisung seiner Vormänner verlangen, daß die bürgerliche Übertragung unmittelbar auf seine Person vorgenommen werde. Ist eine Hypothekarforderung, die außerbücherlich auf einen Dritten übergegangen ist, getilgt worden, so kann der Schuldner die Löschung des Pfandrechtes ohne vorhergehende Eintragung der außerbücherlichen Übertragung begehren.

§ 23. Wird ein zu einer Verlassenschaft gehöriges unbewegliches Gut oder bürgerliches Recht veräußert, so ist dem Erwerber die Eintragung seines Rechtes unmittelbar nach dem Erblasser zu bewilligen.

§ 24. Inwiefern Gläubiger eines Erben die Sicherstellung auf die ihm angefallenen Liegenschaften oder Forderungen des Erblassers erwirken können, bestimmt § 822 ABGB.

§ 25. Inwiefern grundbücherliche Rechte noch nach der Eröffnung eines Konkurses erworben werden können, bestimmt die Konkursordnung.

Anmerkungen

Siehe § 13 KO, RGBl. Nr. 337/1914 sowie § 56 Abs. 3.

5. Urkunden.

§ 26. (1) Einverleibungen und Vormerkungen können nur auf Grund von Urkunden bewilligt werden, die in der zu ihrer Gültigkeit vorgeschriebenen Form ausgefertigt sind.

(2) Diese Urkunden müssen, wenn es sich um die Erwerbung oder Umänderung eines dinglichen Rechtes handelt, einen gültigen Rechtsgrund enthalten.

Anmerkungen

Zu den Formvorschriften siehe insbesondere das Gesetz betreffend das Erfordernis der notariellen Errichtung einiger Rechtsgeschäfte, RGBL. Nr. 76/1871.

§ 27. (1) Die Urkunden, auf Grund deren eine bücherliche Eintragung geschehen soll, müssen frei von solchen sichtbaren Mängeln sein, durch die ihre Glaubwürdigkeit geschwächt wird, und, wenn sie aus mehreren Bogen bestehen, so geheftet sein, daß kein Bogen unterschoben werden kann.

(2) Sie müssen auch eine solche Bezeichnung der an dem Rechtsgeschäft beteiligten Personen, daß sie nicht mit anderen verwechselt werden können, einschließlich des Geburtsdatums natürlicher Personen sowie die Angabe des Ortes, Tages, Monats und Jahres der Ausfertigung der Urkunde enthalten.

6. Wirkung der Eintragung.

§ 28. Inwiefern Rechte, die dritte Personen im Vertrauen auf die öffentlichen Bücher erwerben, angefochten werden können, wird in den §§ 63 ff. bestimmt.

7. Rangordnung.

§ 29. (1) Die Rangordnung einer Eintragung richtet sich nach dem Zeitpunkt, in dem die Eingabe bei dem Grundbuchsgericht eingelangt ist (§§ 438, 440 ABGB.).

(2) Eintragungen, die infolge gleichzeitig eingelangter Eingaben vorgenommen worden sind, stehen untereinander in gleicher Rangordnung (§ 103).

Anmerkungen

Über die Rangordnung einer Simultanhypothek siehe §§ 110 und 114 Abs. 2.

8. Vorrangseinräumung.

§ 30. (1) Durch Einverleibung oder Vormerkung der Vorrangseinräumung kann die Rangordnung der auf einer Liegenschaft verbücherten Rechte geändert werden. Dazu bedarf es der Einwilligung des zurücktretenden und des vortretenden Berechtigten, ferner, wenn das zurücktretende Recht eine Hypothek ist, des Eigentümers und, wenn es mit dem Recht eines Dritten belastet ist, auch dessen Zustimmung. Umfang und Rang der übrigen verbücherten Rechte werden dadurch nicht berührt.

(2) Das vortretende Recht erhält ohne Beschränkung die Rangstelle des zurücktretenden, wenn es bücherlich unmittelbar auf dieses folgt oder ihm der Vorrang auch von allen Zwischenberechtigten eingeräumt wird.

(3) Hat eine Vorrangseinräumung zwischen nicht unmittelbar aufeinanderfolgenden Rechten ohne Zustimmung der Zwischenberechtigten stattgefunden, so wird dem vortretenden Recht im Umfang und nach der Beschaffenheit des zurücktretenden dessen Rang erworben.

(4) Das vortretende Recht geht, wenn nichts anderes vereinbart ist, dem zurücktretenden Recht auch an seiner ursprünglichen Stelle vor.

(5) Treten mehrere Rechte infolge einer gleichzeitig verbücherten Vorrangseinräumung an die Stelle eines anderen, so geht an dieser Stelle, wenn nichts anderes vereinbart ist, das bis dahin im Range frühere vor.

(6) Nachträgliche Änderungen im Bestand oder Umfange des zurücktretenden Rechtes üben, wenn nichts anderes vereinbart ist, auf den Rang des vortretenden Rechtes keinen Einfluß.

ZWEITER ABSCHNITT.

Von der Einverleibung.

§ 31. (1) Die Einverleibung (§ 8 Z 1) kann nur auf Grund öffentlicher Urkunden oder solcher Privaturkunden geschehen, auf denen die Unterschriften der Parteien gerichtlich oder notariell beglaubigt sind und der Beglaubigungsvermerk bei natürlichen Personen auch das Geburtsdatum enthält.

(2) Die gerichtliche oder notarielle Beglaubigung der Unterschrift auf einer Privaturkunde ist nicht erforderlich, wenn diese Urkunde mit der genehmigenden Erklärung einer Behörde des Bundes oder eines Landes versehen ist, die berufen erscheint, die Interessen desjenigen wahrzunehmen, dessen Recht beschränkt, belastet, aufgehoben oder auf eine andere Person übertragen werden soll.

(3) Die Beglaubigung ausländischer Urkunden wird durch Staatsverträge geregelt. Urkunden, die von der österreichischen Vertretungsbehörde, in deren Sprengel die Urkunde errichtet oder beglaubigt worden ist, oder von der inländischen Vertretungsbehörde des Staates, in dem die Urkunde errichtet oder beglaubigt worden ist, beglaubigt sind, bedürfen keiner weiteren Beglaubigung.

(4) Besteht weder für den Staat, in dem die ausländische Urkunde ausgestellt wurde, eine österreichische Vertretungsbehörde noch für Österreich eine Vertretungsbehörde dieses Staates, so kann das Bundesministerium für Justiz von der nach den bestehenden Vorschriften erforderlichen diplomatischen Beglaubigung (Abs. 3) Nachsicht erteilen.

(5) Das gleiche gilt, wenn die Einholung einer Beglaubigung nach Abs. 3 infolge außergewöhnlicher Verhältnisse unmöglich ist oder doch auf erhebliche Schwierigkeiten stößt.

(6) Auf Grund von Urkunden eines Machthabers kann eine Einverleibung gegen den Machtgeber überdies nur dann bewilligt werden, wenn die von diesem ausgefertigte Vollmacht entweder auf das bestimmte Geschäft lautet oder doch nicht früher als drei Jahre vor dem Ansuchen um die Einverleibung ausgestellt ist.

Anmerkungen

1. Öffentliche Urkunden siehe § 33, Beglaubigung in geringfügigen Grundbuchssachen siehe § 34.
2. In Tirol und Vorarlberg sind auch Legalisatoren zur Beglaubigung berechtigt (RGBl. Nr. 77/1897 und 44/1900)
3. Staatsverträge: Haager Beglaubigungsübereinkommen BGBl. Nr. 27/1968.

§ 32. (1) Privaturkunden, auf Grund deren eine Einverleibung stattfinden soll, müssen außer den Erfordernissen der §§ 26, 27 enthalten:

a) die genaue Angabe der Liegenschaft oder des Rechtes, in betreff deren die Einverleibung erfolgen soll;

b) die ausdrückliche Erklärung desjenigen, dessen Recht beschränkt, belastet, aufgehoben oder auf eine andere Person übertragen werden soll, daß er in die Einverleibung einwillige.

(2) Diese Erklärung kann auch in einer besonderen Urkunde oder in dem Grundbuchsgesuch abgegeben werden. In solchen Fällen muß aber die Urkunde oder das Gesuch, in dem die Erklärung enthalten ist, mit den Erfordernissen zur Einverleibung versehen sein.

Anmerkungen

Zum Abs. 1 lit. b: diese Erklärung wird von der Praxis Aufsandungserklärung genannt.

§ 33. (1) Öffentliche Urkunden, auf Grund deren Einverleibungen stattfinden können, sind:

a) die über Rechtsgeschäfte von einer öffentlichen Behörde oder von einem Notar innerhalb der Grenzen ihrer Amtsbefugnisse aufgenommenen Urkunden, wenn sie mit den im § 32 vorgeschriebenen Erfordernissen versehen sind;

b) die von den Gerichten oder anderen dazu berechtigten Behörden oder Personen aufgenommenen exekutionsfähigen Vergleiche;

c) Zahlungsaufträge über gesetzliche Gebühren und Beiträge sowie Ausweise über rückständige Steuern und öffentliche Abgaben, insoweit sie nach den bestehenden Gesetzen vollziehbar sind;

d) andere Urkunden, die die Eigenschaft eines gerichtlich vollziehbaren Ausspruches einer öffentlichen Behörde haben. Dahin gehören insbesondere rechtskräftige Erkenntnisse, Beschlüsse über bücherliche Einverleibungen und Löschungen zur Ausführung des Verteilungsbeschlusses (§ 237 EO.), Amtsbestätigungen über die freiwillige Versteigerung einer Liegenschaft sowie die Einantwortungs- und Bestätigungsurkunden der Abhandlungsbehörden (§§ 177 und 178 des Kaiserlichen Patentes vom 9. August 1854, RGBI. Nr. 208).

(2) Das Bundesministerium für Justiz ist berechtigt zu erklären, ob und unter welchen Voraussetzungen Einverleibungen auf Grund ausländischer Urkunden stattfinden können, die am Ort ihrer Errichtung als öffentliche Urkunden gelten. Die Erklärung ist für die Gerichte bindend.

§ 34. (1) In geringfügigen Grundbuchssachen wird das zum Zweck einer grundbücherlichen Einverleibung vorgeschriebene Erfordernis der gerichtlichen oder notariellen Beglaubigung der Unterschriften einer Privaturkunde durch die Mitfertigung von zwei glaubwürdigen Personen als Zeugen ersetzt, wenn die Einverleibung in dem einem Gerichtshof erster Instanz zugewiesenen Sprengel, in dem die Urkunde errichtet worden ist, vorgenommen werden soll. Die Zeugen haben die Unterschrift ihres Vor- und Zunamens, die Angabe ihres Gewerbes oder ihrer Beschäftigung, ihres Wohnortes, Alters sowie die Erklärung eigenhändig beizusetzen, daß ihnen der, dessen Unterschrift sie als echt bestätigen, persönlich bekannt sei.

(2) Die Bestimmungen des Abs. 1 finden keine Anwendung:

1. auf landtäfliche Urkunden;

2. auf Vollmachten;

3. auf Urkunden, in denen der Betrag einer Forderung oder der Preis oder der Wert einer Liegenschaft oder eines Rechtes überhaupt nicht bestimmt ist oder in denen die angegebene Summe ohne Zinsen und Nebengebühren den Betrag von 8 000 S übersteigt.

DRITTER ABSCHNITT.

Von der Vormerkung.

a) Zulässigkeit.

§ 35. Wenn die beigebrachte Urkunde nicht alle in den §§ 31 bis 34 festgesetzten besonderen Erfordernisse zur Einverleibung, wohl aber die allgemeinen Erfordernisse (§§ 26, 27) zur grundbücherlichen Eintragung besitzt, kann auf Grund der Urkunde die Vormerkung (§ 8 Z. 2) bewilligt werden.

Anmerkungen

1. Siehe auch die §§ 438, 439, 445 und 453 ABGB, JGS Nr. 946/1811.
2. Nach § 85 „begrift das Begehren um Einverleibung jenes um Vormerkung stillschweigend in sich.“

§ 36. Die Vormerkung zur Erwirkung des Pfandrechtes findet nur dann statt, wenn sowohl die Forderung als auch der Rechtsgrund zum Pfandrecht hinlänglich bescheinigt sind.

§ 37. Die Vormerkung des Wiederkaufs-, Vorkaufs- und Bestandrechtes findet nur dann statt, wenn sowohl der Bestand des Rechtes als die Einwilligung zur Eintragung hinlänglich bescheinigt sind.

§ 38. Die Vormerkung findet statt:

- a) auf Grund gerichtlicher Erkenntnisse erster oder höherer Instanz, durch die das dingliche Recht zwar unbedingt zugesprochen oder abgesprochen wird, die aber noch nicht in Rechtskraft erwachsen sind;
- b) auf Grund gerichtlicher Verfügungen, wodurch die Vormerkung als Exekution zur Sicherstellung bewilligt wird;
- c) auf Grund des Einschreitens öffentlicher Behörden in Fällen, wenn diese nach ihrem Wirkungskreise berufen sind, von Amts wegen die pfandweise Sicherstellung von Ansprüchen des Bundes oder eines Landes zu verfügen.

Anmerkungen

Im Fall der lit. c genügt das Ansuchen der Behörde; die Vorlage einer Urkunde ist nicht erforderlich.

§ 39. Wird der Betrag einer Hypothekarschuld, die aus einem der im § 1425 ABGB. erwähnten Gründe dem Gläubiger nicht gezahlt werden kann oder rücksichtlich deren dieser dem Zahler nach § 1422 ABGB. erst seine Rechte abzutreten hat, gerichtlich erlegt, so findet gegen Beibringung der Amtsurkunde über den gerichtlichen Erlag die Vormerkung zum Zweck der Löschung oder zum Zweck der Übertragung der Forderung auf den Zahler statt.

b) Rechtfertigung.

§ 40. Jede Vormerkung begründet die Erwerbung, Übertragung, Beschränkung oder Aufhebung des dinglichen Rechtes nur unter der Bedingung ihrer Rechtfertigung und nur in dem Umfang, in dem die Rechtfertigung erfolgt.

§ 41. Die Rechtfertigung erfolgt:

a) auf Grund einer zur Einverleibung geeigneten Erklärung dessen, gegen den die Vormerkung bewirkt worden ist;

b) in den Fällen des § 38 durch den Ausweis über den Eintritt der Exekutionsfähigkeit des vorgemerkten gerichtlichen Erkenntnisses oder durch das rechtskräftige Erkenntnis der zuständigen Behörde, die über den Bestand des sichergestellten Anspruches zu entscheiden hat;

c) durch ein von dem zuständigen Gericht im Prozeßwege gefälltes Erkenntnis gegen die Person, wider die die Vormerkung erwirkt worden ist.

Anmerkungen

1. Zur lit. a: Aufsandungserklärung im Sinne des § 32 Abs. 1 lit. b.

2. Zur Rechtfertigung im Prozeßweg siehe auch § 439 ABGB, JGS Nr. 946/1811.

§ 42. (1) Muß die Rechtfertigung im Prozeßwege geschehen, so ist die Klage binnen 14 Tagen nach dem Tage der Zustellung des Vormerkungsbeschlusses von dem Vormerkungswerber bei dem zuständigen Gerichte zu erheben.

(2) In dem Rechtfertigungsprozeß hat der Kläger den Rechtsgrund zum Erwerb des angesprochenen bürgerlichen Rechtes, daher hinsichtlich eines vorgemerkten Pfandrechtes nicht nur die Richtigkeit der Forderung, sondern auch den Rechtsgrund zur Erwerbung des Pfandrechtes und dessen Umfang darzutun. Dem Beklagten steht frei, alle seine Einwendungen gegen den Bestand des bürgerlichen Rechtes selbst dann anzubringen, wenn er gegen den Beschluß, wodurch die Vormerkung bewilligt worden ist, den Rekurs nicht oder ohne Erfolg ergriffen haben sollte.

Anmerkungen

Zur Berechnung der Frist siehe § 81.

§ 43. (1) Die Frist zur Erhebung der Rechtfertigungsklage ist in dem Vormerkungsbeschluß auszudrücken. Sie kann aus erheblichen Gründen verlängert werden.

(2) Das Fristgesuch ist bei dem Grundbuchsgericht zu überreichen und nach der Zivilprozeßordnung zu behandeln.

Anmerkungen

Zu Abs. 2: § 128 ZPO, RGBI. Nr. 113/1895.

§ 44. Ist zur Zeit der Überreichung des Ansuchens um Vormerkung der Prozeß über den Bestand des vorgemerkten Rechtes schon anhängig, so bedarf es, solange nach den Bestimmungen der Zivilprozeßordnung das Begehren auch noch auf die Rechtfertigung der Vormerkung ausgedehnt werden darf, keiner besonderen Rechtfertigungsklage.

Anmerkungen

§§ 235 bis 482 ZPO, RGBI. Nr. 113/1895.

§ 45. (1) Unterbleibt die Rechtfertigung, so kann der, gegen den die Vormerkung bewilligt worden ist, um deren Löschung ansuchen.

(2) Liegt dem Grundbuchsgericht vor, daß die Rechtfertigungsklage rechtzeitig erhoben oder die Frist zur Rechtfertigung am Tage der Überreichung des Löschungsgesuches offengehalten ist, so hat es das Löschungsgesuch abzuweisen. Liegt dies nicht vor, so ist eine Tagsatzung auf kurze Zeit anzuordnen, bei der der Vormerkungswerber den Beweis, daß die Frist zur Rechtfertigung offengehalten oder die Klage rechtzeitig erhoben worden ist, zu liefern hat, widrigens die Löschung der Vormerkung zu bewilligen ist.

(3) Die Rechtfertigungsklage ist als rechtzeitig erhoben anzusehen, wenn sie, obgleich nach Ablauf der hierfür bestimmten Frist, doch noch vor Überreichung des Löschungsgesuches oder doch an dem nämlichen Tage erhoben worden ist.

§ 46. (1) Wird die Vormerkung für gerechtfertigt erkannt, so ist auf Ansuchen des Beteiligten die Rechtfertigung nach Maßgabe des rechtskräftigen Erkenntnisses im Grundbuch einzutragen.

(2) Wird dagegen die Vormerkung nicht für gerechtfertigt erkannt, so ist sie auf Ansuchen des Beteiligten auf Grund des rechtskräftigen Erkenntnisses zu löschen.

§ 47. Ist die Vormerkung deshalb gelöscht worden, weil dem Kläger das vorgemerkte Recht endgültig aberkannt oder die Vormerkung nicht für gerechtfertigt erklärt worden ist oder weil der, der sie erwirkt hat, unbedingt darauf verzichtet hat, so ist jede in der Folge auf Grund der nämlichen Urkunde abermals angesuchte Vormerkung desselben Rechtes entweder von Amts wegen abzuweisen oder, wenn dies unterblieben und abermals eine Vormerkung vorgenommen worden ist, diese Vormerkung wieder zu löschen, sobald der Gegner anzeigt, daß sie schon einmal gelöscht worden ist.

§ 48. (1) Ist dagegen die Vormerkung nur aus dem Grunde gelöscht worden, weil die Rechtfertigungsklage nicht in gehöriger Zeit angebracht worden ist, so kann zwar abermals eine Vormerkung angesucht werden; diese äußert jedoch ihre rechtliche Wirksamkeit erst von dem Zeitpunkte der Überreichung des neuen Gesuches.

(2) Auch steht dem Eigentümer der Liegenschaft oder des bürgerlichen Rechtes frei, auf Feststellung des Nichtbestehens des vorgemerkt gewesenen Rechtes zu klagen und im Fall eines günstigen Erkenntnisses durch dessen Anmerkung im Grundbuch einer wiederholten Bewilligung der Vormerkung vorzubeugen.

§ 49. (1) Wenn gegen den, der als Eigentümer einer Liegenschaft einverleibt ist, die Vormerkung des Eigentumsrechtes bewirkt worden ist, können sowohl gegen den einverlebten als gegen den vorgemerkten Eigentümer weitere Eintragungen zwar bewilligt werden, doch hängt deren rechtlicher Bestand davon ab, ob die Vormerkung des Eigentumsrechtes gerechtfertigt wird oder nicht.

(2) Wird die Vormerkung gerechtfertigt, so sind bei Eintragung der Rechtfertigung zugleich alle Eintragungen von Amts wegen zu löschen, die gegen den einverlebten Eigentümer nach dem Einlangen desjenigen Einschreitens erwirkt worden sind, auf das das Eigentumsrecht vorgemerkt worden ist.

(3) Wird dagegen die Vormerkung des Eigentumsrechtes gelöscht, so sind zugleich alle in bezug auf diese Vormerkung vorgenommenen Eintragungen von Amts wegen zu löschen.

(4) Diese Bestimmungen sind auch auf den Fall anzuwenden, daß gegen den Besitzer einer pfandrechtlich sichergestellten Forderung eine Vormerkung ihrer Übertragung auf eine andere Person bewirkt worden ist.

§ 50. (1) Ist die Löschung eines Rechtes nur vorgemerkt worden, so können in Hinsicht desselben zwar weitere Eintragungen, zum Beispiel von Afterpfandrechten oder Zessionen, bewilligt werden; doch hängt der rechtliche Bestand davon ab, ob die Vormerkung der Löschung gerechtfertigt wird oder nicht.

(2) Wird die Vormerkung gerechtfertigt, so sind bei der Eintragung der Rechtfertigung zugleich alle Eintragungen von Amts wegen zu löschen, die hinsichtlich des nunmehr gelöschten Rechtes mittlerweile bewilligt worden sind.

§ 51. (1) Wenn auf einer Hypothekarforderung zur Zeit, als ihre Löschung begehrt wird, noch Afterpfandrechte haften, darf die Löschung der Forderung nur mit dem Beisatz bewilligt werden, daß ihre Rechtswirkung in Ansehung der Afterpfandrechte erst mit deren Löschung einzutreten hat.

(2) Weitere Eintragungen auf diese Hypothekarforderung dürfen, wenn die Löschung einverleibt worden ist, nicht mehr bewilligt werden; ist die Löschung bloß vorgemerkt worden, so können diese nur mit der Rechtswirkung des § 50 erfolgen.

Anmerkungen

Bezüglich der Löschung des Baurechts siehe § 8 BauRG, RGBI. Nr. 86/1912.

VIERTER ABSCHNITT.

Von der Anmerkung.

1. Anmerkung persönlicher Verhältnisse.

§ 52. Die Anmerkung der im § 20 lit. a erwähnten Verhältnisse sowie die Löschung dieser Anmerkung erfolgt auf Ansuchen der Beteiligten, ihrer gesetzlichen Vertreter oder der hiezu berufenen Gerichte auf Grund beweiswirkender Urkunden.

2. Anmerkung der Rangordnung.

§ 53. (1) Der Eigentümer ist berechtigt, die bücherliche Anmerkung für eine beabsichtigte Veräußerung oder Verpfändung zu verlangen, um die bücherliche Rangordnung vom Zeitpunkte dieses Ansuchens für die infolge dieser Veräußerung oder Verpfändung einzutragenden Rechte zu begründen. Hiebei macht es keinen Unterschied, ob die Verpfändung für eine Schuld, deren Betrag anzugeben ist, oder für einen Höchstbetrag (§ 14 Abs. 2) erfolgt und ob die Urkunde, auf Grund deren die aus der Veräußerung oder Verpfändung sich ergebenden Rechte eingetragen werden sollen, vor oder nach dem Ansuchen um die Anmerkung errichtet worden ist. Auf Antrag ist in die Anmerkung der beabsichtigten Verpfändung die Bedingung aufzunehmen, daß die Eintragung eines Pfandrechtes im Range der Anmerkung nur für dieselbe Forderung zulässig ist, für die entweder im Zeitpunkt des Einlangens des Ansuchens um Eintragung des Pfandrechtes bereits im Range einer anderen Anmerkung der beabsichtigten Verpfändung, der eine Bedingung nicht beigesetzt ist, die Eintragung eines anderen Pfandrechtes bewilligt worden ist oder gleichzeitig mit der Bewilligung der Eintragung des Pfandrechtes im Range einer

anderen Anmerkung der beabsichtigten Verpfändung, der eine Bedingung nicht beigefügt ist, die Eintragung eines anderen Pfandrechtes bewilligt wird.

(2) Mit gleicher Rechtsfolge kann ein Hypothekargläubiger die Anmerkung der beabsichtigten Abtretung oder Löschung seiner Forderung verlangen.

(3) Die Anmerkungen solcher Gesuche können jedoch nur dann bewilligt werden, wenn nach dem Grundbuchsstand die Einverleibung des einzutragenden Rechtes oder die Löschung des bestehenden Rechtes zulässig wäre und wenn die Unterschrift der Gesuche gerichtlich oder notariell beglaubigt ist. Die Bestimmungen des § 31 Abs. 3 bis 5 sind anzuwenden.

Anmerkungen

Zu Abs. 1 dritter Satz: Der Zweck der schwer verständlichen Bestimmung liegt im Gerichtsgebührenrecht. In der Praxis wird in die Anmerkung der Beisatz "mit der Bedingung des § 53 Abs. 1 dritter Satz GBG" aufgenommen.

§ 54. Von dem Beschluß, mit dem das Gesuch bewilligt wird, darf nur eine Ausfertigung erteilt werden; diese ist mit der Bestätigung der vollzogenen Anmerkung zu versehen.

§ 55. Die Anmerkung der Rangordnung verliert ihre Wirksamkeit mit Ablauf eines Jahres nach ihrer Bewilligung. Dies ist in dem Beschluß unter Angabe des Kalendertages, an dem die Frist endet, auszusprechen.

Anmerkungen

Fristberechnung siehe § 81.

§ 56. (1) Das Gesuch um Eintragung des Rechtes oder der Löschung, für die die Rangordnung angemerkelt worden ist, ist unter Vorlage der Ausfertigung des die Anmerkung bewilligenden Beschlusses innerhalb der im § 55 festgesetzten Frist anzubringen. Wird über dieses Gesuch die Einverleibung oder Vormerkung bewilligt, so kommt der Eintragung die angemerkte Rangordnung zu. Die Eintragung ist auf der vorerwähnten Ausfertigung anzumerken.

(2) Die Eintragung in der angemerkten Rangordnung kann selbst dann bewilligt werden, wenn die Liegenschaft oder die Hypothekarforderung nach dem Einschreiten um die Anmerkung der Rangordnung an einen Dritten übertragen oder belastet worden wäre.

(3) Verfällt der Eigentümer der Liegenschaft oder der Hypothekargläubiger vor der Überreichung des Eintragungsgesuches in Konkurs, so kann die Eintragung nur dann bewilligt werden, wenn die Urkunde über das Geschäft schon vor dem Tage der Konkurseröffnung ausgefertigt war und der Tag ihrer Ausfertigung durch eine gerichtliche oder notarielle Beglaubigung dargetan ist. Entspricht die Urkunde diesen Voraussetzungen nicht, so ist die Zulässigkeit der Eintragung nach den Vorschriften der Konkursordnung zu beurteilen.

Anmerkungen

Zu Abs. 1: Die Eintragung im Rang der Anmerkung muß ausdrücklich beantragt werden (§§ 85 und 96).

§ 57. (1) Wird die Einverleibung der Veräußerung der Liegenschaft oder der Zession oder Löschung der Forderung in der angemerkten Rangordnung bewilligt, so ist auf Ansuchen der Partei, für die die Einverleibung vorgenommen worden ist, die Löschung der Eintragungen zu verfügen, die etwa in Ansehung dieser Liegenschaft oder Forderung nach Überreichung des Anmerkungsgesuches erwirkt worden sind. Um die Löschung dieser Eintragungen muß jedoch

binnen 14 Tagen nach Rechtskraft der in der angemerkten Rangordnung bewilligten Einverleibung angesucht werden.

(2) Wird das Eintragungsgesuch nicht vor dem Ende der festgesetzten Frist angebracht oder wird der Betrag, für den die Anmerkung der Rangordnung erfolgt ist, bis zum Ende dieser Frist nicht erschöpft, so wird die Anmerkung unwirksam und ist von Amts wegen zu löschen.

(3) Vor Ablauf der gesetzlichen Frist kann die Löschung der Anmerkung nur dann bewilligt werden, wenn die Ausfertigung des Beschlusses über die Bewilligung der Anmerkung vorgelegt wird. Die Löschung ist auf dieser Ausfertigung anzumerken.

3. Rangvorbehalt.

§ 58. (1) Im Falle der Löschung des Pfandrechtes kann der Eigentümer zugleich die Anmerkung im Grundbuch erwirken, daß die Eintragung eines neuen Pfandrechtes im Rang und bis zur Höhe des gelöschten Pfandrechtes binnen drei Jahren nach der Bewilligung der Anmerkung vorbehalten bleibt. Dieser Vorbehalt ist im Fall eines Eigentumswechsels zugunsten des neuen Eigentümers wirksam.

(2) Diese Bestimmung ist sinngemäß anzuwenden, wenn die neue Forderung an die Stelle zweier oder mehrerer im Rang unmittelbar aufeinanderfolgender Hypothekarforderungen treten soll.

Anmerkungen

Es handelt sich um eine Form der Geltendmachung des Verfügungsrechts nach § 469 ABGB, JGS Nr. 946/1811.

4. Bedingte Pfandrechtseintragung.

§ 59. (1) Der Eigentümer einer Liegenschaft kann begehren, daß im Rang und bis zur Höhe eines auf der Liegenschaft haftenden Pfandrechtes das Pfandrecht für eine neue Forderung mit der Beschränkung eingetragen werde, daß es Rechtswirksamkeit erlangt, wenn binnen einem Jahr nach der Bewilligung der Eintragung des neuen Pfandrechtes die Löschung des älteren Pfandrechtes einverleibt wird.

(2) Der Eintritt dieser Bedingung ist im öffentlichen Buch zugleich mit der Löschung des älteren Pfandrechtes anzumerken.

(3) Wenn innerhalb der Frist nicht um die Löschung angesucht oder wenn die Löschung nicht bewilligt wird, erlischt das neue Pfandrecht mit dem Ablauf der Frist und ist samt allen darauf bezüglichen Eintragungen von Amts wegen zu löschen.

(4) Zur Anbringung des Gesuches um Löschung des älteren Pfandrechtes ist außer dem Hypothekarschuldner auch der Gläubiger berechtigt, zu dessen Gunsten das Pfandrecht für die neue Forderung eingetragen ist.

(5) Ist das ältere Pfandrecht belastet, so wird das in dessen Rang eingetragene neue Pfandrecht nur unter der weiteren Bedingung (Abs. 1) rechtswirksam, daß die Belastung gelöscht oder mit Zustimmung der Beteiligten auf das eingetragene neue Pfandrecht übertragen wird.

(6) Haftet das ältere Pfandrecht simultan auf mehreren Grundbuchsobjekten, so wird das neue Pfandrecht nur unter der weiteren Bedingung (Abs. 1) rechtswirksam, daß das ältere Pfandrecht auf sämtlichen Grundbuchsobjekten gelöscht wird.

(7) Die Bestimmungen der Abs. 1 bis 6 sind sinngemäß anzuwenden, wenn die neue Forderung an die Stelle zweier oder mehrerer im Rang unmittelbar aufeinanderfolgender Hypothekarforderungen treten soll.

Anmerkungen

1. Es handelt sich um eine weitere Form der Geltendmachung des Verfügungsrechts nach § 469 ABGB, JGS Nr. 946/1811.

2. Fristberechnung siehe § 81.

5. Anmerkung der Aufkündigung und der Hypothekarklage.

§ 60. (1) Die Anmerkung einer gerichtlich oder notariell beurkundeten Aufkündigung einer Hypothekarforderung sowie die Anmerkung einer Hypothekarklage ist auf Begehren des Gläubigers vom Grundbuchsgericht zu bewilligen, wenn der, gegen den die Aufkündigung oder die Klage gerichtet ist, als Eigentümer der verpfändeten Liegenschaft eingetragen und die Anhängigkeit der Hypothekarklage ausgewiesen ist.

(2) Die Anmerkung der Hypothekarklage kann auch vom Prozeßgericht sofort bewilligt werden.

(3) Eine solche Anmerkung hat zur Folge, daß die Aufkündigung oder die Klage ihre Wirksamkeit auch gegen jeden späteren Eigentümer des Pfandes äußert und daß insbesondere die Exekution auf die verpfändete Liegenschaft auf Grund des über die angemerkte Klage erfolgten rechtskräftigen Urteiles oder exekutionsfähigen Vergleiches unmittelbar gegen jeden Eigentümer dieser Liegenschaft geführt werden kann.

6. Löschungsklagen und Streitankerkungen.

§ 61. (1) Wenn jemand, der durch eine Einverleibung in seinem bürgerlichen Rechte verletzt erscheint, die Einverleibung aus dem Grunde der Ungültigkeit im Prozeßwege bestreitet und die Wiederherstellung des vorigen bürgerlichen Standes begehrt, kann er die Anmerkung eines solchen Streites im Grundbuch entweder gleichzeitig mit der Klage oder später verlangen. Die Anmerkung des Streites kann sowohl bei dem Prozeßgericht als auch bei dem Grundbuchsgericht angesucht werden.

(2) Diese Streitankerkung hat zur Folge, daß das über die Klage ergehende Urteil auch gegen die Personen, die erst nach dem Zeitpunkt, in dem das Gesuch um die Streitankerkung an das Grundbuchsgericht gelangt ist, bürgerliche Rechte erlangt haben, seine volle Wirksamkeit äußert.

§ 62. Wenn die Löschungsklage gegen die Personen gerichtet werden soll, die unmittelbar durch die Einverleibung, auf deren Löschung geklagt wird, Rechte erworben haben oder von einer Last befreit worden sind, oder wenn sich die Klage auf solche Verhältnisse stützt, die unmittelbar zwischen dem Kläger und Beklagten obwalten, ist die Dauer des Klagerechtes nach den zivilrechtlichen Bestimmungen über die Verjährung zu beurteilen.

Anmerkungen

Rekursfrist siehe § 123 Abs. 1.

§ 63. (1) Wer jedoch eine Einverleibung, von deren Bewilligung er vorschriftsmäßig verständigt worden ist, auch gegen dritte Personen als ungültig bestreiten will, hat binnen der Frist, die ihm

zum Rekurs gegen deren Bewilligung zukäme, bei dem Grundbuchsgericht die Anmerkung, daß diese Einverleibung streitig sei, anzusehen und entweder zugleich oder längstens binnen weiteren sechzig Tagen nach Ablauf der Rekursfrist die Klage auf Löschung gegen alle Personen zu überreichen, die durch die bestrittene Einverleibung ein bürgerliches Recht erlangt oder weitere Einverleibungen oder Vormerkungen darauf erwirkt haben.

(2) Nach Ablauf dieser Fristen kann auf Löschung der bestrittenen Einverleibung gegen dritte Personen, die noch vor der Streitankündigung weitere bürgerliche Rechte darauf erwirkt haben, nur dann erkannt werden, wenn sie sich hinsichtlich ihrer Gültigkeit nicht im guten Glauben befunden haben.

Anmerkungen

Rekursfrist siehe § 123 Abs. 1.

§ 64. Sollte aber die vorschriftsmäßige Verständigung des Klägers von der Bewilligung einer Einverleibung, deren Ungültigkeit er behauptet, aus was immer für einem Grund unterblieben sein, so erlischt das Klagerecht auf deren Löschung gegen dritte Personen, die weitere bürgerliche Rechte darauf in gutem Glauben erworben haben, erst binnen drei Jahren von dem Zeitpunkt an, in dem die angefochtene Einverleibung bei dem Grundbuchsgericht angesucht worden ist.

§ 65. (1) Steht der Kläger von der Klage ab oder wird er durch rechtskräftiges Erkenntnis abgewiesen oder hat er die Klage im Falle des § 63 in der vorgeschriebenen Frist nicht überreicht, so ist auf Ansuchen des Gegners die Löschung der Streitankündigung zu verfügen.

(2) Wird aber durch ein rechtskräftiges Urteil oder einen Vergleich die bestrittene Einverleibung ganz oder teilweise aufgehoben, so ist auf Ansuchen des Klägers die Vornahme der Löschung der bestrittenen Einverleibung in der in dem Urteil oder Vergleich ausgedrückten Art und Ausdehnung zu bewilligen und zugleich sowohl die Löschung der Streitankündigung als aller der Einverleibungen und Vormerkungen anzuordnen, die hinsichtlich des zu löschenden Rechtes erst nach dem Zeitpunkt, in dem das Gesuch um die Streitankündigung an das Grundbuchsgericht gelangt ist, angesucht worden sind.

§ 66. (1) Wer behauptet, daß eine Einverleibung in Folge einer strafgesetzlich verbotenen Handlung erwirkt worden sei, kann, um die im § 61 bezeichnete Rechtswirkung gegen spätere Eintragungen zu begründen, bei dem Grundbuchsgericht unter Beibringung der Bestätigung der zuständigen Behörde, daß die Strafanzeige bei ihr erstattet worden ist, die Anmerkung ansuchen, daß die Einverleibung streitig sei.

(2) Soll jedoch durch die Streitankündigung die Wirkung begründet werden, daß der Anspruch auf die Ungültigerklärung einer Einverleibung auch gegen dritte Personen, die bürgerliche Rechte noch vor der Streitankündigung im guten Glauben darauf erworben haben, gewahrt werde, so muß das Gesuch um die Streitankündigung bei dem Grundbuchsgericht innerhalb der Frist eingebracht werden, die der Partei zum Rekurs gegen die Bewilligung der Einverleibung zukäme.

§ 67. Erklärt das Strafgericht, daß die Einverleibung samt den bürgerlichen Rechten, die etwa vor der im § 66 bezeichneten Anmerkung erworben worden sind, zu löschen sei, so hat das Grundbuchsgericht, wenn von der verletzten Partei das Erkenntnis hierüber mit der Bestätigung seiner Rechtskraft beigebracht wird, diese Löschung nach den Bestimmungen des § 65 in Voll-

zug setzen zu lassen. Hat das Strafgericht dagegen zwar auf die Schuld des Angeklagten, jedoch nicht auf eine solche Löschung erkannt und die geschädigte Partei hinsichtlich der angesprochenen Löschung der Einverleibung auf den Zivilrechtsweg gewiesen, so steht der Partei für die Klage auf Löschung der Einverleibung und der oben bezeichneten bürgerlichen Rechte eine Frist von sechzig Tagen nach Eintritt der Rechtskraft dieser Entscheidung zu. Nach dem fruchtlosen Ablauf dieser Frist sowie wenn das Strafgericht auf die Schuld des Angeklagten nicht erkannt hat, ist die Löschung der Streitanmerkung auf Ansuchen dessen, der an der Aufrechterhaltung der Einverleibung ein Interesse hat, zu bewilligen.

Anmerkungen

Fristberechnung siehe § 81.

§ 68. Wird die Löschung einer Streitanmerkung aus dem Grunde begehrt, weil die Klage auf Löschung nicht innerhalb der in den §§ 63, 67 bestimmten Fristen erhoben worden ist, so hat das Grundbuchgericht, falls ihm nicht das Gegenteil bekannt ist, eine Tagsatzung auf kurze Zeit anzuordnen, bei welcher der, der die Streitanmerkung erwirkt hat, den Beweis zu liefern hat, daß die Klage rechtzeitig erhoben worden ist, widrigens die Löschung der Anmerkung zu bewilligen ist.

§ 69. Wenn ein bürgerlicher Eigentümer oder Gläubiger, auf dessen Gut oder Forderung ein Recht einverleibt ist, aus dem Grunde der Verjährung auf gänzliche oder teilweise Löschung klagt, kann die Anmerkung des Streites bewilligt werden.

Anmerkungen

Zur Löschung verjährter Rechte siehe auch § 131 Abs. 2 lit. b.

§ 70. Die Anmerkung des Streites kann auch dem bewilligt werden, der aus dem Grund der Ersitzung (§ 1498 ABGB.) die Zuerkennung eines dinglichen Rechtes begehrt.

§ 71. (1) Die Streitanmerkung einer Löschungsklage wegen Verjährung (§ 69) oder einer Klage auf Zuerkennung eines dinglichen Rechtes in Folge der Ersitzung (§ 70) hat jedoch gegen dritte Personen keine Wirkung, die im Vertrauen auf das Grundbuch bürgerliche Einverleibungen oder Vormerkungen vor dem Zeitpunkt erwirkt haben, in dem das Gesuch um die Streitanmerkung an das Grundbuchgericht gelangt ist. Das zuerkannte ersessene Recht genießt die Rangordnung vor allen Eintragungen, die erst nach der Streitanmerkung vorgenommen worden sind; alle damit im Widerspruch stehenden, nach der Streitanmerkung eingetragenen Rechte sind zu löschen.

(2) Im übrigen ist nach den Bestimmungen des § 65 vorzugehen.

7. Anmerkung der Erteilung des Zuschlages.

§ 72. (1) Jenes Gericht, bei dem die Zwangsversteigerung einer Liegenschaft vollzogen worden ist, hat die Anmerkung dieses Vollzuges von Amts wegen im Grundbuch zu verfügen.

(2) Diese Anmerkung hat die Folge, daß weitere Eintragungen gegen den bisherigen Eigentümer nur für den Fall ein Recht bewirken, als die Versteigerung für unwirksam erklärt wird.

(3) Ist eine Anfechtung der Versteigerung entweder nicht erfolgt oder endgültig abgewiesen worden, so findet auf Ansuchen der Beteiligten die Löschung aller nach der Anmerkung der Erteilung des Zuschlages gegen den bisherigen Eigentümer erwirkten Eintragungen und der etwa in bezug auf diese weiter vorgenommenen Eintragungen statt.

§ 73. Inwieweit das Grundbuchgericht oder ein anderes Gericht in anderen Fällen eine Anmerkung anzuordnen hat, wird teils in diesem Bundesgesetz teils im Liegenschaftsteilungsgesetz, BGBl. Nr. 3/1930, teils in anderen Gesetzen bestimmt.

FÜNFTER ABSCHNITT.

Von der Abschreibung von Bestandteilen eines Grundbuchkörpers.

§ 74. (1) Die Abschreibung des Bestandteiles eines Grundbuchkörpers und seine Zuschreibung zu einem anderen Grundbuchkörper oder die Eröffnung einer neuen Einlage für diesen Bestandteil ist nur dann zulässig, wenn der abzuschreibende Teil genau, nötigenfalls durch Pläne, von denen eine Kopie in der Urkundensammlung aufzubewahren ist, bezeichnet ist und wenn die das Begehren begründenden Urkunden den zu einer Einverleibung des Eigentumsrechtes vorgeschriebenen Erfordernissen entsprechen.

(2) Bei der Durchführung der Abschreibung ist nach den Bestimmungen des Liegenschaftsteilungsgesetzes, BGBl. Nr. 3/1930, vorzugehen.

DRITTES HAUPTSTÜCK.

Von dem Verfahren in Grundbuchssachen.

ERSTER ABSCHNITT.

Allgemeine Bestimmungen.

1. Zuständigkeit.

§ 75. Die Bewilligung einer Eintragung ist mit Ausnahme der in diesem Bundesgesetz sowie in den Gesetzen über das gerichtliche Verfahren bestimmten Fällen bei dem Grundbuchgericht anzusuchen, bei dem sich die Einlage, in der die Eintragung erfolgen soll, befindet.

Anmerkungen

1. Zur Zuständigkeit als Grundbuchgericht siehe § 118 JN, RGBl. Nr. 111/1895.
2. Weitere Bestimmungen über die Zuständigkeit enthalten ua.
 - a) §§ 60 und 61 über die Bewilligung von Anmerkungen durch das Prozeßgericht;
 - b) §§ 108 und 111 für Simultanhypotheken;
 - c) §§ 23 und 24 LiegTeilG, BGBl. Nr. 3/1930, für Ab- und Zuschreibungen;
 - d) § 177 AußStrG, RGBl. Nr. 208/1854, für die Verbücherung der Einantwortungsurkunde;
 - e) die EO, RGBl. Nr. 79/1896 für das Exekutionsverfahren.

2. Grundsatz des Verfahrens.

§ 76. Das Grundbuchgericht ordnet, außer den in diesem oder in einem anderen Gesetz bestimmten Fällen, Eintragungen nicht von Amts wegen, sondern nur auf Ansuchen von Parteien oder Behörden an.

3. Berechtigung zum Ansuchen.

§ 77. (1) Wenn jemand im Namen eines anderen einschreitet, muß dargetan sein, daß er zur Anbringung von Grundbuchsgesuchen befugt sei.

(2) Zum Ansuchen um eine Eintragung im Namen dessen, dem sie zum Vorteil gereicht, genügt eine allgemeine Vollmacht.

(3) Gesetzliche oder gerichtlich bestellte Vertreter bedürfen keiner besonderen Ermächtigung, um die Eintragung von Rechten der ihrer Vertretung zugewiesenen Personen oder die Löschung von Lasten des ihrer Verwaltung anvertrauten Vermögens zu bewirken.

Anmerkungen

Die Antragslegitimation selbst ist im GBG nicht geregelt; sie ist daher nach dem AußStrG, RGBI. Nr. 208/1854 zu beurteilen.

§ 78. Wenn der, an den eine Liegenschaft oder ein bürgerliches Recht außerbücherlich gelangt ist, darauf ein Recht, das Gegenstand der öffentlichen Bücher ist, einem anderen eingeräumt hat, kann letzterer die Eintragung der Rechte seines Vormannes verlangen.

§ 79. Auch der Bürge kann, wenn der Gläubiger das ihm eingeräumte Recht zur Erlangung des Pfandrechtes an der Liegenschaft oder dem bürgerlichen Recht seines Schuldners nicht ausübt, im Namen des Gläubigers die Eintragung begehren.

§ 80. Um die Eintragung gemeinschaftlicher Rechte, die eine Teilung im Verhältnis zum Ganzen nicht zulassen, kann jeder Teilhaber für sich und im Namen der übrigen Teilhaber ansuchen.

4. Fristen.

§ 81. (1) Die nicht auf einen Kalendertag festgesetzten Fristen beginnen mit dem Tag nach der Zustellung.

(2) Bei ihrer Berechnung dürfen Sonn- oder Feiertage sowie die Tage, während deren sich eine bei dem Grundbuchsgericht zu überreichende Schrift auf der Post befindet, nicht abgerechnet werden.

(3) Diese Fristen lassen, mit Ausnahme der Frist zur Rechtfertigung einer Vormerkung (§ 43) und der Frist zur Beibringung der Originalurkunde (§ 88) oder der Übersetzung (§ 89), keine Erstreckung zu.

§ 82. Wegen der Versäumung der in diesem Bundesgesetz bestimmten Fristen ist keine Wiedereinsetzung in den vorigen Stand zulässig.

ZWEITER ABSCHNITT.

Von den Gesuchen.

1. Form des Ansuchens.

§ 83. (1) Grundbuchsgesuche können sowohl schriftlich als auch mündlich angebracht werden.

(2) Wird das Gesuch mündlich angebracht, so ist darüber unter Beachtung der für den Inhalt der schriftlichen Gesuche gegebenen Vorschriften ein Protokoll aufzunehmen und der Antragsteller zu einem bestimmten Begehren anzuleiten.

2. Erfordernisse.

§ 84. In jedem Grundbuchsgesuch sind das Grundbuchsgericht, bei dem es zu überreichen ist, sowie der Vor- und Zuname, der Stand und Wohnort des Antragstellers und der Personen, die von der Erledigung zu verständigen sind, und wenn sie juristische Personen (Körperschaften usw.) sind, die ihnen zukommenden Benennungen anzugeben.

Anmerkungen

Grundbuchsansträge sind nach § 58 Geo, BGBl. Nr. 264/1951, von außen als Grundbuchsstücke zu bezeichnen.

§ 85. (1) Die Grundbuchseinlagen, in denen eine Eintragung geschehen soll, sind mit der nämlichen Bezeichnung anzuführen, unter der sie im Grundbuch erscheinen.

(2) Im Begehren ist genau anzugeben, was im Grundbuch eingetragen werden soll.

(3) Das Begehren um Einverleibung begreift jenes um Vormerkung stillschweigend in sich, wenn der Antragsteller die Vormerkung nicht ausdrücklich ausgeschlossen hat. Kann oder will der Antragsteller nur auf die Früchte der Liegenschaft ein dingliches Recht erwerben, so hat er dies ausdrücklich zu bemerken.

3. Kumulierung der Gesuche.

§ 86. Mehrere Eintragungen, die durch dieselbe Urkunde begründet werden, sowie die Eintragung eines Rechtes in mehreren Grundbuchseinlagen oder die Eintragung mehrerer Rechte in einer Grundbuchseinlage können mit einem einzigen Gesuch begehrt werden.

Anmerkungen

1. Voraussetzung ist grundsätzlich, daß ein Grundbuchsgericht zuständig ist (Ausnahmen: § 108 sowie § 23 LiegTeilG, BGBl. Nr. 3/1930).

2. Die Verbindung eines Antrags mit Anmerkung der Rangordnung mit anderen Eintragungen ist mit Rücksicht auf § 54 unzulässig.

4. Beilagen.

a) *Originale.*

§ 87. (1) Die Urkunden, auf Grund deren eine Eintragung erfolgen soll, sind im Original beizulegen.

(2) Liegt die Originalurkunde bei dem Grundbuchsgericht entweder in den Akten oder in Aufbewahrung oder ist sie einem im Zug befindlichen Gesuch angeschlossen, so genügt es, eine Abschrift beizubringen und anzugeben, wo sich das Original befindet.

Anmerkungen

1. Bei einer gerichtlichen Entscheidung oder einem Notariatsakt ist die Ausfertigung das Original.
2. § 87 gilt nicht für Urkunden, die nur Voraussetzung der Bewilligung sind wie die Einschreitervollmacht.
3. Siehe auch § 56 Abs. 1.

§ 88. (1) Kann das Original nicht sogleich beigebracht werden, weil es sich bei einer anderen Behörde befindet, so ist dies in dem Gesuch anzugeben und eine beglaubigte Abschrift beizulegen.

(2) Könnte das Gesuch, selbst wenn die Originalurkunde vorläge, nicht bewilligt werden, so ist es sogleich abzuweisen.

(3) Könnte aber unter jener Voraussetzung dem Gesuch stattgegeben werden, so ist dieses zur Wahrung der Rangordnung des Rechtes sogleich mit dem Beisatze „Bis zum Einlangen des Originals“ im Grundbuch anzumerken.

(4) Dem Antragsteller ist zugleich, wenn die Originalurkunde nicht schon von Amts wegen von einem Grundbuchsgericht, bei dem sie sich befindet, einzusenden ist, eine angemessene Frist zu ihrer Beibringung zu bestimmen; wird die Originalurkunde von dem Grundbuchsgericht eingesendet oder in der gegebenen Frist überreicht, so ist das Gesuch in der Sache selbst zu erledigen.

(5) Wird die Originalurkunde in der gegebenen oder erweiterten Frist nicht überreicht, so ist das Gesuch sofort abzuweisen und die Anmerkung von Amts wegen zu löschen.

b) Übersetzungen.

§ 89. (1) Sind die Urkunden nicht in einer Sprache verfaßt, in der Eingaben bei dem Grundbuchsgericht überreicht werden können, so muß eine vollen Glauben verdienende Übersetzung beigebracht werden.

(2) Fehlt die Übersetzung und geht nicht aus dem Gesuch hervor, daß es jedenfalls abzuweisen ist, so ist das Gesuch zur Wahrung der Rangordnung des Rechtes mit dem Beisatz "Bis zum Einlangen der Übersetzung" im Grundbuch anzumerken. Zugleich ist dem Antragsteller eine angemessene Frist zur Vorlage der Übersetzung zu bestimmen. Wird die Übersetzung in der gegebenen oder erweiterten Frist eingereicht, so ist das Gesuch in der Sache selbst zu erledigen; im entgegengesetzten Fall ist es abzuweisen und die Anmerkung von Amts wegen zu löschen.

c) Abschriften.

§ 90. Sofern für die Urkundensammlungen Abschriften erforderlich sind (§ 6), sind sie von den Stempel- und Gerichtsgebühren befreit. Werden sie nicht beigebracht oder sind sie nicht brauchbar, so sind die Originale in der Urkundensammlung aufzubewahren und die Parteien von dem Grundbuchsführer zu verständigen, daß es ihnen freistehe, sie gegen nachträgliche Beibringung ordnungsmäßiger Abschriften zu beheben; das Gericht kann für die Urkundensammlung auch Abschriften gegen Einhebung der einfachen für gerichtliche Abschriften bestimmten Gebühr anfertigen und die Originalurkunden bei den Akten zur Ausfolgung bereithalten. Wenn aber das Gesuch, in dem eine Eintragung bei mehreren Grundbuchsgerichten beantragt wird, nebst der Originalurkunde von einem Grundbuchsgericht zum anderen gelangen soll, hat jedes Grundbuchsgericht, wenn die für sein Grundbuch erforderlichen Abschriften

nicht beiliegen oder unbrauchbar sind, solche gegen Einhebung der doppelten für beglaubigte Abschriften bestimmten Gerichtsgebühr auszufertigen.

Anmerkungen

Nähere Bestimmungen enthält § 577 Geo, BGBl. Nr. 264/1951.

§ 91. Der Grundbuchsführer hat auf den eingelegten Abschriften die Übereinstimmung mit den Originalurkunden von Amts wegen zu bestätigen.

5. Ausfertigungen des Gesuches und Halbschriften.

§ 92. (1) Grundbuchsgesuche sind, sofern nicht eine Ausnahme gesetzlich festgesetzt ist, in einer Ausfertigung zu überreichen.

(2) Den Gesuchen sind so viele Halbschriften beizulegen, als Verständigungen von der Gesuchserledigung stattzufinden haben. Der Mangel dieser Halbschriften bildet jedoch keinen Grund zur Abweisung des Gesuches.

(3) Auf den Halbschriften ist das in dem Gesuch gestellte Begehren in den wesentlichen Punkten anzugeben.

(4) Statt der Halbschriften können vollständige Abschriften des Gesuches beigelegt werden. In diesem Fall ist anzugeben, wem sie zuzustellen sind.

(5) Ist das Gesuch zu Protokoll genommen worden, so hat das Gericht die erforderlichen Halbschriften und auf Ansuchen vollständige Protokollsabschriften zur Verständigung der Beteiligten anzufertigen.

Anmerkungen

Zu Abs. 1: Eine Ausnahme enthält z. B. § 24 LiegTeilG, BGBl. Nr. 3/1930.

DRITTER ABSCHNITT.

Von der Erledigung der Gesuche.

1. Prüfung und Entscheidung.

§ 93. Der Zeitpunkt, in dem ein Ansuchen bei dem Grundbuchsgericht einlangt, ist für die Beurteilung dieses Ansuchens entscheidend.

Anmerkungen

Nach diesem Zeitpunkt richtet sich auch der Rang (§ 29).

§ 94. (1) Das Grundbuchsgericht hat das Ansuchen und dessen Beilagen einer genauen Prüfung zu unterziehen und darf eine grundbücherliche Eintragung nur dann bewilligen, wenn

1. aus dem Grundbuch in Ansehung der Liegenschaft oder des Rechtes kein Hindernis gegen die begehrte Eintragung hervorgeht;

2. kein begründetes Bedenken gegen die persönliche Fähigkeit der bei der Eintragung Beteiligten zur Verfügung über den Gegenstand, den die Eintragung betrifft, oder gegen die Befugnis der Antragsteller zum Einschreiten vorhanden ist;

3. das Begehren durch den Inhalt der beigebrachten Urkunden begründet erscheint und

4. die Urkunden in der Form vorliegen, die zur Bewilligung einer Einverleibung, Vormerkung oder Anmerkung erforderlich ist.

(2) Bei grundbücherlichen Eintragungen, die nicht von dem Grundbuchsgericht, sondern von einem anderen Gericht bewilligt werden, hat sich das Grundbuchsgericht darauf zu beschränken, über die Zulässigkeit der Eintragung mit Rücksicht auf den Grundbuchsstand zu entscheiden; hinsichtlich der übrigen Erfordernisse steht die Entscheidung dem bewilligenden Gericht zu.

Anmerkungen

Zur Einschreiterbefugnis (Abs. 1 Z 2) siehe § 77, zu Abs. 1 Z 3 und 4 siehe insbesondere §§ 26, 27 und 31 bis 34.

§ 95. (1) Über jedes Grundbuchsgesuch hat das Grundbuchsgericht, mit Ausnahme der in den §§ 45, 68 und 104 dieses Bundesgesetzes sowie im Liegenschaftsteilungsgesetz, BGBl. Nr. 3/1930, festgesetzten Fälle, ohne Einvernehmung der Parteien und in der Regel (§§ 88 und 89) ohne Zwischenerledigung in der Sache zu entscheiden und in dem zu erlassenden Beschluß die Bewilligung oder Abweisung des Gesuches ausdrücklich auszusprechen.

(2) Kann dem Begehren zwar nicht im vollen Umfang, aber doch zum Teil stattgegeben werden, so ist die Eintragung, soweit sie zulässig ist, anzuordnen und der Teil des Begehrens, dem nicht entsprochen werden kann, abzuweisen.

(3) Wird das Gesuch ganz oder teilweise abgewiesen, so sind in dem Beschluß alle Gründe anzugeben, die der Bewilligung entgegenstehen.

Anmerkungen

Zum Abs. 1: Weitere Ausnahmen: § 14 Abs. 4, § 33 Abs. 2 und § 133 Abs. 1 lit. b.

2. Besondere Bestimmungen in Ansehung

a) der Bewilligung:

§ 96. (1) Mehr oder etwas anderes, als die Partei angesucht hat, darf nicht bewilligt werden, auch wenn sie nach den beigebrachten Urkunden zu einem ausgedehnteren oder anderen Begehren berechtigt wäre.

(2) Ist nur die Vormerkung angesucht worden, so darf die Einverleibung nicht angeordnet werden, wenn sie auch zulässig wäre (§ 85).

§ 97. (1) Wenn aus einer Urkunde hervorgeht, daß dem Erwerber eines dinglichen Rechtes die Bewilligung zur Einverleibung erteilt worden ist, daß ihm aber zugleich Beschränkungen in der Verfügung über das erworbene Recht oder Gegenverpflichtungen auferlegt worden sind, hinsichtlich deren die gleichzeitige Einverleibung für die daraus Berechtigten bedungen worden ist, darf die Eintragung jenes Rechtes nicht bewilligt werden, wenn nicht zugleich hinsichtlich der bedungenen Beschränkungen oder Gegenverpflichtungen die Einverleibung oder nach der Beschaffenheit der Urkunde doch die Vormerkung angesucht wird.

(2) Das Gesuch um die gleichzeitige Eintragung der gegenseitigen Rechte kann sowohl von dem einen als von dem anderen Teil angebracht werden.

§ 98. In den Beschlüssen, womit eine Eintragung bewilligt wird, sind die Grundbuchseinlagen zu bezeichnen, in denen die Eintragung erfolgen soll; ferner sind unter Beziehung auf die der Bewilligung zugrunde liegenden Urkunden die Personen, für die, und die Objekte, auf die die Eintragung erfolgen soll, endlich die einzutragenden Rechte nebst den wesentlichen Bestimmungen mit den in das Hauptbuch einzutragenden Worten anzuführen (§ 5). Bei natürlichen Personen ist auch das Geburtsdatum anzuführen.

b) der Abweisung:

§ 99. (1) Wird ein Einverleibungs- oder Vormerkungsgesuch oder ein Gesuch um Abschreibung von Grundstücken oder um Anmerkung der Rangordnung abgewiesen, so ist das abgewiesene Gesuch im Grundbuch anzumerken. Das gleiche gilt, wenn ein Antrag auf Bewilligung der Zwangsversteigerung oder der Zwangsverwaltung zur Hereinbringung einer Forderung abgewiesen wird, für die kein Pfandrecht einverleibt ist.

(2) Diese Anmerkung findet nicht statt, wenn das Gut oder das Recht, auf das die Eintragung begehrt wird,

a) weder aus dem Gesuch noch aus dessen Beilagen ersichtlich oder in den Büchern des Grundbuchsgerichtes nicht eingetragen ist oder

b) für eine andere Person eingetragen ist, als die, gegen die nach Inhalt der Urkunde eine Einverleibung oder Vormerkung stattfinden kann.

§ 100. Ist die Abweisung eines der im § 99 angeführten Gesuche von einem anderen Gericht als von dem Grundbuchsgericht ausgegangen, so ist dieses von Amts wegen um die Anmerkung der Abweisung zu ersuchen.

Anmerkungen

Siehe auch § 129 Abs. 2.

§ 101. Sobald das Grundbuchsgericht Kenntnis erlangt, daß ein Beschluß, wodurch eines der im § 99 angeführten Gesuche abgewiesen worden ist, durch Unterlassung des Rekurses rechtskräftig geworden ist, hat es die Anmerkung des abgewiesenen Gesuches von Amts wegen zu löschen und die Beteiligten hievon zu verständigen.

VIERTER ABSCHNITT.

Von dem Vollzug der Eintragungen.

§ 102. (1) Eine Eintragung in das Grundbuch darf nur auf Grund eines schriftlichen Auftrages des Grundbuchsgerichtes und nicht anders als nach dem Inhalt dieses Auftrages vorgenommen werden.

(2) Wenn sich der Vollzug eines Auftrages nach dem Grundbuchsstand als unausführbar herausstellt, kann der erteilte Auftrag nur durch einen neuen Auftrag des Grundbuchsgerichtes berichtigt werden.

Anmerkungen

1. Mit dem "schriftlichen Auftrag" ist der Grundbuchsbeschluß gemeint.

2. Abweichende Bestimmungen enthalten die §§ 13 bis 15 GUG, BGBl. Nr. 550/1980.

§ 103. (1) Jede Eintragung (§ 8) hat nebst der Bezeichnung ihrer Art die Angabe des Tages, Monates, Jahres und der Einreichungszahl zu enthalten, unter denen das Ansuchen an das Grundbuchsgericht gelangt ist.

(2) Sind bei dem Grundbuchsgericht mehrere denselben Grundbuchkörper betreffende Ansuchen gleichzeitig eingelangt, so sind bei jeder Eintragung auf Grund dieser Ansuchen die Einreichungszahlen der gleichzeitig eingelangten Ansuchen mit einem ihre Gleichzeitigkeit ausdrückenden Beisatz anzumerken.

Anmerkungen

1. Die Einreichungszahl wird üblicherweise als Tagebuchzahl oder TZ bezeichnet.
2. Abweichende Bestimmungen enthält § 12 GUG, BGBl. Nr. 550/1980.

§ 104. (1) Im Grundbuch darf nichts radiert und das Eingetragene auch nicht in anderer Weise unleserlich gemacht werden.

(2) Wird ein Fehler bei der Eintragung begangen und noch während der Eintragung bemerkt, so ist er ohne Einholung eines Auftrages des Grundbuchsgerichtes zu berichtigen.

(3) Ein nach vollendeter Eintragung wahrgenommener Fehler kann nur im Auftrag des Grundbuchsgerichtes berichtigt werden; es hat, wenn der Fehler irgendeine Rechtsfolge nach sich ziehen könnte, die Beteiligten zu vernehmen. Die Einleitung des Verfahrens ist auf dem Blatt, wo die fehlerhafte Eintragung vollzogen worden ist, anzumerken. Die Anmerkung hat die Wirkung, daß spätere Eintragungen der Berichtigung des Fehlers nicht entgegenstehen. Sie ist nach Rechtskraft des über den wahrgenommenen Fehler ergangenen Beschlusses von Amts wegen zu löschen.

Anmerkungen

Nähere Bestimmungen enthält § 575 Geo, BGBl. Nr. 264/1951.

FÜNFTER ABSCHNITT.

Von den Simultanhypotheken.

1. Bestimmung einer Haupteinlage.

§ 105. (1) Bei Simultanhypotheken (§ 15), die durch Eintragung in verschiedene Grundbucheinlagen gebildet werden sollen, ist eine Einlage als Haupteinlage und sind die übrigen Einlagen als Nebeneinlagen zu bezeichnen. Fehlt eine solche Bezeichnung, so wird die im Gesuch erstgenannte Einlage als Haupteinlage angenommen.

(2) Wird um die Ausdehnung einer für dieselbe Forderung bereits haftenden Hypothek auf andere Grundbucheinlagen angesucht, so wird die ursprünglich belastete Einlage als Haupteinlage behandelt.

(3) Bei der Haupteinlage ist auf die Nebeneinlagen und bei jeder Nebeneinlage auf die Haupteinlage durch eine Anmerkung hinzuweisen.

2. Anzeige und Eintragung der Simultanhypotheken.

§ 106. (1) Wenn ein Gläubiger um die Ausdehnung des für seine Forderung haftenden Pfandrechtes ansucht, ist er verpflichtet, die für diese Forderung bereits bestehende Hypothek anzuzeigen, damit die Simultanhaftung angemerkt werde.

(2) Den durch Verschweigung einer bereits bestehenden Hypothek entstandenen Schaden hat der Gläubiger zu tragen.

§ 107. (1) Sollte die Anmerkung einer Simultanhaftung aus was immer für einen Grund unterblieben sein, so kann der Hypothekarschuldner um die Anmerkung ansuchen. Die hiedurch verursachten Kosten hat der Gläubiger zu ersetzen, wenn ihm diesfalls ein Verschulden zur Last fällt.

(2) Wenn ein Grundbuchsgericht bei der Bewilligung der Einverleibung oder Vormerkung des Pfandrechtes für eine Forderung wahrnimmt, daß ein Pfandrecht für diese Forderung in seinen oder in den Büchern eines anderen Grundbuchsgerichtes eingetragen ist, hat es von Amts wegen darauf Bedacht zu nehmen, die Einlage, in der das Pfandrecht eingetragen ist, als Haupteinlage anzusehen und die Grundbuchsgerichte, in deren Büchern die Forderung bereits eingetragen ist, hievon zu verständigen.

§ 108. (1) Die Eintragung einer Simultanhypothek bei mehreren Grundbuchsgerichten kann unter Anschluß von Originalurkunden oder beglaubigten Abschriften (§ 88) gleichzeitig bei den einzelnen Grundbuchsgerichten verlangt oder in einem einzigen Gesuch begehrt werden.

(2) Im ersten Fall sind in jedem Gesuch die Haupteinlage und alle Nebeneinlagen anzugeben.

(3) Im zweiten Fall ist das Gesuch bei dem Grundbuchsgericht anzubringen, bei dem die Haupteinlage geführt werden soll, und die Reihenfolge zu bezeichnen, in der das Gesuch den übrigen Grundbuchsgerichten zur Erledigung zuzusenden ist.

§ 109. (1) Wenn bei der ursprünglichen oder späteren Eintragung einer Simultanhypothek mehrere Grundbuchsgerichte mitzuwirken haben, hat jedes von ihnen hinsichtlich der in seinen Büchern enthaltenen Hypothekarobjekte über die Frage der Einverleibung oder Vormerkung des Pfandrechtes selbständig zu entscheiden und die Entscheidung dem Grundbuchsgericht der Haupteinlage bekanntzugeben.

(2) Der Rekurs gegen jeden der Beschlüsse ist bei dem Grundbuchsgericht anzubringen, das ihn erlassen hat.

(3) Ist die von einem Grundbuchsgericht in den Nebeneinlagen bewilligte Einverleibung oder Vormerkung im Rekursweg aufgehoben und gelöscht worden, so muß diese Löschung dem Grundbuchsgericht der Haupteinlage zur Anmerkung mitgeteilt werden.

§ 110. Für die Rangordnung einer Simultanhypothek ist bei jedem einzelnen Hypothekarobjekt der Zeitpunkt entscheidend, in dem das Ansuchen um die Bewilligung der Eintragung bei dem Grundbuchsgericht eingelangt ist, in dessen Büchern die Eintragung stattgefunden hat.

3. Eintragungen der Änderungen in der Haupteinlage.

§ 111. (1) Alle Grundbuchsgesuche, die sich auf ein in mehreren Einlagen simultan haftendes Pfandrecht beziehen, sind bei dem Grundbuchsgericht, bei dem die Haupteinlage geführt wird, anzubringen und nach dem Stand dieser Einlage zu beurteilen.

(2) Wenn das Gesuch bei einem anderen Grundbuchsgericht überreicht wird, ist es mit der Weisung zurückzustellen, daß es bei dem Grundbuchsgericht der Haupteinlage anzubringen ist.

Anmerkungen

Um die Rangänderung einer Simultanhypothek ist jedoch beim Gericht der Nebeneinlage anzusuchen.

§ 112. (1) Alle Änderungen, die an dem simultan haftenden Pfandrecht durch Übertragung, Beschränkung, Belastung, Löschung oder auf andere Weise vorgenommen werden sollen, sind nur in der Haupteinlage einzutragen.

(2) Die Eintragung der Änderungen in der Haupteinlage gilt rechtlich als in allen schon bestehenden oder noch hinzukommenden Nebeneinlagen vollzogen; doch ist die teilweise oder gänzliche Löschung der Simultanhypothek hinsichtlich aller Hypothekarobjekte auch in allen Nebeneinlagen und die Löschung des Pfandrechtes hinsichtlich einzelner Nebeneinlagen in diesen anzumerken.

§ 113. (1) Wenn das Pfandrecht hinsichtlich des in der Haupteinlage eingetragenen Hypothekarobjektes gelöscht wird, sind auch alle darauf erfolgten weiteren Eintragungen in der Haupteinlage zu löschen und in eine Nebeneinlage desselben Grundbuchsgerichtes zu übertragen. Sofern eine Simultanhypothek noch fortbesteht, ist diese Nebeneinlage in der Folge als Haupteinlage zu behandeln.

(2) Besteht in den Büchern dieses Grundbuchsgerichtes keine Nebeneinlage, so hat dieses Gericht, insofern eine Erklärung des Hypothekargläubigers nicht vorliegt, zu bestimmen, welche Nebeneinlage in Zukunft als Haupteinlage zu behandeln ist, und dem Grundbuchsgericht der neuen Haupteinlage beglaubigte Abschriften der im Hauptbuch bestehenden Eintragungen und der hierauf bezüglichen Urkundenabschriften von Amts wegen zu übermitteln.

(3) Die Umwandlung einer Nebeneinlage in die Haupteinlage ist den Grundbuchsgerichten aller Nebeneinlagen bekanntzugeben und bei jeder fortbestehenden Nebeneinlage von Amts wegen anzumerken.

§ 114. (1) Dem Grundbuchsgericht, an das die Führung der Haupteinlage übergeht, sind die Grundbuchsgesuche zu übersenden, die wegen der bereits erfolgten Löschung des Pfandrechtes in der Haupteinlage nicht mehr erledigt werden können; die Antragsteller sind hievon zu verständigen.

(2) Die Rangordnung dieser Gesuche untereinander wird durch den Zeitpunkt bestimmt, in dem sie bei dem Grundbuchsgerichte der früheren Haupteinlage eingelangt sind.

§ 115. (1) Wenn in Ansehung einer vor dem 16. Februar 1872 in verschiedenen Grundbucheinlagen erwirkten Simultanhypothek weitere Eintragungen erfolgen sollen, ist bei dem Ansuchen um eine neue Eintragung die Einlage zu bezeichnen, die als Haupteinlage geführt werden soll.

(2) In diese Einlage sind alle Eintragungen, die nach der Begründung der Simultanhypothek in Ansehung derselben in den anderen Einlagen vorgenommen worden sind, zu übertragen. Diese Übertragung ist unter Bezeichnung der Haupteinlage in den übrigen Einlagen, die fortan als Nebeneinlagen zu behandeln sind, anzumerken.

4. Rechtfertigungsklage.

§ 116. (1) Zur Rechtfertigung einer bei verschiedenen Grundbuchgerichten für dieselbe Forderung simultan haftenden Vormerkung des Pfandrechtes ist nur eine Rechtfertigungsklage erforderlich.

(2) Die Rechtfertigungsklage ist entweder bei dem allgemeinen Gerichtsstand des Hypothekarschuldners oder bei einem Gericht zu erheben, das in Ansehung eines der Hypothekarobjekte, auf die die Vormerkung bewilligt worden ist, die Realinstanz ist.

Anmerkungen

Zur Realinstanz siehe § 81 JN, RGBl. Nr. 111/1895.

5. Grundbuchsauszüge.

§ 117. In Grundbuchsauszügen über solche Einlagen, die in Ansehung einer Simultanhypothek als Nebeneinlagen geführt werden, ist der Hinweis auf die Haupteinlage und die Bemerkung aufzunehmen, daß die an dem simultan eingetragenen Pfandrecht vorgenommenen Änderungen nur in der Haupteinlage eingetragen sind.

SECHSTER ABSCHNITT.

Von der Zustellung.

§ 118. In jedem Beschluß sind die Personen sowie die Amtsstellen zu bezeichnen, denen er zuzustellen ist. Auch ist anzugeben, an wen mit dem Beschluß eine Urkunde zuzustellen ist. Inwieweit hievon bei Anmerkungen im Exekutionsverfahren abgesehen werden kann, bestimmt die Geschäftsordnung.

Anmerkungen

Nähere Bestimmungen enthält § 114 Abs. 4 Geo, BGBl. Nr. 264/1951.

§ 119. von den Erledigungen der Grundbuchsgesuche sind nebst dem Antragsteller nachstehende Personen von Amts wegen zu verständigen:

1. Derjenige, auf dessen Eigentum ein bürgerliches Recht erworben wird oder dessen bürgerliche Rechte abgetreten, belastet, beschränkt oder aufgehoben werden oder gegen den eine grundbücherliche Anmerkung erfolgt.

2. Wird die gänzliche oder teilweise Löschung einer Eintragung bewilligt, so ist der Beschluß auch allen zuzustellen, für die auf dem eingetragenen Recht weitere Einverleibungen oder Vormerkungen haften.

3. Beschlüsse über eine Einverleibung oder Vormerkung, wodurch bereits eingetragene Rechte dritter Personen verpfändet oder abgetreten werden, sind auch dem Eigentümer des Gutes zuzustellen.

4. Wird eine Eintragung gegen einen Machtgeber auf Ansuchen seines Machthabers erwirkt, so ist der Beschluß dem Machtgeber zuzustellen, es sei denn die Bevollmächtigung durch eine den Erfordernissen des § 31 entsprechende Vollmacht dargetan.

5. Von Änderungen, welche die im Grenzkataster oder im Grundsteuerkataster enthaltenen Angaben berühren, ist das Vermessungsamt in Kenntnis zu setzen.

6. Beschlüsse über die Einverleibung oder Vormerkung des Eigentums sind auch dem an erster Stelle stehenden Pfandgläubiger zuzustellen.

Anmerkungen

1. Siehe auch die §§ 130 und 134.

2. Abweichende Bestimmungen enthält § 16 GUG, BGBl. Nr. 550/1980.

§ 120. (1) Die Zustellung an die im § 119 Z. 1 bis 4 bezeichneten Personen hat nach den über die Zustellung zu eigenen Händen in der Zivilprozeßordnung enthaltenen Vorschriften zu geschehen.

(2) Die Originalurkunden sind, insofern nicht in dem Gesuch eine andere Verfügung beantragt wird, dem zurückzustellen, der sie überreicht hat.

(3) Die Grundbuchsgerichte sind verpflichtet, über die schnelle und richtige Zustellung der Beschlüsse in Grundbuchssachen zu wachen.

Anmerkungen

Zu Abs. 1: Von den zitierten Vorschriften gilt nur noch § 106 ZPO, RGBl. Nr. 113/1895, im übrigen ist die Zustellung nunmehr im Zustellgesetz, BGBl. Nr. 200/1982, geregelt.

§ 121. Der Umstand, daß eine Zustellung ordnungswidrig oder gar nicht erfolgt ist, gibt keine Berechtigung, die Gültigkeit der bücherlichen Eintragung zu bestreiten. Wer aus einer bücherlichen Eintragung für sich Rechte oder eine Befreiung von Verbindlichkeiten ableitet, ist nicht schuldig, den Beweis der Zustellung zu liefern.

Anmerkungen

Siehe auch § 64.

SIEBENTER ABSCHNITT.

Vom Rekurs.

1. Anbringung des Rekurses.

§ 122. (1) Gegen Grundbuchsbeschlüsse ist nur das Rechtsmittel des Rekurses zulässig.

(2) Im Rekurs dürfen weder neue Angaben gemacht noch dürfen ihm neue Urkunden beigelegt werden.

(3) Der Rekurs ist stets in erster Instanz anzubringen. Er kann auch mündlich zu Protokoll gegeben werden.

(4) Einem schriftlichen Rekurs sind die zur Verständigung der Beteiligten erforderlichen Halbschriften beizulegen.

(5) Ein unmittelbar bei der zweiten oder dritten Instanz überreichten Rekurs ist zurückzuweisen.

(6) Beschwerden über Verzögerungen können unmittelbar bei den höheren Gerichten angebracht werden.

Anmerkungen

1. Die Rekurslegitimation richtet sich nach den Vorschriften des AußStrG, RGBl. Nr. 208/1854.

2. Abweichende Bestimmungen enthalten § 134 sowie § 62 AllgGAG, BGBl. Nr. 2/1930, § 32 LiegTeilG, BGBl. Nr. 3/1930, § 43 EAG, RGBl. Nr. 70/1874.

§ 123. (1) Die Rekursfrist beträgt bei Zustellungen im Inland 30 Tage, bei Zustellungen im europäischen Ausland, mit Ausnahme von Island und den Färöern, 60 Tage, bei Zustellungen im außereuropäischen Auslande sowie in Island und den Färöern 90 Tage (§ 81).

(2) Ein verspäteter Rekurs ist von der ersten Instanz sogleich zurückzuweisen, wenn auch die in das Grundbuch eingetragene Anmerkung des abschlägigen Beschlusses noch nicht gelöscht sein sollte.

§ 124. Ein rechtzeitig angebrachter Rekurs ist unter Anschluß der zur Entscheidung erforderlichen Akten der zweiten Instanz zur eigenen Entscheidung oder, wenn der Rekurs gegen eine Erledigung der zweiten Instanz gerichtet sein sollte, zur Beförderung an die dritte Instanz vorzulegen. Hievon sind die Personen, denen der angefochtene Beschluß zugestellt worden ist, zu verständigen. Eine Verständigung des Rekurrenten ist nicht erforderlich.

§ 125. (1) Ist der Rekurs gegen die Bewilligung einer Einverleibung oder Vormerkung gerichtet, so ist er im Grundbuch anzumerken und diese Anmerkung nach der Erledigung des Rekurses zu löschen, wenn ein Rekurs an den Obersten Gerichtshof nach § 14 Abs. 2 AußStrG unzulässig ist.

(2) Diese Anmerkung sowie die Löschung haben von Amts wegen zu erfolgen.

Anmerkungen

Der Einverleibung steht die Ab- oder Zuschreibung gleich (§ 74).

Zu Abs. 1: ÜR: Art. XLI Z 5, BGBl. Nr. 343/1989

§ 126. (1) Für die Entscheidung des Rekursgerichts gilt § 13 AußStrG.

(2) Der Beschluß des Rekursgerichts kann nach Maßgabe der §§ 14 und 15 AußStrG angefochten werden, wobei die Bestimmungen der §§ 122 bis 125 zu beachten sind. Ein Revisionsrekurs, der aus einem anderen Grund als wegen des Fehlens der Voraussetzungen nach § 14 Abs. 1 AußStrG unzulässig ist, ist von der ersten Instanz zurückzuweisen.

(3) Der Erledigung des Rekurses sind, wenn der Beschluß, gegen den er gerichtet war, abgeändert oder aus wesentlich abweichenden Gründen bestätigt wird, die Entscheidungsgründe beizufügen.

Anmerkungen

Zu Abs. 1 und 2: ÜR: Art. XLI Z 5, BGBl. Nr. 343/1989

2. Wirkung der Rekurerledigung.

§ 127. Wenn ein Revisionsrekurs nach § 14 Abs. 2 AußStrG unzulässig ist, so ist die Löschung der im Grundbuch eingetragenen Anmerkung der Abweisung und die Verständigung der Beteiligten von Amts wegen zu veranlassen.

Anmerkungen

ÜR: Art. XLI Z 5, BGBl. Nr. 343/1989

§ 128. Ist einem der im § 99 angeführten Gesuche, das in erster Instanz abgewiesen worden ist, von der höheren Instanz stattgegeben worden, so ist diese Bewilligung im Grundbuch einzutragen. Die Wirkung dieser Eintragung ist so zu beurteilen, als ob sie in dem Zeitpunkt der Überreichung des ersten Gesuches erfolgt wäre.

Anmerkungen

ÜR: Art. XLI Z 5, BGBl. Nr. 343/1989

§ 129. (1) Wird eine von der ersten Instanz bewilligte Löschung von der zweiten Instanz aufgehoben, so muß die gelöschte Einverleibung oder Vormerkung wiederhergestellt werden.

(2) Wird aber ein anderes der im § 99 angeführten Gesuche, das in erster Instanz bewilligt worden ist, von der zweiten Instanz abgewiesen und ist der Revisionsrekurs nicht nach § 14 Abs. 2 AußStrG unzulässig, so ist diese Verfügung im Grundbuch anzumerken, das eingetragene Recht aber nicht zu löschen, solange nicht entweder die Entscheidung des Obersten Gerichtshofs ergangen oder die Frist zur Ergreifung des Rekurses gegen die Anordnung der zweiten Instanz verstrichen ist. Bestätigt die dritte Instanz den Beschluß der ersten Instanz, so ist die durch den Rekurs veranlaßte Anmerkung zu löschen. Wird die abändernde Verfügung der zweiten Instanz von der dritten bestätigt oder in gehöriger Zeit kein Rekurs dagegen ergriffen, so ist das einverleibte oder vorgemerkte Recht zu löschen.

Anmerkungen

ÜR: Art. XLI Z 5, BGBl. Nr. 343/1989

VIERTES HAUPTSTÜCK.

Von der Bereinigung und Berichtigung des Grundbuches.

ERSTER ABSCHNITT.

Bereinigung des Grundbuches von Amts wegen.

1. Unzulässige Eintragungen.

§ 130. Ergibt sich aus einer Eintragung, daß ihr Inhalt nach dem Gesetz nicht Gegenstand einer grundbücherlichen Eintragung sein kann, so ist sie von Amts wegen als unzulässig zu löschen. Die Vorschriften des ersten bis dritten und fünften Hauptstückes, insbesondere über die Verständigung der Beteiligten und den Rekurs, sind entsprechend anzuwenden.

2. Gegenstandslose Eintragungen.

§ 131. (1) Ist eine Eintragung gegenstandslos, so kann sie das Grundbuchsgericht gemäß den §§ 132 bis 135 von Amts wegen löschen.

(2) Eine Eintragung ist gegenstandslos, soweit das ihren Gegenstand bildende Recht oder das Recht, auf das sie sich bezieht,

- a) nicht besteht oder aus tatsächlichen Gründen dauernd nicht ausgeübt werden kann,
- b) verjährt ist,

c) für den Berechtigten einen lediglich wirtschaftlichen Wert darstellt, der 1 500 S, bei wiederkehrenden Leistungen 500 S jährlich, nicht übersteigt, sofern die Eintragung des Rechtes vor dem 1. Mai 1945 erfolgt ist.

(3) Im Falle des Abs. 2 lit. c bedarf es zur Löschung eines Pfandrechtes nicht der Zustimmung des Eigentümers, dem das Verfügungsrecht nach § 469 ABGB. zusteht.

(4) Abs. 2 lit. c gilt auch für Pfandrechte, bei denen gemäß Artikel 3 der Grundbuchsnovelle, BGBl. Nr. 4/1930, ein Antrag auf Aufrechterhaltung angemerkte ist.

Anmerkungen

Vgl. § 19 Abs. 2 GUG, BGBl. Nr. 550/1980, über die Unterlassung der Ersterfassung gegenstandsloser Eintragungen.

§ 132. (1) Das Grundbuchsgericht soll das Verfahren zur Löschung gegenstandsloser Eintragungen einleiten, wenn besondere äußere Umstände (zum Beispiel Umschreibung der Grundbuchseinlage wegen Unübersichtlichkeit, Teilveräußerung oder Neubelastung des Grundstückes, Anregung seitens eines Beteiligten) hinreichenden Anlaß dazu geben und Grund zu der Annahme besteht, daß die Eintragung gegenstandslos ist.

(2) Das Grundbuchsgericht entscheidet nach freiem Ermessen, ob das Lösungsverfahren einzuleiten und durchzuführen ist; diese Entscheidung ist unanfechtbar.

§ 133. (1) Voraussetzung für die Löschung ist, daß

- a) die Gegenstandslosigkeit der Eintragung offenkundig oder durch öffentliche oder gerichtlich oder notariell beglaubigte Urkunden nachgewiesen ist, oder daß, falls dies nicht zutrifft,
- b) dem Betroffenen eine Lösungsankündigung unter kurzer Bekanntgabe des Grundes zugestellt ist und er nicht binnen einer vom Grundbuchsgericht zugleich zu bestimmenden Frist Widerspruch erhoben hat, oder daß, falls auch nach lit. b nicht verfahren werden kann, insbesondere wenn Widerspruch erhoben ist,
- c) durch einen mit Gründen versehenen Beschluß rechtskräftig festgestellt ist, daß die Eintragung gegenstandslos ist.

(2) Kann die Löschung nicht sogleich angeordnet werden, so ist die Einleitung des Verfahrens im Grundbuch anzumerken. Die Anmerkung hat die Wirkung, daß spätere Eintragungen die Löschung nicht hindern. Sie ist zu löschen, wenn die gegenstandslose Eintragung gelöscht oder von der Fortsetzung des Verfahrens Abstand genommen wird. Einer Verständigung der Beteiligten von der Anordnung dieser Anmerkung und ihrer Löschung bedarf es nicht. Gegen diese Anordnungen ist ein Rechtsmittel nicht zulässig.

§ 134. Für das Verfahren gelten sinngemäß die Vorschriften des dritten Hauptstückes. Die Vorschriften über das Verfahren außer Streitsachen sind, soweit erforderlich, ergänzend heranzuziehen. Dabei gilt folgendes:

a) Eine Verweisung der Beteiligten auf den Rechtsweg oder das Verwaltungsverfahren (§ 2 Z. 7 des Kaiserlichen Patentgesetzes vom 9. August 1854, RGBl. Nr. 208) findet nicht statt;

b) die Löschungsankündigung (§ 133 Abs. 1 lit. b) kann nicht durch öffentliche Bekanntmachung zugestellt werden;

c) ist die Person des Beteiligten, dem zugestellt werden soll, unbekannt, so sind die Vorschriften über die Zustellung durch öffentliche Bekanntmachung sinngemäß anzuwenden;

d) die Rechtsmittel gegen die Entscheidungen des Grundbuchsgerichtes, mit denen die Löschung gegenstandslos gewordener Eintragungen angeordnet wird, richten sich nach dem siebenten Abschnitt des dritten Hauptstückes; im übrigen gelten für die Anfechtung von Entscheidungen die Vorschriften über das Verfahren außer Streitsachen. Gegen die Löschungsankündigung (§ 133 Abs. 1 lit. b) ist kein Rechtsmittel gegeben.

§ 135. Ist ein Beteiligter durch eine nach den §§ 131 ff. bewilligte Löschung in seinem bürgerlichen Rechte verletzt, so kann er im Prozeßwege die Wiederherstellung des vorigen bürgerlichen Standes begehren. Die Vorschriften der §§ 61 ff. sind sinngemäß anzuwenden.

ZWEITER ABSCHNITT.

Berichtigung des Grundbuches auf Ansuchen.

§ 136. (1) Gibt das Grundbuch die wirkliche Rechtslage nicht richtig wieder, so ist auf Ansuchen die zur Berichtigung erforderliche Eintragung vorzunehmen, ohne daß die sonst für eine solche Eintragung von diesem Bundesgesetz geforderten Voraussetzungen erfüllt sein müssen, wenn die Unrichtigkeit offenkundig oder durch öffentliche Urkunden nachgewiesen ist. Soweit dieser Nachweis durch die Erklärung eines Beteiligten erbracht werden kann, genügt eine gerichtlich oder notariell beglaubigte Privaturkunde.

(2) Würden durch die Berichtigung nach Abs. 1 bestehende bürgerliche Rechte Dritter betroffen, so kann die Berichtigung nur unter Wahrung dieser Rechte (zum Beispiel nach § 51) bewilligt werden.

(3) Die Löschung eines Rechtes auf wiederkehrende Leistungen kann nach Abs. 1 nur bewilligt werden, wenn seit dem Erlöschen des Bezugsrechtes (§ 18) drei Jahre verstrichen sind und keine Klage auf Zahlung von Rückständen im Grundbuch angemeldet ist.

Anmerkungen

Die Voraussetzung ist nur dann gegeben, wenn die Rechtsänderung ohne Eintragung in das Grundbuch eingetreten ist (Ausnahmen vom Eintragungsgrundsatz).

FÜNFTES HAUPTSTÜCK.

Schlußbestimmungen.

§ 137. (1) Dieses Bundesgesetz tritt drei Monate nach seiner Kundmachung in Kraft.

(2) Gleichzeitig treten außer Kraft:

1. das Gesetz vom 25. Juli 1871, RGBl. Nr. 95, über die Einführung eines Allgemeinen Grundbuchgesetzes;

2. die §§ 1 und 3 des Gesetzes vom 4. Juni 1882, RGBl. Nr. 67, enthaltend Bestimmungen über die Entbehrlichkeit der Legalisierung gewisser Unterschriften auf Tabularurkunden und

über Erleichterungen des Beweises der Identität einer Person bei Legalisierungen und anderen Beurkundungen;

3. das Gesetz vom 5. Juni 1890, RGBl. Nr. 109, betreffend die grundbücherliche Einverleibung auf Grund von Privaturkunden in geringfügigen Grundbuchsachen;

4. Artikel XXXIX des Gesetzes vom 1. August 1895, RGBl. Nr. 112, betreffend die Einführung des Gesetzes über das gerichtliche Verfahren in bürgerlichen Rechtsstreitigkeiten (Zivilprozeßordnung);

5. die §§ 37 erster und zweiter Satz, 38 bis 41, 44 bis 46, 47 Abs. 1, 48 bis 50 sowie die Worte „des § 30 des Allgemeinen Grundbuchsgesetzes und“ in § 51 Abs. 2 der Kaiserlichen Verordnung vom 19. März 1916, RGBl. Nr. 69, über die dritte Teilnovelle zum Allgemeinen Bürgerlichen Gesetzbuch;

6. das Bundesgesetz vom 4. Juni 1923, BGBl. Nr. 306, über die Erhöhung der Wertgrenze im Gesetz vom 5. Juni 1890, RGBl. Nr. 109, betreffend die grundbücherliche Einverleibung auf Grund von Privaturkunden in geringfügigen Grundbuchsachen;

7. die Artikel I bis III des Bundesgesetzes vom 31. März 1927, BGBl. Nr. 118, betreffend geringfügige Grundbuchsachen;

8. Artikel 1 der Grundbuchs-novelle BGBl. Nr. 4/1930;

9. die Verordnung zur Änderung und Ergänzung des Grundbuchsrechts im Geltungsbereich des Österreichischen Allgemeinen Grundbuchsgesetzes vom 19. Jänner 1942, Deutsches RGBl. I S. 37, in der Fassung der Druckfehlerberichtigung Deutsches RGBl. 1942 I S. 50; § 10 Abs. 2 dieser Verordnung tritt jedoch erst am 1. Jänner 1960 außer Kraft;

10. die §§ 1 und 3 des Bundesgesetzes vom 21. Juni 1950, BGBl. Nr. 141, über die Änderung einiger grundbuchsrechtlicher Vorschriften;

11. Artikel XXVI des Einführungsgesetzes zur Exekutionsordnung, BGBl. Nr. 6/1953.

(3) Andere Vorschriften grundbuchsrechtlichen Inhaltes bleiben unberührt.

§ 138. In Ansehung der Bergbücher sind nebst diesem Bundesgesetz auch die Vorschriften des Berggesetzes BGBl. Nr. 73/1954 zu beachten.

Anmerkungen

Statt: Berggesetz, BGBl. Nr. 73/1954, nunmehr: Berggesetz, BGBl. Nr. 259/1975.

§ 139. Soweit in anderen Bundesgesetzen die durch dieses Bundesgesetz aufgehobenen Vorschriften zitiert sind, treten an deren Stelle die entsprechenden Bestimmungen dieses Bundesgesetzes.

§ 140. Mit der Vollziehung sind betraut:

a) hinsichtlich des § 90, soweit sich dieser auf die Befreiung von Stempelgebühren bezieht, das Bundesministerium für Finanzen;

b) hinsichtlich des § 137 Abs. 2 Z. 9, soweit sich diese Bestimmung auf die Aufhebung des § 10 Abs. 2 der in § 137 Abs. 2 Z. 9 genannten Verordnung bezieht, das Bundesministerium für Inneres;

c) im übrigen das Bundesministerium für Justiz.

Übergangsregelungen:

§§ 125 bis 127 und 129 GBG 1955 sind anzuwenden, wenn das Datum der Entscheidung der zweiten Instanz nach dem 31. Dezember 1997 liegt.

§§ 125 bis 129 GBG sind anzuwenden, wenn das Datum der Entscheidung der zweiten Instanz nach dem 31. Dezember 1989 liegt.

Anhang E: Modell des GBG55

Zusätzliche Operationen für Listen

```
module AddList where
```

```
import List
```

```
toOrd :: (Eq a) => a -> a -> Ordering
```

```
toOrd a b c = if a == c then LT else GT
```

```
insertBefore :: (Eq a) => a -> a -> [a] -> [a]
```

```
insertBefore s i xs = insertBy (toOrd s) i xs
```

Rechte und Dokumente

```

module Recht where

import List
import AddList

-- *****
-- allgemeine Funktionen
-- *****

pair (f,g) a = (f a, g a)

branch f (g,h) a = if f a then g a else h a

getChangePut :: ((a -> b),(b -> b),(a -> b -> a)) -> a -> a
getChangePut (g,c,p) ds = p ds (c (g ds))

-- *****
-- Datentypen
-- *****

type Anteil      = Rational -- § 10
zero = 0 % 1 :: Anteil
one  = 1 % 1 :: Anteil

type Betrag      = Float
type Person      = String
type Beschreibung = String
type Abgrenzung  = String
type Ort          = String
type Art          = String
type Zeuge       = Person
type Nr          = String
type RONr        = Int

```

```

noRO = -1 :: RONr
oeffentlUrkunde = "oeffUrk"           :: Art
notar           = "Notar"             :: Art
gericht        = "Gericht"           :: Art
zweiZeugen     = "Zeugen"           :: Art
  -- § 34/1, bei geringfuegigen Grundbuchssachen
behoerdeGenemigt = "Behoerde"        :: Art
ausland        = "Vertretungsbehoerde" :: Art
ausland2       = "BM Justiz"         :: Art

erlaubteUrk = [oeffentlUrkunde,notar,gericht,
               behoerdeGenemigt,ausland,ausland2]
  -- § 31/1 - 31/5

data DokTyp      = DokTyp Art Zeuge deriving (Eq, Show)

data Datum = Datum Int Int Int deriving (Eq, Show)
  -- Tag, Monat, Jahr

months = [31,28,31,30,31,30,31,31,30,31,30,31] :: [Int]

data RTyp = Eigentum Anteil RONr |
          Pfandrecht Betrag Betrag RONr |
  -- Geldsumme, Zinssatz ziffernm. Geldsumme, Zinssatz
  -- (§ 14/1) oder Hoechstbetrag (§ 14/2)
  -- § 13/1 Pfandrecht auf Anteil: Nicht alle Eigentuemmer
  -- sind Rechtsbegruender (b. Dokument)
          Simultanhypothek Betrag Betrag [Nr] RONr |
  -- § 105/1 [Nr]: Grundbuchseinlagen, 1.: Haupteinlage
          PfandR2Simult (Recht RTyp) [Nr] RONr | -- § 105/2
  -- Umwandlung Pfandrecht in Simultanhypothek, 1. Grund-
  -- buchseinlage ist Einlage des Pfandrechtes
          Afterpfandrecht (Recht RTyp) RONr |
          Dienstbarkeit Beschreibung RONr | -- § 12/1
          BeschrDienstbarkeit Beschreibung Abgrenzung RONr |
  -- § 12/2
          Reallast Beschreibung RONr | -- § 12/1
          Wiederkaufsrecht Betrag RONr |
          Vorkaufsrecht RONr |

```

```

        Bestandrecht Beschreibung RONr | -- § 19
        DinglRecht Beschreibung RONr |
        Vorrang (Recht RTyp) (Recht RTyp) |
-- bevorrangt zuruecktretend
        Rangordnung RONr |
        Rechtfertigung (Recht RTyp) |
        Anm Beschreibung |
-- § 20: persoenl.Verhaeltnisse, Begrueendung
--      von.Rechtswirkungen
-- § 60: Hypothekarklage
-- § 61: Loeschungsklage
        AnmZwangsversteigerung | -- § 72
        AnmAbweisung (Recht RTyp) | -- § 99
        LoeschVB Datum Datum |
-- fuer § 57: von Datum,bis Datum Loeschung (Recht RTyp)
-- § 8
deriving (Eq, Show) -- § 9

data Recht t = R t [Person] {- Berechtigte -}
              [Person] {- Belastete -}
              deriving (Eq, Show)
-- bei Loeschung Eigentuemmer, Berechtigte des zuloeschenden
-- Rechtes
-- t ... Rechtstyp

data DokumentArt = Original | Abschrift String -- § 87
                 deriving (Eq, Show)
-- String: Ort, an dem das Original liegt

data Eigenschaften = Eig Int Bool Bool Bool DokumentArt
                   deriving (Eq, Show)
-- Seiten, geheftet, maengel, beglaubigt, Original?

data Dokument r t = D (r t) [Person] [Nr] DokTyp
                    Eigenschaften
                    deriving (Eq, Show)
-- Recht, Unterzeichner, Art des Dokuments
-- Rechtsbegrueendung, § 26/2

```

```

-- Nr = Liegenschaft(en) (§ 32/1)
-- Liste von Grundbuchseinlagen, in die eingetragen werden
-- soll: § 86

-- *****
-- Klassen
-- *****

class Datums a where
  newdatum :: Int -> Int -> Int -> a
  tag,monat,jahr    :: a -> Int

  incTag :: a -> a
  incTag d = if tag d + 1 <= (months !! (monat d -1))
             then newdatum (tag d + 1) (monat d) (jahr d)
             else if monat d + 1 <= 12
                   then newdatum 1 (monat d + 1) (jahr d)
                   else newdatum 1 1 (jahr d + 1)

  incTage :: a -> Int -> a
  incTage d 0 = d
  incTage d x = incTage (incTag d) (x-1)

class DokTypen t where
  erstellung :: t -> Art
  zeuge      :: t -> Zeuge

  spezErfOK    :: t -> Bool
  spezErfOK dt = elem (erstellung dt) erlaubteUrk

class TypTest r where
  istEigentum,istPfandrecht,istSimultan,istAfterpfand,
  istDienstbarkeit,istBDienstbarkeit,istReallast,
  istWiederkaufsrecht,istVorkaufsrecht,istBestandrecht,
  istDinglRecht, istVorrang,istRechtfertigung,
  istRangordnung,istAnmerkung,istAbweisung,
  istZwangsversteigerung,istLoeschung :: r -> Bool

```



```

                                (eigentumsanteil r))
                                else []

anteilVon :: Person -> r t -> Anteil
anteilVon p r =
    if istEigentum r then (foldl (+) 0.map snd.
                                filter t.eigentuemer) r
    else zero
        where t (e,a) = e == p

enthRecht    :: r t -> Bool
enthRecht    = typEnthRecht.rechtstyp

betrags      :: r t -> Betrag
betrags      = typBetrags.rechtstyp

beschreibung  :: r t -> Beschreibung
beschreibung  = typBeschr.rechtstyp

roNummer     :: r t -> RONr
roNummer     = typRONummer.rechtstyp

rangsort :: [r t] -> [r t]
rangsort rs = filter (istEigentum.rechtstyp) srs ++
                filter (not.istEigentum.rechtstyp) srs where
-- § 30/1 (ohne Zustimmungen), § 30/2, § 30/4
    srs = rsort rs rs
    rsort [] res = res
    rsort (r:rs) res =
        if (istVorrang.rechtstyp) r
        then rsort rs ((delete r.insertBefore r1 r2.delete r2)
                        res)
        else if (istRangordnung.rechtstyp) r
        then if r3 /= r
            then rsort rs ((delete r.insertBefore r
                        r3.delete r3) res)
            else rsort rs res
        else rsort rs res
    where r1 = betrRecht2 r
          r2 = betrRecht r
          r3 = if length r3' == 0 then r else head r3'
          r3' = filter ((roNummer r ==).roNummer)
                (res \\ [r])

```

```

class (Rechte r t) => Dokumente d r t where
  recht          :: d r t -> r t
  unterzeichner :: d r t -> [Person]
  dokumentTyp    :: d r t -> DokTyp
  liegenschaften:: d r t -> [Nr]
  eigenschaften  :: d r t -> Eigenschaften
  makeRO         :: [Person] -> Nr -> [d r t] -> d r t
  makeLoeschVB  :: Datum -> Datum -> [Person] -> Nr -> d r t
  makeAbweisung :: r t -> [Nr] -> [Person] -> d r t
  makeLoeschung :: r t -> [Nr] -> [Person] -> d r t
  makeAnmSimultan :: [Person] -> [Nr] -> d r t

-- derived:
eintragErlaubt :: d r t -> [Person] -> Bool
  -- Dokument, Eigentuemer
eintragErlaubt d es =
  (sort (intersect (gegen ++ fuer) (unterzeichner d))
   == sort (gegen ++ fuer)) &&
  (sort (intersect es (unterzeichner d)) == sort es)
  where fuer = (berechtigte.recht) d
        -- bei Anmerkung: Eintragende Instanz (§ 52)
        gegen = (belastete .recht) d
        -- bei Anmerkung: Bezeugung z.B. durch Gericht (§ 52)

dokumentOK      :: d r t -> Bool
dokumentOK      = eigOK.eigenschaften

dienende        :: (Rechte r t) => d r t -> [Person]
dienende        = belastete.recht

spezErfordernisse:: d r t -> Bool
spezErfordernisse = spezErfOK.dokumentTyp -- § 31/1-31/6

-- *****
-- Instanzen
-- *****

instance Datums Datum where
  newdatum t m j = Datum t m j
  tag (Datum t m j) = t

```

```

monat (Datum t m j) = m
jahr (Datum t m j)  = j

```

```

instance Ord Datum where
  compare (Datum t1 m1 j1) (Datum t2 m2 j2)
    | (t1 == t2) && (m1 == m2) && (j1 == j2) = EQ
    | (j1 > j2) = GT
    | (j1 < j2) = LT
    | (m1 > m2) = GT
    | (m1 < m2) = LT
    | (t1 > t2) = GT
    | (t1 < t2) = LT
    | otherwise = error "no other case fits"

```

```

instance DokTypen DokTyp where
  erstellung (DokTyp a z) = a
  zeuge      (DokTyp a z) = z

```

```

instance TypTest RTyp where
  istEigentum (Eigentum _ _) = True
  istEigentum _                = False
  istPfandrecht (Pfandrecht _ _ _) = True
  istPfandrecht _                = False
  istSimultan (Simultanhypothek _ _ _ _) = True
  istSimultan (PfandR2Simult _ _ _)      = True
  istSimultan _                          = False
  istAfterpfand (Afterpfandrecht _ _) = True
  istAfterpfand _                          = False
  istDienstbarkeit (Dienstbarkeit _ _) = True
  istDienstbarkeit _                          = False
  istBDienstbarkeit (BeschrDienstbarkeit _ _ _) = True
  istBDienstbarkeit _                          = False
  istReallast (Reallast _ _) = True
  istReallast _                = False

  istWiederkaufsrecht (Wiederkaufsrecht _ _) = True
  istWiederkaufsrecht _                          = False
  istVorkaufsrecht (Vorkaufsrecht _) = True
  istVorkaufsrecht _                = False

```

```

istBestandrecht (Bestandrecht _ _) = True
istBestandrecht _                    = False
istDinglRecht (DinglRecht _ _) = True
istDinglRecht _                    = False
istVorrang (Vorrang _ _) = True
istVorrang _                    = False
istRechtfertigung (Rechtfertigung _) = True
istRechtfertigung _                    = False
istAnmerkung (Anm _) = True
istAnmerkung _      = False
istAbweisung (AnmAbweisung _) = True
istAbweisung _      = False
istZwangsversteigerung (AnmZwangsversteigerung) = True
istZwangsversteigerung _                    = False
istRangordnung (Rangordnung _) = True
istRangordnung _                    = False
istLoeschung (Loeschung _) = True
istLoeschung _                    = False

instance (TypTest t, Rechte Recht t) => TypTest (Recht t)
  where
    istEigentum           = istEigentum.rechtstyp
    istPfandrecht         = istPfandrecht.rechtstyp
    istSimultan           = istSimultan.rechtstyp
    istAfterpfand         = istAfterpfand.rechtstyp
    istDienstbarkeit     = istDienstbarkeit.rechtstyp
    istBDienstbarkeit    = istBDienstbarkeit.rechtstyp
    istReallast           = istReallast.rechtstyp
    istWiederkaufsrecht  = istWiederkaufsrecht.rechtstyp
    istVorkaufsrecht     = istVorkaufsrecht.rechtstyp
    istBestandrecht       = istBestandrecht.rechtstyp
    istDinglRecht        = istDinglRecht.rechtstyp
    istVorrang            = istVorrang.rechtstyp
    istRechtfertigung    = istRechtfertigung.rechtstyp
    istAnmerkung         = istAnmerkung.rechtstyp
    istAbweisung         = istAbweisung.rechtstyp
    istZwangsversteigerung = istZwangsversteigerung.rechtstyp
    istRangordnung       = istRangordnung.rechtstyp
    istLoeschung         = istLoeschung.rechtstyp

```

```

instance Rechtstypen RTyp where
  anteil (Eigentum a _) = a
  anteil _ = zero
  typBetrag (Pfandrecht b p _) = b
  typBetrag (Wiederkaufsrecht b _) = b
  typBetrag _ = 0
  typBeschr (Dienstbarkeit b _) = b
  typBeschr (BeschrDienstbarkeit b a _) = b
  typBeschr (Reallast b _) = b
  typBeschr (Bestandrecht b _) = b
  typBeschr (DinglRecht b _) = b
  typBeschr (Anm b) = b
  typBeschr _ = ""
  typRONummer (Rangordnung n) = n
  typRONummer (Eigentum _ n) = n
  typRONummer (Pfandrecht _ _ n) = n
  typRONummer (Afterpfandrecht _ n) = n
  typRONummer (Dienstbarkeit _ n) = n
  typRONummer (BeschrDienstbarkeit _ _ n) = n
  typRONummer (Reallast _ n) = n
  typRONummer (Wiederkaufsrecht _ n) = n
  typRONummer (Vorkaufsrecht n) = n
  typRONummer (Bestandrecht _ n) = n
  typRONummer (DinglRecht _ n) = n
  typRONummer _ = noRO
  typHaupteinlage (Simultanhypothek _ _ (x:xs) _) = x
  typHaupteinlage (Pfandr2Simult _ (x:xs) _) = x
  typHaupteinlage _ = error "Keine Simultanhypothek"
  typNebeneinlage (Simultanhypothek _ _ (x:xs) _) = xs
  typNebeneinlage (Pfandr2Simult _ (x:xs) _) = xs
  typNebeneinlage _ = error "Keine Simultanhypothek"
  typVonDt (LoeschVB d1 d2) = d1
  typVonDt _ = Datum 31 12 5000
  typBisDt (LoeschVB d1 d2) = d2
  typBisDt _ = Datum 1 1 1

instance Eigenschafte Eigenschaften where
  geheftet (Eig s e1 e2 e3 a) = e1
  maengel (Eig s e1 e2 e3 a) = e2
  beglaubigt (Eig s e1 e2 e3 a) = e3

```

```

seiten      (Eig s e1 e2 e3 a) = s
originalOK (Eig _ _ _ _ (Abschrift "")) = False
originalOK _                               = True

```

```

instance Rechte Recht RTyp where
  berechnete (R t pls p2s) = pls
  belastete  (R t pls p2s) = p2s
  rechtstyp  (R t pls p2s) = t
  betrRecht (R (Afterpfandrecht r _) _ _) = r
  betrRecht (R (Vorrang r1 r2) _ _)      = r1
  betrRecht (R (Rechtfertigung r) _ _)   = r
  betrRecht (R (Loeschung r) _ _)        = r
  betrRecht (R (AnmAbweisung r) _ _)     = r
  betrRecht (R (PfandR2Simult r _ _) _ _) = r
  betrRecht r                             = r
  betrRecht2 (R (Vorrang r1 r2) _ _)     = r2
  betrRecht2 r                             = betrRecht r

```

```

instance Dokumente Dokument Recht RTyp where
  recht          (D r us ls {-d o-} t e) = r
  unterzeichner (D r us ls {-d o-} t e) = us
  dokumentTyp   (D r us ls {-d o-} t e) = t
  eigenschaften (D r us ls {-d o-} t e) = e
  liegenschaften (D r us ls {-d o-} t e) = ls
  makeRO ps nr ds =
    D (R (Rangordnung ((1+).length.
                    filter (istRangordnung.recht)) ds))
      ps ps) ps [nr]
      (DokTyp gericht "Grundbuch")
      (Eig 1 False False True Original)
  makeLoeschVB d1 d2 ps nr =
    D (R (LoeschVB d1 d2) ps []) ps [nr]
      (DokTyp gericht "")
      (Eig 1 False False True Original)
  makeAbweisung r nrs es = D (R
      (AnmAbweisung r) es []) es nrs
      (DokTyp gericht "Grundbuch")
      (Eig 1 False False True Original)
  makeLoeschung r nrs ps =
    D (R (Loeschung r) [] ps) ps nrs

```

```
(DokTyp gericht "Grundbuch")
(Eig 1 False False True Original)
makeAnmSimultan ps es =
  D (R (Anm ("Simultanhypothek: " ++ (list es))) ps ps)
    ps es (DokTyp gericht "Grundbuch")
  (Eig 1 False False True Original)
  where list = (foldl (++) "").(map (" "++))
```

Grundbuch

```
module Grundb3 where

import List
import AddList
import Recht

-- nicht formalisierbar:
-- § 4
-- § 11 - Widerspruch zu § 3/1
-- § 14/3, 14/4
-- § 15/2
-- § 23, 24, 25
-- § 28 (Verweis auf §§ 63 ff)
-- § 33 (Definition oeffentlicher Urkunden)
-- § 34 (Wann reichen 2 Zeugen fuer die Urkunde)
-- § 38 (Gruende fuer Vormerkung)
-- § 39 (Gruende fuer Vormerkung der Loeschung)
-- § 40 (Rechtsbegrueundung durch Vormerkung)
-- § 42 (Rechtfertigung ueber Klage)
-- § 44
-- § 45/2, 45/3 (Rechtfertigungsklage fristgemaess erhoben)
-- § 48/2 (Klage auf Feststellung des Nichtbestehens einer
--      Vormerkung)
-- § 61 (Einreichen der Loeschungsklage)
-- § 62 (Dauer des Klagerechts nach zivilrechtlichen
--      Bestimmungen ueber das Klagerecht)
-- § 63 (Klage auf Loeschung gegen Dritte)
-- § 64 (Fristverlaengerung fuer Klage wenn nicht
--      verstaendigt)
-- §§ 65 - 71 (Loeschungsklage)
-- § 72/1 (Anmerkung des Vollzuges der Zwangsversteigerung
--      von Amts wegen)
-- § 74 (Verweis auf LiegTeilG)
-- § 78 (ausserbuecherliches Recht)
-- § 79 (begehren der Eintragung des Rechtes eines Dritten)
-- § 81 (Zustellung?)
-- § 90 (Befreiung von Stempelgebuehren b.Abschriften)
```

-- fuer Urkundensammlung)

-- § 100 (Abweisung der Eintragung durch anderes Gericht -
-- Anmerkung von Amts wegen)

-- § 102 (schriftlicher Auftrag fuer Eintragung)

-- § 104/1 (Radieren verboten)

-- § 104/2-3 (Fehler bei Eintragung)

-- § 118-120 (Zustellung)

-- § 121 (nicht erfolgte Zustellung)

-- § 123 (Rekursfrist 30/60/90 Tage ab Zustellung)

-- § 126 (Verweis auf AussStrG)

-- § 132 (Einleitung des Verfahrens der Loeschung
-- gegenstandsloser Eintragungen)

-- § 134 (Verweis auf ausserStrG + Ergaenzungen)

-- § 137 (Inkrafttreten)

-- § 138 (Verweis auf Bergbuecher)

-- § 139 (Derogation von Vorschriften anderer Gesetze)

-- § 140 (Vollziehung)

-- Problem:

-- § 30/1 - Zustimmungen

-- § 34/1 - geringfuegige Grundbuchssachen

-- § 36 - hinglaengliche Bescheinigung von Forderung und
-- Rechtsgrund (Pfandrecht)

-- § 37 - hinglaengliche Bescheinigung von Bestand und
-- Einwilligung

-- § 43/1 - Fristverlaengerung bei Vormerkung

-- § 46/2 - Unterschied zu 45/1?

-- § 83 - Unterscheidung schriftl./muendl.Ansuchen

-- § 92 - Belegen von Grundbuchsgesuchen und
-- Halbschriften

-- § 94/1.1 - Hindernisse bei Eintragung aus anderen
-- Eintragungen

-- § 96/1 - momentanes Modell bewertet die Urkunde, nicht
-- ein Ansuchen

-- § 97 - Pruefen der Gegenbedingungen - momentan nur
-- moeglich, wenn ein einziges Dokument, dann
-- aber ohne Pruefung

-- § 124 - Instanzenzug

-- § 130 - Verstaendigung, Rekurs

-- § 131/2 - gegenstandslose Eintragungen

-- § 131/3-4 - Zustimmung des Eigentuemers bei
-- automatischem Vorgang???

```

-- § 133 - Vorauss. f.autom.Loeschung, Anmerkung
-- § 136/2 - Wahrung der Rechte Dritter bei Berichtigung
-- § 136/3 - Loeschung als Berichtigung

type GstNr          = String
type Loesch         = Bool
type Rechtfertigung = Bool
type Abweisung     = Bool

type Frist = Datum

type Adresse = String
type Stand  = String
data DetPerson = DetPers Person Adresse Stand [Person]
  -- Verstaendigungen

data AbZuschreibung = AZ Ort Datum [Person] Person Nr Nr
  [GstNr]
  deriving (Eq, Show)
  -- Ab-/Zuschreibung
  -- Ort: § 27/2

data Aenderung = Abschreibung Nr | Zuschreibung Nr
  deriving (Eq, Show)
data Koerper    = K [Nr] [Aenderung] deriving (Eq, Show)
data Eintragungstyp = Ev | Vo | An | AEv deriving (Eq, Show)
data Eintragung r t = Einverleibung (r t) Datum Loesch |
  Vormerkung (r t) Datum Frist Rechtfertigung
  Abweisung Loesch |
  -- Frist: § 43/1
  Anmerkung (r t) Datum Loesch
  deriving (Eq, Show) -- § 8

data Einlage r t = E Nr Koerper [Eintragung r t]
  deriving (Eq, Show)
-- Nr: § 85: Name mit der die Grundbuchseinlage im Grundbuch
-- aufscheint

```

```
data Hauptbuch r t = H [Einlage r t] deriving (Eq, Show)
```

```
data Grundbuch d r t = G (Hauptbuch r t) [d r t]  
  deriving (Eq, Show)
```

```

-- *****
-- Objekt-Klassen
-- *****

class AnsuchenDaten a where
  person      :: a -> Person
  adresse     :: a -> Adresse
  stand       :: a -> Stand
  zuVerstaendigen :: a -> [Person]

class Aenderungen a where
  abschreibung :: a -> Bool
  zuschreibung :: a -> Bool
  grundstueck  :: a -> Nr

class (Aenderungen Aenderung) => Koerpers k where
  koerpers      :: [Nr] -> [Aenderung] -> k -- § 2/2
  grundstuecke :: k -> [Nr]
  aenderungen  :: k -> [Aenderung]

  koerpAendern :: Aenderung -> k -> k -- § 3/2
  koerpAendern za k =
    if abschreibung za then
      if not (elem (grundstueck za) (grundstuecke k))
        then koerpers ((grundstueck za):grundstuecke k)
                  (za:aenderungen k)
        else k
    else
      if elem (grundstueck za) (grundstuecke k)
        then koerpers ((grundstuecke k) \\  

                      [(grundstueck za)])
                  (za:aenderungen k)
        else k

  toDelK :: k -> Bool
  toDelK = ((0==).length.grundstuecke) -- § 3/3

```

```

class (Rechte r t) => Eintragungen e r t where
  einverleibung      :: r t -> Datum -> Bool -> e r t
  vormerkung        :: r t -> Datum -> Datum -> Bool -> Bool
                    -> Bool -> e r t
  anmerkung         :: r t -> Datum -> Bool -> e r t
  eintragRecht      :: e r t -> r t -- § 5
  eintragDatum      :: e r t -> Datum
  eintragGeloescht  :: e r t -> Bool
  eintragAbgewiesen:: e r t -> Bool
  istEinverleibung  :: e r t -> Bool
  istVormerkung     :: e r t -> Bool
  istEintrAnm       :: e r t -> Bool
  eintragRechtf     :: e r t -> Bool
  eintragFrist      :: e r t -> Datum
  eintragLoeschen   :: e r t -> e r t

  eigentumsAnteil   :: (Rechte r t) => e r t -> Anteil
  eigentumsAnteil   = anteil.rechtstyp.eintragRecht

  eigentumVon       :: (Rechte r t) => [e r t] -> Person
                    -> Anteil

  eigentumVon es p = foldl (+) (zero) (map eigentumsAnteil
                                         (filter f es))

  where f e = istEinverleibung e &&
            elem p ((berechtigte.eintragRecht) e)

  vormLoeschenErlaubt :: Datum -> e r t -> Bool
  vormLoeschenErlaubt dt e = (((dt >).eintragFrist) e) &&
                              ((not.gerechtfertigt) e)
                              -- § 45/1

```

```

eintragungLoeschen :: e r t -> e r t
eintragungLoeschen e
  | istEinverleibung e = einverleibung (eintragRecht e)
                        (eintragDatum e) (eintragGegen e)
  | istVormerkung     e = vormerkung (eintragRecht e)
                        (eintragDatum e)
                        (eintragFrist e)
                        (eintragRechtf e)
                        (eintragAbgewiesen e) True
  | istEintrAnm       e = anmerkung (eintragRecht e)
                        (eintragDatum e)
                        (eintragGegen e)
  | otherwise =
    error "Der uebergegebene Datensatz ist keine Eintragung"

gerechtfertigt :: e r t -> Bool
gerechtfertigt e = (not.istVormerkung) e || eintragRechtf e

doRechtfertigen :: e r t -> e r t -- § 46/1
doRechtfertigen e =
  if (istEinverleibung e) || (istEintrAnm e) then e
  else vormerkung (eintragRecht e) (eintragDatum e)
                (eintragFrist e) (True) (eintragAbgewiesen e)
                (eintragGeloescht e)

delVO :: Datum -> [e r t] -> [e r t]
delVO _ [] = []
delVO dt (e:es) = if istVormerkung e &&
                  vormLoeschenErlaubt dt e
                  then delVO dt es else e:delVO dt es

delRO :: Datum -> [e r t] -> [e r t]
-- loeschen der Rangordnung, wenn nicht innerhalb eines
-- Jahres verwendet (§ 55)
delRO dt el = del' el where
  del' (e:es) =
    if istEintrAnm e && (istRangordnung.eintragRecht) e &&
      notElem ((roNummer.eintragRecht) e)
      ((map roNummer.filter(not.istRangordnung).
        map eintragRecht) el) &&
      incTage (eintragDatum e) 365 > dt
    then eintragungLoeschen e:del' es
    else e:del' es

```

```

vorgemerktetesEigentum :: (Rechte r t) => [e r t] -> Bool
vorgemerktetesEigentum [] = False
vorgemerktetesEigentum (e:es) =
    (istVormerkung e && (istEigentum.eintragRecht) e) ||
    vorgemerktetesEigentum es

voDel :: [e r t] -> [Person] -> Datum -> [e r t]
-- notwendig fuer § 49/1-4
-- Loeschen von Vormerkungen, die nach einem angegebenen
-- Datum gegen eine bestimmte Personen eingetragen wurden
voDel [] _ _ = []
voDel (e:es) ps dt = if (istVormerkung e) &&
    ((eintragDatum e) > dt) &&
    (((sort.belastete.eintragRecht) e)
        == sort ps)
    then voDel es ps dt else e:voDel es ps dt

recht2Eintrag :: (Eq r t) => r t -> [e r t] -> e r t
recht2Eintrag r = head.(filter ((r==).eintragRecht))

loeschvb :: r t -> Datum -> Datum -> [e r t] -> [e r t]
loeschvb r v b es = map f es
    where f e = if ((eintragDatum e) >= v) &&
        ((eintragDatum e) <= b) &&
        ((eintragRecht e) /= r)
        then eintragungLoeschen e else e

eintragungRechtmaessig :: e r t -> Bool -- ntw. fuer § 130
eintragungRechtmaessig _ = True -- nur fuer das Modell
gegenstandslos :: e r t -> Bool -- § 131/2
gegenstandslos e = ((betrag.eintragRecht) e /= 0) &&
    ((betrag.eintragRecht) e < 1500)
    -- nur so definiert, um das Modell lauffaehig zu machen

illegaleEintragungen :: [e r t] -> [e r t]
illegaleEintragungen = map f
    where f e = if eintragungRechtmaessig e then e
        else eintragLoeschen e

autoLoeschen :: [e r t] -> [e r t] -- § 131/1
autoLoeschen es = map f es
    where f e = if gegenstandslos e then eintragLoeschen e
        else e

```

```

class (Koerpers Koerper, Eintragungen Eintragung r t,
      Eq (r t)) => Einlagen e r t where
einlagen    :: Nr -> Koerper -> [Eintragung r t] -> e r t
            -- § 2/2, 3/1
koerper     :: e r t -> Koerper
eintrliste  :: e r t -> [Eintragung r t]
einlagezahl:: e r t -> Nr

eintragen   :: e r t -> Eintragung r t -> e r t
eintragen einlage e =
  if (istVormerkung e) && eintragExists &&
    (eintragGeloescht vormerk) &&
    ((not.eintragAbgewiesen) vormerk)
-- § 47, 48/1
-- gerechtfertigt abgewiesen geloescht      bedeutet
--   True           False   True =>Recht wurde normal
--                                   geloescht
--   False          True    True =>Recht wurde
--                                   abgewiesen
--   False          False   True =>Recht wurde nicht
--                                   gerechtfertigt
-- neuerliche Eintragung im 2. Fall nicht moeglich
  then einlage
  else
    if ((istLoeschung.eintragRecht) e) &&
      (istEinverleibung e)
      then einlagen (einlagezahl einlage)
        (koerper einlage)
        (e:(loesch (recht2Eintrag
                    (betrRecht (eintragRecht e))
                    (eintrliste einlage))
              (eintrliste einlage)))
      else einlagen (einlagezahl einlage)
        (koerper einlage)
        (e:eintrliste einlage)
where vormerk = (eintrliste einlage) !! untag
              (elemIndex (eintragRecht e) rs)
rs           = map eintragRecht (eintrliste einlage)
eintragExists = elem (eintragRecht e) rs
untag (Just a) = a
loesch sz [] = []
loesch sz (z:zs) = if sz == z

```

```

                                then eintragungLoeschen z:zs
                                else z:loesch sz zs

toDelE      :: e r t -> Bool
toDelE = toDelK.koerper

rechtfRecht :: (Rechte r t) => e r t -> r t -> Datum
           -> e r t
rechtfRecht e r d = einlagen (einlagezahl e) (koerper e)
                  (voLoeschen (map rf' (eintrliste e)))
where rf' et = if (istVormerkung et) &&
                ((not.gerechtfertigt) et) &&
                ((eintragFrist et) >= d)
                then doRechtfertigen et else et
voLoeschen es =
    if istEigentum r then voDel es (belastete r)
    ((eintragDatum.(recht2Eintrag r)) es)
    else es

autoEinl      :: Datum -> e r t -> e r t
autoEinl dt e = einlagen (einlagezahl e) (koerper e)
                ((autoLoeschen.illegaleEintragungen.delRO dt.
                  delVO dt) (eintrliste e))

class (Einlagen Einlage r t) => Hauptbuecher h r t where
  hauptbuecher :: [Einlage r t] -> h r t -- § 2/1
  gbEinlagen   :: h r t -> [Einlage r t]

  autoDelEinl :: h r t -> h r t -- § 3/3
  autoDelEinl h = hauptbuecher (filter (not.toDelK.koerper)
                                (gbEinlagen h))

  eintragenHB :: h r t -> Eintragung r t -> [Nr] -> h r t
  eintragenHB h et ezs =
    hauptbuecher (map eintr' (gbEinlagen h)) where
      eintr' einlage = if elem (einlagezahl einlage) ezs
                        then eintragen einlage et else einlage

  simultanhypothek :: h r t -> Eintragung r t -> [Nr]
                  -> h r t -- § 15/1

```

```

simultanhypothek h e ezs =
    hauptbuecher (map eintrag (gbEinlagen h))
  where eintrag ez = if elem (einlagezahl ez) ezs
                    then eintragen ez e
                    else ez

rechtfHB :: (Rechte r t) => h r t -> Nr -> r t -> Datum
          -> h r t
rechtfHB h ez r d = hauptbuecher (map rf' (gbEinlagen h))
  where rf' et = if (einlagezahl et) == ez
                 then rechtfRecht et r d else et

autoHB :: Datum -> h r t -> h r t
autoHB dt h = hauptbuecher (map (autoEinl dt)
                                (gbEinlagen (autoDelEinl h)))

loeschNRO :: r t -> Nr -> Datum -> Datum -> h r t -> h r t
loeschNRO r ez v b h = hauptbuecher (map f (gbEinlagen h))
  where f e = if (einlagezahl e) == ez
               then einlagen ez (koerper e)
               ((loeschvb r v b.eintrliste) e)
               else e

class GBTeile g d r t where
  grundbuecher      :: Hauptbuch r t -> [d r t] -> g d r t
  hauptbuch        :: g d r t -> Hauptbuch r t -- § 1
  urkundensammlung :: g d r t -> [d r t]         -- § 1, 6/2

class (GBTeile g d r t, Dokumente d r t,
       Hauptbuecher Hauptbuch r t, Rechte r t, Ord (r t),
       Ord (Eintragung r t)) => Abschriften g d r t where
  abfrageErlaubt :: g d r t -> Person -> Bool
  abfrageErlaubt g _ = True -- § 7/1, 7/2

  eintragungenVEinl :: g d r t -> Nr -> Person
                    -> [Eintragung r t]

  eintragungenVEinl g ez p =
    if (abfrageErlaubt g p)
    then (map chg.sort.eintrliste.rd.gbEinlagen.hauptbuch) g
        -- § 30/2
    else error "Abfrage nicht erlaubt"
  where rd = head.filter ((ez ==).einlagezahl)

```

```

    chg ein = if (istVormerkung ein) &&
                (gerechtfertigt ein)
                then einverleibung (eintragRecht ein)
                    (eintragDatum ein)
                    (eintragGeloescht ein)
                else ein

rechteVon :: g d r t -> Nr -> Person -> [r t]
rechteVon g ez = rangsort.map eintragRecht.
                filter (not.eintragGeloescht).eintragungenVEinl g ez
-- § 30/2

eigentuemervon :: g d r t -> Nr -> [Person]
eigentuemervon g ez = foldl (++) []
                    (map berechnete eigentumsrechte)
    where eigentumsrechte = filter istEigentum
                    (rechteVon g ez "Amt")

class (GBTeile g d r t, Abschriften g d r t, Dokumente d r t,
      Hauptbuecher Hauptbuch r t) => Grundbuecher g d r t
where
    berichtigen :: g d r t -> Eintragungstyp -> d r t
                -> Datum -> g d r t -- § 136/1
    berichtigen g art d dt =
        if elem ((erstellung.dokumentTyp) d) [notar,gericht]
        then grundbuecher (eintragenHB (hauptbuch g)
                                (Einverleibung
                                 (recht d) dt True) (liegenschaften d))
                                (d:urkundensammlung g)
        else g

eintragenGB :: (Rechte r t, Ord (r t)) =>
              g d r t -> Eintragungstyp -> d r t
              -> Datum -> g d r t
    -- Datum: Zeitpunkt des Einlagens beim Grundbuchsgericht
    -- § 95/1: keine Einvernehmung, keine Zwischenerledigung
eintragenGB g Ev d dt =
    -- § 75: Einlage bei diesem GB abgelegt
    if (istLoeschung.recht) d then
        if (beglaubigt (eigenschaften d)) && (dokumentOK d) &&
            -- Zustimmungen ok:
            (belastete.recht) d ==

```

```

(berechtigte.betrRecht.recht) d then
  if (spezErfordernisse d) -- § 35
    then grundbuecher (eintragenHB (hauptbuch g)
      (Einverleibung (recht d) dt True)
      (liegenschaften d)) (d:urkundensammlung g)
      -- § 26/1
    else eintragenGB g Vo d dt
  else anmAbweisung g d dt -- § 99
else
  if (beglaubigt (eigenschaften d)) &&
    (dokumentOK d) && (eintragErlaubt d
      (nub (foldl (++) []) (map (eigentuemerVon g)
        (liegenschaften d)))))) then -- § 21
    if (spezErfordernisse d) then -- § 35
      if (istSimultan.recht) d
        -- Zusätzliche Anmerkung notwendig!
      then eintragenGB
        (grundbuecher (eintragenHB (hauptbuch g)
          (Einverleibung (recht d) dt False)
          (liegenschaften d))
          (d:urkundensammlung g))
          An anmSim dt
        else grundbuecher (eintragenHB (hauptbuch g)
          (Einverleibung (recht d) dt False)
          (liegenschaften d))
          (d:urkundensammlung g)
        else eintragenGB g Vo d dt -- § 85/3
        else anmAbweisung g d dt -- § 99
    where anmSim = makeAnmSimultan (eigentuemerVon g
      (head (liegenschaften d))) (liegenschaften d)
  eintragenGB g t d dt =
    if (beglaubigt.eigenschaften) d && (dokumentOK d) &&
      (eintragErlaubt d (nub (foldl (++) [])
        (map (eigentuemerVon g) (liegenschaften d)))))) then
-- § 21, § 52
    if (istSimultan.recht) d
      -- Zusätzliche Anmerkung notwendig!
    then eintragenGB
      (grundbuecher ((eintragenHB (hauptbuch g)
        (rechtseintrag t) (liegenschaften d)))
        (d:urkundensammlung g))

```

```

        An anmSim dt
    else grundbuecher (eintragenHB (hauptbuch g)
                        (rechtseintrag t) (liegenschaften d))
                        (d:urkundensammlung g)
    else anmAbweisung g d dt -- § 99
where rechtseintrag AEv =
        Einverleibung (recht d) dt False
        -- § 85/3 - Vormerkung ausgeschlossen
rechtseintrag Vo =
        Vormerkung (recht d) dt (incTage dt 14)
        False False False
        -- § 26/1
        -- § 96/2 keine Umwandlung in Ev moeglich!

rechtseintrag An = Anmerkung (recht d) dt False
anmSim = makeAnmSimultan (eigentuemerVon g
                          (head (liegenschaften d))) (liegenschaften d)
anmAbweisung :: (Rechte r t, Ord (r t)) =>
                g d r t -> d r t -> Datum -> g d r t
anmAbweisung g d dt =
    if (beglaubigt.eigenschaften) d && (dokumentOK d)
    then g
    else eintragenGB g An
        (makeAbweisung (recht d) (liegenschaften d)
          nub (foldl (++) [] (map (eigentuemerVon g)
                                   (liegenschaften d))))
        dt

rangordnung :: (Ord (r t)) => Datum -> Nr -> g d r t
              -> (RONr,g d r t)
-- § 53, 54
rangordnung dt nr g = ((roNummer.recht) ro,gb)
    where ro = makeRO (eigentuemerVon g nr) nr
              (urkundensammlung g)
          gb = eintragenGB g An ro dt

rechtfertigen :: (Rechte r t) => g d r t -> Nr -> d r t
              -> Datum -> g d r t
-- § 41

```

```

rechtfertigen g ez d dt = grundbuecher (rechtfHB
                                     (hauptbuch g) ez
                                     (recht d) dt)
                                     (d:urkundensammlung g)

automatisch :: Datum -> g d r t -> g d r t
  -- § 76: nur Ausnahmefaelle
automatisch dt g = grundbuecher (autoHB dt (hauptbuch g))
                               (urkundensammlung g)

loeschenNachRangordn :: (Ord (r t), Ord (Eintragung r t),
                        Rechte r t) =>
                        Datum -> (r t,Datum) -> Nr -> g d r t
  --
  --      aktDatum,(Partei,Recht,EintrDatum),EZ
  --      -> g d r t
  -- § 57: Loeschen aller Eintragungen, die zwischen
  --      Anmerkung der Rangordnung und Verwendung der
  --      Rangordnung fuer eine Veraeusserung eingetragen
  --      wurden.
loeschenNachRangordn ad (r,ed) nr g =
  if (istEigentum r) && (ad <= (incTage ed 14)) then
    grundbuecher (loeschNRO r nr von bis (hauptbuch g))
                  (nd:urkundensammlung g)
  else g
where nd    = makeLoeschVB von bis (berechtigte r) nr
      von   = eintragDatum ro
      bis   = (eintragDatum.recht2Eintrag r) es
      es    = eintragungenVEinl g nr ""
      roNr  = (roNummer.eintragRecht.recht2Eintrag r) es
      ro    = (head.filter ((roNr ==).roNummer.
                           eintragRecht)) es

```

```

-- *****
-- Instanzen
-- *****

instance Ord (Recht RTyp) where
  compare r1 r2 = EQ

--instance Ord (Eintragung Recht RTyp) where
--  compare r1 r2 = compare (eintragDatum r1)
--                        (eintragDatum r2)

instance (Ord (r t), Eintragungen Eintragung r t) =>
  Ord (Eintragung r t) where
  compare e1 e2 = compare (eintragDatum e1) (eintragDatum e2)
  -- § 29/1: Bestimmt Rangordnung

instance AnsuchenDaten DetPerson where
  person      (DetPers p a s ps) = p
  adresse     (DetPers p a s ps) = a
  stand       (DetPers p a s ps) = s
  zuVerstaendigen (DetPers p a s ps) = ps

instance Aenderungen Aenderung where
  abschreibung (Abschreibung _) = True
  abschreibung _                 = False
  zuschreibung (Zuschreibung _) = True
  zuschreibung _                 = False
  grundstueck (Abschreibung n) = n
  grundstueck (Zuschreibung n) = n

instance Koerpers Koerper where
  koerpers nrs aends = K nrs aends
  grundstuecke (K nrs aends) = nrs
  aenderungen (K nrs aends) = aends

```

```

instance Eintragungen Eintragung Recht RTyp where
  einverleibung r d g = Einverleibung r d g
  vormerkung r d f rf a g = Vormerkung r d f rf a g
  anmerkung r d g = Anmerkung r d g
  eintragRecht      (Einverleibung r d g) = r
  eintragRecht      (Vormerkung    r d f rf a g) = r
  eintragRecht      (Anmerkung    r d g) = r
  eintragDatum      (Einverleibung r d g) = d
  eintragDatum      (Vormerkung    r d f rf a g) = d
  eintragDatum      (Anmerkung    r d g) = d
  eintragFrist      (Vormerkung    r d f rf a g) = f
  eintragFrist      _ = Datum 0 0 0
  eintragGeloescht  (Einverleibung r d g) = g
  eintragGeloescht  (Vormerkung    r d f rf a g) = g
  eintragGeloescht  (Anmerkung    r d g) = g
  istEinverleibung (Einverleibung r d g) = True
  istEinverleibung _ = False
  istVormerkung    (Vormerkung    r d f rf a g) = True
  istVormerkung    _ = False
  istEintrAnm     (Anmerkung    r d g) = True
  istEintrAnm     _ = False
  eintragRechtf    (Vormerkung    r d f rf a g) = rf
  eintragRechtf    _ = False
  eintragAbgewiesen (Vormerkung    r d f rf a g) = a
  eintragAbgewiesen _ = False
  eintragLoeschen  (Einverleibung r d _) =
    (Einverleibung r d True)
  eintragLoeschen  (Vormerkung r d f rf a _) =
    (Vormerkung r d f rf a True)
  eintragLoeschen  (Anmerkung r d _) = (Anmerkung r d True)

```

```

instance Einlagen Einlage Recht RTyp where
  einlagen nr k es = E nr k es
  koerper (E nr k es) = k
  eintrliste (E nr k es) = es
  einlagezahl (E nr k es) = nr

```

```
instance Hauptbuecher Hauptbuch Recht RTyp where
  hauptbuecher es    = H es
  gbEinlagen (H es) = es
```

```
instance GBTeile Grundbuch Dokument Recht RTyp where
  grundbuecher h u      = G h u
  hauptbuch      (G h u) = h
  urkundensammlung (G h u) = u
```

```
instance Abschriften Grundbuch Dokument Recht RTyp
```

```
instance Grundbuecher Grundbuch Dokument Recht RTyp
```


Curriculum Vitae



Name: Gerhard Navratil
(Eltern: Walter und Elfriede Navratil)
Geburtsdatum: 25. August 1969
Geburtsort: Wien
Familienstand: Ledig

Bildungsweg:

1975 – 1979 Volksschule Wien
1979 – 1981 BRG Wien VII Kandlgasse
1981 – 1987 BRG Laa a.d. Thaya
Juni 1987 Ablegung der Matura
1987 – 1989 Studium Elektrotechnik
1988 – 1989 Studium Informatik
1989 – 1998 Studium Vermessungswesen
Diplomarbeit: *An object-oriented model of a cadaster*
Juni 1998 Ablegung der Diplomprüfung
1998 – 2002 Doktoratsstudium
Dissertation: *Formalisierung von Gesetzen am Beispiel des Österreichischen allgemeinen Grundbuchgesetzes*

Studienbegleitende Tätigkeiten:

August 1990	Messhelfer beim Vermessungsamt Korneuburg
August 1991	Messhelfer beim Vermessungsamt Wr. Neustadt
1992 - 1995	Ferialpraktikant bei Zivilingenieur Dipl.-Ing. Swatschina jeweils von Anfang Juli bis Mitte September
1994 – 1995	Tutor am Institut für Landesvermessung und Ingenieur- geodäsie/Abt.Geoinformation und Landesvermessung für die Übungen ‚Geodätisches Zeichnen‘ und ‚Introduction to GIS‘

Sonstige Tätigkeiten

Jänner – August 1988 Präsenzdienst in Wöllersdorf

Beruflicher Werdegang:

1998 - 2002	Universitätsassistent am Institut für Landesvermessung und Ingenieurgeodäsie/Abt. Geoinformation und Landesvermes- sung (später Institut für Geoinformation und Landesvermes- sung)
Herbst 2001	Als Experte für Meixner Consulting Engineers in Zypern im Rahmen des EU-Consulting-Projektes ‚ <i>Mapping and Property Valuation – Technical Assistance in Computerised Valuation</i> ‘ (CYP/03/600/015/A1)