

Fachbereich .../...
Studiengang ...

Abschlussarbeit

Der Titel dieser
Arbeit

Vorname Nachname
Matrikelnr.: ...

1. Januar 1970

1. Gutachter: Prof. Dr. Vorname Nachname
Arbeitsgruppe ... (AG ...), Universität Musterstadt
2. Gutachter: Prof. Dr. Vorname Nachname
Arbeitsgruppe ... (AG ...), Universität Musterstadt
- Betreuung: Vorname Nachname
Arbeitsgruppe ... (AG ...), Universität Musterstadt

Erklärung

Ich versichere, diese Arbeit ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Musterstadt, 1. Januar 1970

(Vorname Nachname)

Zusammenfassung

Hier die deutschsprachige Zusammenfassung...

Abstract

Hier ggf. die englischsprachige Zusammenfassung...

Danksagung

Mein Dank gilt...

Außerdem bedanke ich mich bei...

Bedanken möchte ich mich auch bei...

Musterstadt, im Januar 1970

Vorname Nachname

*Hier kommt ein passendes Zitat
oder z. B. eine Widmung hin.*

EUGENE C. FREUDER, CONSTRAINTS, APRIL 1997

Inhaltsverzeichnis

Abbildungsverzeichnis	xii
Tabellenverzeichnis	xiii
Verzeichnis der Definitionen	xiv
Verwendete Symbole	xv
I Anwendungsdomäne und Problemstellung	1
1 Einleitung	3
1.1 Motivation	3
1.2 Angestrebte Ergebnisse	3
1.3 Aufbau der Arbeit	3
2 Beschreibung der Anwendungsdomäne	5
2.1 Einordnung	5
2.2 Methoden	6
2.2.1 Erste Methode	6
2.2.2 Zweite Methode	6
2.2.3 Dritte Methode	7
2.3 Systeme	7
2.4 Das zu betrachtende System	8
2.4.1 Erste Komponente	8
2.4.2 Zweite Komponente	8
2.4.3 Dritte Komponente	8
2.4.4 Diskussion zur betrachteten Komponente	8
2.5 Anforderungen	9
II Grundlagen	11
3 Stand der Technik	13
3.1 Einführung	13

3.2	Konzepte	13
3.2.1	Erstes Konzept	13
3.2.2	Zweites Konzept	14
3.2.3	Drittes Konzept	14
3.3	Algorithmen	15
3.3.1	Erster Algorithmus	15
3.3.2	Zweiter Algorithmus	15
3.3.3	Dritter Algorithmus	16
3.4	Verfügbare Systeme	16
3.4.1	Erstes System	16
3.4.2	Zweites System	16
3.4.3	Drittes System	16
3.5	Zusammenfassung und Diskussion	16
III	Konzeption und Realisierung	19
4	Konzept	21
4.1	Einführung	21
4.2	Erster Bereich	21
4.3	Zweiter Bereich	21
4.4	Zusammenführung und Weiterentwicklung	21
4.5	Systemarchitektur	22
4.6	Diskussion	22
5	Implementierung	23
5.1	Einleitung	23
5.2	Eingesetzte Konzepte	23
5.3	Übersicht über die Packages	23
5.4	Erste Komponente	23
5.5	Zweite Komponente	24
5.6	Dritte Komponente	24
5.7	Ausnahmebehandlung	24
5.8	Integration	24
5.8.1	Anbindung an das betrachtete System	24
5.8.2	Integration in andere Systeme	24
6	Validierung	25
6.1	Einleitung	25
6.2	Eigenschaften des Prototypen	25
6.3	Validierung anhand synthetischer Probleme	26
6.3.1	Übersicht über die Problemstellungen	26
6.3.2	Vorbereitende Maßnahmen	26
6.3.3	Ergebnisse	26

6.3.4	Zusammenfassung	26
6.4	Validierung im Praxiseinsatz	26
6.4.1	Vorbereitende Maßnahmen	26
6.4.2	Ergebnisse	26
6.4.3	Zusammenfassung	26
6.5	Vergleich mit weiterführenden Ansätzen	26
6.6	Zusammenfassung	26
7	Zusammenfassung und Ausblick	27
7.1	Zusammenfassung	27
7.2	Ausblick	27
IV	Anhänge	29
A	Installation	31
A.1	Abschnittsüberschrift	31
A.1.1	Unterabschnittsüberschrift	31
B	Programm	33
C	API-Dokumentation	37
C.1	Package	37
C.1.1	Interface	37
C.1.2	Klasse	38
D	Glossar	39
	Literaturverzeichnis	41
	Abkürzungsverzeichnis	43
	Stichwortverzeichnis	47

Abbildungsverzeichnis

2.1	Einordnung der Konfigurierung	6
2.2	Regel aus dem XCON-System	7
2.3	Wurzelkonzept für eine PC-Konfiguration	9
3.1	Aufwand von Problemreduktion vs. Suchaufwand	14
3.2	Der Kantenkonsistenz-Algorithmus AC-3	15

Tabellenverzeichnis

3.1	Klassifizierung von Suchstrategien	16
3.2	Beispiel für Forward Checking	16

Verzeichnis der Definitionen

2.1.1 Konstruktionssystem	5
3.2.1 Constraint-Erfüllung	13

Verwendete Symbole

P	Constraint Satisfaction Problem (CSP)
C	Constraint
C^u	unäres Constraint
C^b	binäres Constraint
R	Relation
R^u	unäre Relation
R^b	binäre Relation
$R_{i,j}$	Relation zwischen den Variablen v_i und v_j
v	Constraint-Variable
D	Wertebereich einer Constraint-Variable (Domäne)
$D_1 \times \dots \times D_n$	kartesisches Produkt der Wertebereiche D_1 bis D_n
d	Variablenwert aus der Domäne einer Constraint-Variable $d \in D$
\vec{a}	Teilbelegung der Variablen eines CSP
I	Intervall
$[a_1, a_2]$	abgeschlossenes Werteintervall mit a_1 als untere und a_2 als obere Schranke
$]a_1, a_2[$	offenes Werteintervall mit a_1 als untere und a_2 als obere Schranke
x^+	innerhalb der Präzisionsgrenzen die kleinste darstellbare Zahl $> x$
x^-	innerhalb der Präzisionsgrenzen die größte darstellbare Zahl $< x$
\mathbb{R}	Menge der reellen Zahlen
$I(\mathbb{R})$	Menge der beschränkten, abgeschlossenen, reellen Intervalle (ohne $-\infty$ und $+\infty$)
$I^*(\mathbb{R})$	Menge der unbeschränkten, abgeschlossenen, reellen Intervalle (inkl. $-\infty$ und $+\infty$)
$L^*(\mathbb{R})$	Menge der reellwertigen, unteren Grenzen (inkl. $-\infty$ und $+\infty$)
$U^*(\mathbb{R})$	Menge der reellwertigen, oberen Grenzen (inkl. $-\infty$ und $+\infty$)
B^\square	Box
$\Phi_{kB}(P)$	P ist kB -konsistent
$\Phi_{Box}(P)$	P ist Box-konsistent

Teil I

Anwendungsdomäne und Problemstellung

Kapitel 1

Einleitung

*Zu jedem komplexen Problem gibt es eine verblüffend triviale Lösung,
die sich einfach erklären lässt und vollkommen falsch ist.*

ANONYM

An dieser Stelle wird neben einer Motivation zum Thema ein Überblick über die zu erreichenden Ziele sowie eine Übersicht über den Aufbau der vorliegenden Arbeit gegeben. [...]

1.1 Motivation

Das ... ist ein stetig wachsender Bereich, dem insbesondere in Bezug auf ... eine immer größere Bedeutung beigemessen wird. [...]

1.2 Angestrebte Ergebnisse

Das Ziel dieser Arbeit besteht in [...]

1.3 Aufbau der Arbeit

Der Aufbau dieser Arbeit gliedert sich in die folgenden Abschnitte: In **Kapitel 2** erfolgt eine Einführung in ... In **Kapitel 3** wird ein detaillierter Überblick über ... gegeben. [...] In **Kapitel 4** wird das Konzept für ... entwickelt. In **Kapitel 5** wird die Implementierung von ... dargelegt. Es werden der Reihe nach die enthaltenen Komponenten aufgezählt und dokumentiert. Abschließend erfolgt die Beschreibung der Integration von ... in ... [...] In **Kapitel 6** folgt eine Validierung der Umsetzung anhand der realisierten Funktionalität und der zuvor benannten Anforderungen. Neben synthetischen Problemstellungen wird die Integration von ... in ... validiert. Außerdem erfolgt eine Positionierung im Vergleich mit

weiterführenden Arbeiten. In **Kapitel 7** erfolgt eine Zusammenfassung sowie ein kurzer Ausblick hinsichtlich der Erweiterungsmöglichkeiten der Konzeption und des entwickelten Prototypen.

Neben den genannten Kapiteln existieren in dieser Arbeit eine Reihe von Anhängen: In **Anhang A** sind die ausführlichen Informationen über ... zu finden, die im Rahmen dieser Arbeit recherchiert und für Kapitel 3 zusammengefasst wurden. Der **Anhang A** enthält eine Beschreibung der Installation des Prototypen, die aus Gründen der Übersichtlichkeit an dieser Stelle platziert wurde. Beispiele für ... sind in **Anhang B** dokumentiert. [...] Die vollständige API-Dokumentation für das entwickelte System mit einer Beschreibung der entwickelten Klassen und Methoden befindet sich in **Anhang C**.

Für das bessere Verständnis werden gebräuchliche englischsprachige Begriffe als *terminus technicus* beibehalten und nicht ins Deutsche übersetzt. Ein Glossar, in dem wichtige Begriffe erläutert werden, befindet sich in **Anhang D**. Geschützte Warennamen und Warenzeichen sind innerhalb der Arbeit nicht gesondert kenntlich gemacht. Aus dem Fehlen eines solchen Hinweises kann demnach nicht geschlossen werden, dass es sich um einen freien Warennamen oder ein freies Warenzeichen handelt.

Kapitel 2

Beschreibung der Anwendungsdomäne

*Ein einleitendes Zitat
für dieses Kapitel.*

VORNAME NACHNAME

Im Folgenden wird eine Einführung in ... und ein Überblick über ... gegeben. [...]

2.1 Einordnung

Als erstes folgen in diesem Abschnitt einige Beispiele für Literaturzitate, das Einbinden einer Abbildung sowie für eine Definition:

Den Syntheseaufgaben lassen sich, wie in Abbildung 2.1 auf der nächsten Seite dargestellt, *Konstruktionsaufgaben* zuordnen, die wiederum in *Konfigurierungsaufgaben* und *Planungsaufgaben* unterteilt werden (vgl. Günter 1991, S. 1). Konstruktionsaufgaben sind dadurch gekennzeichnet, dass im Verlauf der Konstruktion aus einer Reihe von Bauteilen bzw. Planschritten ein Gesamtkonzept, eine *Konfiguration* bzw. ein *Plan*, erstellt wird (vgl. Neumann 1991, S. 12):

Definition 2.1.1 (Konstruktionssystem)

Ein Konstruktionssystem ist ein Expertensystem, das beim Entwurf von Aggregaten aus Komponenten hilft und dabei Zielvorgaben sowie Expertenwissen verwendet.

Es folgt ein Beispiel für eine Aufzählung:

- Erster Punkt,
- Zweiter Punkt,

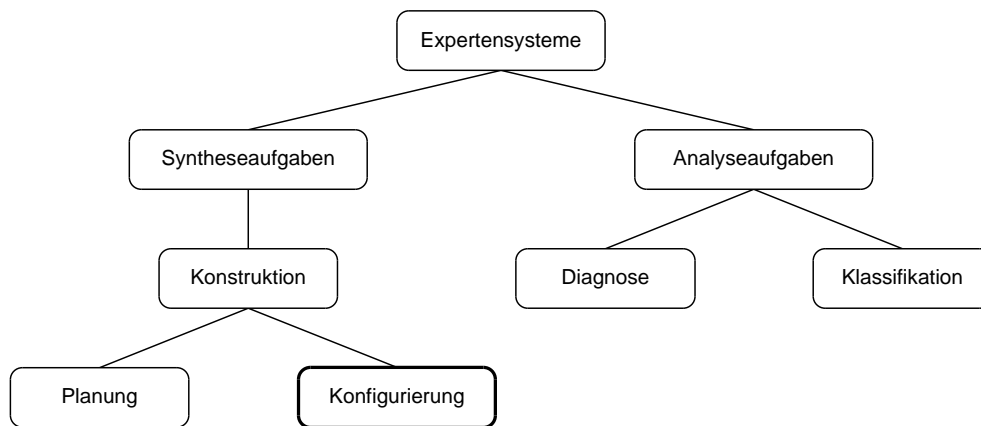


Abbildung 2.1: Einordnung der Konfigurierung

- Dritter Punkt.

Wenn der nachfolgende Absatz ohne Zeileneinschub erfolgen soll, kann dies mit `\noindent` erreicht werden. Mehrere Autoren Seitengenau zitieren ist mit den hier genutzten Anweisungen möglich (vgl. Günter 1991, S. 1; Neumann 1991, S. 12 f.).

2.2 Methoden

Sehr schlechter Stil ist es, wenn Überschriften verwendet werden, unter denen sich *kein* Text, sondern direkt die nächste Überschrift der nachfolgenden Gliederungsstufe befindet. Im Zweifelsfall sollte man wenigstens ein oder zwei Sätze einfügen, die auf die jeweiligen Inhalte der nachfolgenden Unterabschnitte hinweisen.

2.2.1 Erste Methode

Ein Beispiel für das Auszeichnen in nichtproportionaler Schriftart:

```
IF <Bedingungen> THEN DO <Aktionen>
```

Der nachfolgende Absatz je nach Wunsch wieder ohne Einrückung. Normale Absätze sollten für eine bessere Lesbarkeit allerdings wieder Eingedrückt werden. Dies ist auch die Standardeinstellung.

2.2.2 Zweite Methode

Fußnoten werden wie gewohnt erstellt.¹ Beachtet werden muss dabei die äußere Umklammerung von `\footnote` mit `{...}`, damit die HTML-Darstellung fehlerfrei erfolgt.

¹Hier kommt eine Fußnote hin.

```
ASSIGN-POWER-SUPPLY-1
```

```
IF:
```

```
THE MOST CURRENT ACTIVE CONTEXT IS ASSIGNING A POWER SUPPLY  
AND AN SBI MODULE OF ANY TYPE HAS BEEN PUT IN A CABINET  
AND THE POSITION IT OCCUPIES IN THE CABINET IS KNOWN  
AND THERE IS SPACE IN THE CABINET FOR A POWER SUPPLY  
AND THERE IS NO AVAILABLE POWER SUPPLY  
AND THE VOLTAGE AND FREQUENCY OF THE COMPONENTS IS KNOWN
```

```
THEN:
```

```
FIND A POWER SUPPLY OF THAT VOLTAGE AND FREQUENCY  
AND ADD IT TO THE ORDER
```

Abbildung 2.2: Regel aus dem XCON-System (vgl. Neumann 1991, S. 16)

Abkürzungen wie d. h. und z. B. sollten mit speziellen Auszeichnungen notiert werden, um mit \LaTeX die korrekten Abstände zwischen den einzelnen Zeichen und zu nachfolgenden Wörtern zu erreichen.

2.2.3 Dritte Methode

In dem nachfolgenden Absatz sind einige Indexmarken gesetzt. Außerdem wird für den Namen des genannten Systems ein eigenes Makro genutzt, um eine einheitliche Darstellung zu erreichen:

Der erste und wohl bekannteste Vertreter der Gattung der regelbasierten Expertensysteme ist das System R1/XCON. XCON war lange Zeit ein sehr erfolgreiches System zur Konfigurierung von VAX-Rechnern der Firma Digital Equipment Corporation (DEC) (vgl. Stumptner 1997, S. 111). Ein Beispiel für die Beschreibung einer Regel aus dem XCON-System ist in Abbildung 2.2 zu sehen.

Die Abbildung 2.2 ist ein Beispiel für Quelltext in nichtproportionaler Schriftart, der mit einem Rahmen umgeben wird. Dieser Rahmen wird sowohl in der DVI-, PS- und PDF-Version, als auch in der HTML-Fassung erzeugt (vgl. Abbildung 2.3, S. 9).

2.3 Systeme

Wie im \LaTeX -Quelltext des obigen Abschnitts und in diesem Abschnitt zu sehen ist, sollten Bindestriche nicht durch - sondern mittels "=" notiert werden. Die erste Variante mit einem einfachen - führt dazu, dass \LaTeX bei der Silbentrennung das Trennen von Wörtern nur an den Stellen durchführt, an denen sich der Bindestrich befindet. Der Ursprung dieses Vorgehens liegt darin begründet, um in Sprachen wie dem Englischen, in denen nur sehr wenige Wörter mit Bindestrich geschrieben werden, die Lesbarkeit von getrennten Wörtern zu erhöhen. Im Deutschen, in dem recht viele Wörter mit Bindestrich geschrieben werden,

kann dies ggf. kontraproduktiv sein. Durch die Variante mit "=" wird die Silbentrennung in Wörtern auch an anderen Stellen als an dem Bindestrich ermöglicht.

Zu beachten ist allerdings, dass innerhalb von Fließumgebungen, z.B. bei Abbildungen und Tabellen, innerhalb von `\caption{}` kein "=" als Bindestrich verwendet werden darf, weil in der HTML-Fassung ansonsten die Nummerierung der Abbildungen und/oder Tabellen fehlerhaft erfolgt.

2.4 Das zu betrachtende System

Es folgt ein Beispiel für ein abgesetztes, wörtliches Zitat. Es handelt sich um eine Definition von Expertensystemen, die der Arbeit von Günter (1992, S. 1) entnommen wurde:

„Expertensysteme sind *wissensbasierte* Programmsysteme. Sie werden für Aufgaben in schwach strukturierten Anwendungsbereichen eingesetzt, für deren Lösung Experten benötigt werden, und die mit ‘klassischen’ algorithmischen Lösungsverfahren nur unzureichend bearbeitet werden können. Dies beinhaltet sowohl die Lösung von Expertenaufgaben, als auch die ‘intelligente’ Unterstützung des Experten bei seiner Arbeit. Ein Programm ist dann *wissensbasiert*, wenn das Wissen des Anwendungsbereiches deklarativ und explizit repräsentiert wird.“

2.4.1 Erste Komponente

Um die Code-Darstellung in der Abbildung 2.3 auf der nächsten Seite wird ebenfalls ein Rahmen gezeichnet (vgl. Abbildung 2.2, S. 7). Diese Art Rahmen wird in der DVI-, PS- und PDF-Version erzeugt, *nicht* jedoch in der HTML-Fassung. Der Grund hierfür ist die in Abbildung 2.3 verwendete `verbatim`-Umgebung, für die der in Abbildung 2.2 verwendete Rahmen leider nicht verwendbar ist.

2.4.2 Zweite Komponente

[...]

2.4.3 Dritte Komponente

[...]

2.4.4 Diskussion zur betrachteten Komponente

[...]

2.4.4.1 Mögliche Alternativen

Beispiel für eine Aufzählung:

1. ...


```

(def-do
  :name PC
  :oberkonzept domaaenenobjekt
  :parameter ((Preis [0.0 inf])
              (Icon "pc" (non-config true)))
  :relationen ((hat-komponente {[(ein PC_Komponente) 7 16] :=
                                [(ein Gehäuse) 1 1]
                                [(ein Mainboard) 1 1]
                                [(ein Prozessor) 1 1]
                                [(ein Speicher) 1 3]
                                [(ein Netzwerkkarte) 0 2]
                                [(ein VGA_Karte) 1 2]
                                [(ein TV_Karte) 0 1]
                                [(ein Soundkarte) 0 1]
                                [(ein Festplatte) 1 2]
                                [(ein CD_Rom) 1 2]})
               (hat-peripherie {[(ein Peripherie_Komponente) 3 8] :=
                                [(ein Monitor) 1 2]
                                [(ein Maus) 1 1]
                                [(ein Tastatur) 1 1]
                                [(ein Drucker) 0 1]
                                [(ein Scanner) 0 1]
                                [(ein Joystick) 0 1]
                                [(ein Boxen_Set) 0 1]})
               :dokumentation "Ein Standard-PC")

```

Abbildung 2.3: Wurzelkonzept für eine PC-Konfiguration

2. ...
3. ...
4. ...
5. ...

2.4.4.2 Besonderheiten

Nachfolgend wird die Beispielumgebung verwendet:

Beispiel 2.4.1 *Hier kommt ein **Beispiel** hin.*

2.5 Anforderungen

Hier werden die Anforderungen an die zu erstellende Software aufgeführt. Es wird beispielhaft die `description`-Umgebung genutzt:

Anforderungen an ... [...]

Anforderungen an ... [...]

Anforderungen an ... [...]

Anforderungen an ... [...]

Anforderungen an ... [...]

Gegebenenfalls eine Unterscheidung in *funktionale* und *nicht-funktionale* Anforderungen vornehmen.

Am Ende dieses Kapitels sollten einige Sätze stehen, die den Übergang zum nächsten Kapitel vorbereiten.

Teil II

Grundlagen

Kapitel 3

Stand der Technik

`fct . < . < . = (nat, nat, nat) bool` *Mixfix*

MANFRED BROY, INFORMATIK, TEIL 1, 1992

In diesem Kapitel wird ein Überblick über ... gegeben. [...]

3.1 Einführung

Für das oben beispielhaft genutzte Zitat wurde eine im Literaturverzeichnis angegebene Quelle verwendet. Im \LaTeX -Quelltext ist ersichtlich wie verfahren werden muss, um in der \LaTeX -, der pdf \LaTeX - und in der $\text{\LaTeX}2\text{HTML}$ -Variante jeweils eine mit entsprechend klickbaren Links hinterlegte Fassung der Referenz zu erhalten.

3.2 Konzepte

Nachfolgend ein Beispiel für eine Definition mit darin enthaltenen mathematischen Symbolen (vgl. Güsgen 2000, S. 269):

Definition 3.2.1 (Constraint-Erfüllung)

Gegeben sei ein CSP mit den Constraints C_1, \dots, C_m auf den Variablen v_1, \dots, v_n mit den Wertebereichen D_1, \dots, D_n . Ein Tupel $(d_1, \dots, d_n) \in D_1 \times \dots \times D_n$ erfüllt ein Constraint C_j , $j \in \{1, \dots, m\}$, falls die zu den Variablen von C_j gehörenden Werte aus (d_1, \dots, d_n) ein Element der Relation von C_j bilden.

3.2.1 Erstes Konzept

Hier ein Beispiel für einen abgesetzten Formelblock *ohne* Nummerierung (vgl. Tsang 1993, S. 91):

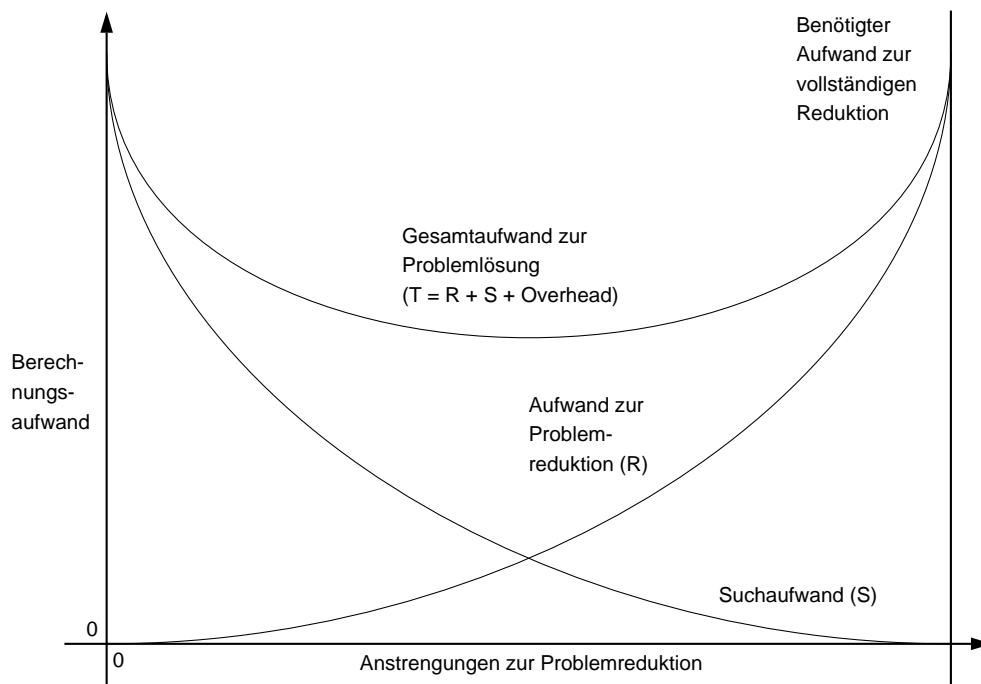


Abbildung 3.1: Aufwand von Problemreduktion vs. Suchaufwand (vgl. Tsang 1993, S. 42)

$$C_{i,j} = C_{i,k} \cdot C_{k,j}, \text{ gdw.}$$

$$C_{i,j,r,s} = (C_{i,k,r,1} \wedge C_{k,j,1,s}) \vee (C_{i,k,r,2} \wedge C_{k,j,2,s}) \vee \dots \vee (C_{i,k,r,t} \wedge C_{k,j,t,s})$$

Wird anstatt `eqnarray*` die Umgebung `eqnarray` verwendet, so wird standardmäßig an der rechten Seite des Formalblocks eine Nummerierung der Zeilen vorgenommen. Einzelne Zeilen können von der Nummerierung ausgenommen werden, indem am Ende der betreffenden Zeile das Makro `\nonumber` aufgerufen wird (vor dem Umbruch mit `\\`).

3.2.2 Zweites Konzept

In der Abbildung 3.1 wird für die HTML-Ausgabe die `thumbnail`-Option eingesetzt. In der HTML-Fassung daher wird anstelle der Abbildung ein verkleinerte Ansicht in den Text eingefügt. Erst durch einen Mausklick auf diese verkleinerte Ansicht wird die Abbildung in der angegebenen vollen Größe angezeigt.

3.2.3 Drittes Konzept

[...]

```

begin
  for  $i \leftarrow 1$  until  $n$  do  $\text{NC}(v_i)$ ;
   $Q \leftarrow \{(v_i, v_j) \mid (v_i, v_j) \in \text{arcs}(G), v_i \neq v_j\}$ 
  while  $Q$  not empty do
    begin
      select and delete any arc  $(v_k, v_m)$  from  $Q$ ;
      if  $\text{REVISE}((v_k, v_m))$  then  $Q \leftarrow Q \cup \{(v_i, v_k) \mid (v_i, v_k) \in \text{arcs}(G), i \neq k, i \neq m\}$ 
    end
  end
end

```

Abbildung 3.2: Der Kantenkonsistenz-Algorithmus AC-3 (vgl. Mackworth 1977, S. 106)

3.3 Algorithmen

In der Abbildung 3.2 ist ein Beispiel für einen Algorithmus zu sehen. Auch um diese Abbildung wird sowohl in der DVI-, PS- und PDF-Version als auch in der HTML-Fassung ein Rahmen erzeugt (vgl. Abbildung 2.2, S. 7 und Abbildung 2.3, S. 9).

Alternativ zu dieser manuellen Erstellung von Pseudocode für die Darstellung von Algorithmen könnten Pakete wie `algorithm` (enthalten in dem Bundle `algorithms`¹) und `algpseudocode` (enthalten in dem Bundle `algorithmicx`², alternativ `algorithmic` aus dem Bundle `algorithms`) eingesetzt werden. Als Alternativen hierzu wären außerdem die Pakete `alg`³ und `algorithm2e`⁴ zu nennen. Deren Kompatibilität zur Ausgabe mit L^AT_EX2HTML ist jedoch fraglich und müsste getestet werden.

3.3.1 Erster Algorithmus

Eine normale Tabelle wird beispielhaft mit dem L^AT_EX-Code für die Tabelle 3.1 auf der nächsten Seite erzeugt. Tabellen, die länger als eine Seite sind, können mit dem Paket `longtable` erzeugt werden. Die von dem Paket angebotene gleichnamige Umgebung `longtable` wird verwendet, um die Verzeichnisse für die Definitionen, die verwendeten Symbole und die Abkürzungen zu erzeugen (siehe den L^AT_EX-Code an den entsprechenden Stellen).

3.3.2 Zweiter Algorithmus

Der L^AT_EX-Code zur Erzeugung der Tabelle 3.2 auf der nächsten Seite ist etwas komplexer: Diese Tabelle verfügt über einzelne Zellen, die über mehrere Spalten hinweg definiert sind. Außerdem wird hier das Paket `ulem` eingesetzt, mit dem Text u. a. unterstrichen und durchgestrichen werden kann.

¹<http://www.ctan.org/tex-archive/help/Catalogue/entries/algorithms.html>

²<http://www.ctan.org/tex-archive/help/Catalogue/entries/algorithmicx.html>

³<http://www.ctan.org/tex-archive/help/Catalogue/entries/alg.html>

⁴<http://www.ctan.org/tex-archive/help/Catalogue/entries/algorithm2e.html>

allgemeine Suchstrategien	Generate & Test (GT) chronologisches Backtracking (BT)
look-back-Strategien	Backjumping (BJ) Backchecking (BC) Backmarking (BM)
look-ahead-Strategien	Forward Checking (FC) Partial Look-Ahead (PLA) Full Look-Ahead (FLA) bzw. Maintaining Arc Consistency (MAC)

Tabelle 3.1: Klassifizierung von Suchstrategien

	P_FSB_Rate	MB_FSB_Rate	S_FSB_Rate
1. initiale Domänen	66 100 133	66 100 133	66 100 133
2. P_FSB_Rate = 100	66 <u>100</u> 133	66 100 133	66 100 133
3. MB_FSB_Rate = 100	100	<u>100</u>	100 133
4. S_FSB_Rate = 100	100	100	<u>100</u> 133
5. Lösung	100	100	100

Tabelle 3.2: Beispiel für Forward Checking

3.3.3 Dritter Algorithmus

[...]

3.4 Verfügbare Systeme

[...]

3.4.1 Erstes System

[...]

3.4.2 Zweites System

[...]

3.4.3 Drittes System

[...]

3.5 Zusammenfassung und Diskussion

[...]

Am Ende dieses Kapitels sollten einige Sätze stehen, die den Übergang zum nächsten Kapitel vorbereiten.

Teil III

Konzeption und Realisierung

Kapitel 4

Konzept

*Ein einleitendes Zitat
für dieses Kapitel.*

VORNAME NACHNAME

Im folgenden Kapitel wird das Konzept für ... entwickelt. [...]

4.1 Einführung

Übergang vom *Stand der Technik* formulieren und konkret die *inneren* Anforderungen auführen, die das Konzept erfüllen muss. [...]

4.2 Erster Bereich

[...]

4.3 Zweiter Bereich

[...]

4.4 Zusammenführung und Weiterentwicklung

[...]

4.5 Systemarchitektur

Hier eine Abbildung und Beschreibung zur Systemarchitektur einfügen (z. B. ein Schichtenmodell). [...]

4.6 Diskussion

Das vorgestellte Konzept an sich und gegen die in der Einführung in Abschnitt 4.1 auf Seite 21 genannten *inneren* Anforderungen an das Konzept diskutieren. Gegebenenfalls unterschiedliche Realisierungsmöglichkeiten des Konzepts gegeneinander abwägen. [...]

Am Ende dieses Kapitels sollten einige Sätze stehen, die den Übergang zum nächsten Kapitel vorbereiten.

Kapitel 5

Implementierung

*Ein einleitendes Zitat
für dieses Kapitel.*

VORNAME NACHNAME

Dieser Abschnitt enthält die Beschreibung der Implementierung von ... Es werden der Reihe nach die enthaltenen Komponenten aufgezählt und dokumentiert. Abschließend erfolgt die Beschreibung der Integration in ... [...]

5.1 Einleitung

Die verwendete Plattform und eingesetzte Werkzeuge und Bibliotheken beschreiben. [...]

5.2 Eingesetzte Konzepte

Zum Beispiel eingesetzte Architektur- und Entwurfsmuster beschreiben. [...]

5.3 Übersicht über die Packages

[...]

5.4 Erste Komponente

[...]

5.5 Zweite Komponente

[...]

5.6 Dritte Komponente

[...]

5.7 Ausnahmebehandlung

[...]

5.8 Integration

[...]

5.8.1 Anbindung an das betrachtete System

[...]

5.8.2 Integration in andere Systeme

[...]

Am Ende dieses Kapitels sollten einige Sätze stehen, die den Übergang zum nächsten Kapitel vorbereiten.

Kapitel 6

Validierung

*Ein einleitendes Zitat
für dieses Kapitel.*

VORNAME NACHNAME

An dieser Stelle erfolgt eine Validierung der Umsetzung von ... anhand der realisierten Funktionalität und der zuvor benannten Anforderungen. Neben synthetischen Problemstellungen wird die Integration von ... in ... validiert. Außerdem erfolgt eine Positionierung im Vergleich mit weiterführenden Arbeiten.

6.1 Einleitung

Im Folgenden wird erläutert, welche Teile der Konzeption aus Kapitel 4 mit der Implementierung realisiert wurden. Daneben erfolgt eine Validierung sowohl anhand von „synthetischen“ Problemstellungen als auch im praktischen Einsatz. [...]

Im Zuge der Erstellung dieser Arbeit gerieten zunehmend weitergehende Lösungen in den Blickwinkel. [...] Diesen Aspekten und einer Abgrenzung zu dem in dieser Arbeit entwickelten System ist daher ein eigener Abschnitt gewidmet.

6.2 Eigenschaften des Prototypen

Allgemeine Eigenschaften der entwickelten Software beschreiben. Welche Realisierungsvariante des Konzepts wurde weshalb gewählt (vgl. Abschnitt 4.6, S. 22)? Inwiefern erfüllt die entwickelte Software die *internen* Anforderungen an das Konzept (vgl. Abschnitt 4.1, S. 21)? [...]

6.3 Validierung anhand synthetischer Probleme

[...]

6.3.1 Übersicht über die Problemstellungen

[...]

6.3.2 Vorbereitende Maßnahmen

[...]

6.3.3 Ergebnisse

[...]

6.3.4 Zusammenfassung

[...]

6.4 Validierung im Praxiseinsatz

[...]

6.4.1 Vorbereitende Maßnahmen

[...]

6.4.2 Ergebnisse

[...]

6.4.3 Zusammenfassung

[...]

6.5 Vergleich mit weiterführenden Ansätzen

Wie machen es die anderen? [...]

6.6 Zusammenfassung

Abgleich mit den *externen* Anforderungen in Abschnitt 2.5 auf Seite 9, die sich aus der Anwendungsdomäne ergeben. [...]

Kapitel 7

Zusammenfassung und Ausblick

*Ein einleitendes Zitat
für dieses Kapitel.*

VORNAME NACHNAME

In diesem Kapitel erfolgt eine Zusammenfassung sowie ein kurzer Ausblick hinsichtlich der Erweiterungsmöglichkeiten der Konzeption und des entwickelten Prototypen.

7.1 Zusammenfassung

Mit den Zielen und „Angestrebten Ergebnissen“ in der Einleitung in Abschnitt 1.2 auf Seite 3 abgleichen. [...]

7.2 Ausblick

Was zählt ist das, was hinten raus kommt. [...]

Teil IV

Anhänge

Anhang A

Installation

A.1 Abschnittsüberschrift

[...]

A.1.1 Unterabschnittsüberschrift

[...]

A.1.1.1 Unterunterabschnittsüberschrift

[...]

Paragraphüberschrift [...]

Unterparagraphüberschrift [...]

Abschnittsüberschrift

[...]

Unterabschnittsüberschrift

[...]

Unterunterabschnittsüberschrift

[...]

Paragraphüberschrift [...]

Unterparagraphüberschrift [...]

Anhang B

Programm

Als Beispiel für das Einbinden von Quelltext ist in diesem Anhang der Quelltext einer Parser-Grammatik dokumentiert. Für die Darstellung des Quelltextes wird das Paket `listings` und das darin enthaltene Makro `\lstinputlisting{}` verwendet. Durch `\lstinputlisting{}` kann eine separate Datei mit Quelltext eingebunden werden. Alternativ kann der Quelltext direkt in der \LaTeX -Datei innerhalb der Umgebung `lstlisting` angegeben werden.

Weil die HTML-Ausgabe mittels $\text{\LaTeX}2\text{HTML}$ das Paket `listings` nicht unterstützt, ist hierfür eine separate Ausgabe des Quelltextes innerhalb einer `verbatim`-Umgebung notwendig. Hier kann das Makro `\verbatiminput{}` aus dem Paket `verbatim` verwendet werden, um eine separate Datei einzulesen. Für die direkte Angabe von Quelltext innerhalb der \LaTeX -Datei kann die gewohnte `verbatim`-Umgebung verwendet werden.

Zuerst wird an dieser Stelle die vom Parser zum Einlesen von gültigen Zeichen benötigte Datei `scanner.lex` für die lexikalische Analyse durch JLex aufgeführt (vgl. Berk 2000):

```
1  package yacs.parser;
2
3  import java_cup.runtime.Symbol;
4
5  %%
6
7  %cup
8  %public
9
10 ALPHA=[A-Za-z]
11 DIGIT=[0-9]
12
13 %%
14
15 " ; "      {return new Symbol(sym.SEMI);}
16 " , "      {return new Symbol(sym.COMMA);}
17 " = "      {return new Symbol(sym.EQUAL);}
18 " != "     {return new Symbol(sym.NOT_EQUAL);}
19 " > "      {return new Symbol(sym.GREATER);}
20 " < "      {return new Symbol(sym.LOWER);}
21 " >= "     {return new Symbol(sym.GREATER_EQUAL);}
22 " <= "     {return new Symbol(sym.LOWER_EQUAL);}
23 " + "      {return new Symbol(sym.PLUS);}
```

```

24 "-" {return new Symbol(sym.MINUS);}
25 "*" {return new Symbol(sym.TIMES);}
26 "/" {return new Symbol(sym.DIVIDE);}
27 "(" {return new Symbol(sym.LPAREN);}
28 ")" {return new Symbol(sym.RPAREN);}
29 "[" {return new Symbol(sym.LBRACKET);}
30 "]" {return new Symbol(sym.RBRACKET);}
31 {DIGIT}+ {return new Symbol(sym.NUMBER,
32                                     new Integer(yytext()));}
33 {DIGIT}+"."{DIGIT}+ {return new Symbol(sym.FLOAT,
34                                     new Double(yytext()));}
35 {ALPHA}({ALPHA}|{DIGIT}|_".")* {return new Symbol(sym.VARIABLE, yytext());}
36 [\t\r\n\f\b] {/* ignore white space. */}
37 . {System.err.println("Illegal character: "
38                       +yytext());}

```

Von Scanner erfasste Zeichen werden an den Parser weitergegeben und anhand der Java-CUP-Grammatik in der Datei `parser.cup` ausgewertet (vgl. Hudson 1999):

```

1 package yacs.parser;
2
3 import java.util.HashMap;
4
5 import yacs.domain.*;
6
7 import java_cup.runtime.*;
8
9 action code {
10     /* this is where the action code goes */
11
12     /** der erzeugte Ausdruck */
13     public Expression expression;
14
15     /** HashtMap zum "Wiederfinden" der generierten Variablen */
16     public HashMap actionVariablesMap = new HashMap();
17
18     /** temporaere Variable zum Zwischenspeichern */
19     public Variable tmpVariable;
20
21 :};
22
23 parser code {
24     /* this is where the parser code goes */
25
26     /** HashtMap zum Zwischenspeichern der bereits vorhandenen Variablen */
27     public HashMap parserVariablesMap = new HashMap();
28
29     /**
30      * Hinzufuegen bereits vorhandener Variablen. Anstatt neue Variablen
31      * zu erzeugen, werden so die Referenzen bestehender Objekte
32      * verwendet.
33      * @param variablesMap HashMap
34      */
35     public void addVariablesMap(HashMap variablesMap) {
36         this.parserVariablesMap.putAll(variablesMap);

```

```

37     }
38
39     /**
40      * Gibt den gescannten und geparsten Ausdruck als Expression zurueck.
41      * @return Expression
42      */
43     public Expression expression() {
44         return action_obj.expression;
45     }
46
47 :};
48
49 init with {:
50
51     // Die in "parser" bereits befindlichen Variablenreferenzen in das
52     // "action_obj" uebertragen. Dies muss hier separat geschehen, da
53     // erst waehrend der "init"-Phase das "action_obj" instantiiert ist.
54     action_obj.actionVariablesMap.putAll(this.parserVariablesMap);
55
56 :};
57
58 /* Terminals (tokens returned by the scanner). */
59 terminal      UMINUS, SEMI, LPAREN, RPAREN, COMMA, LBRACKET, RBRACKET;
60 terminal      PLUS, MINUS, TIMES, DIVIDE;
61 terminal      EQUAL, NOT_EQUAL, GREATER, LOWER, GREATER_EQUAL, LOWER_EQUAL;
62 terminal Integer NUMBER;
63 terminal Double  FLOAT;
64 terminal String  VARIABLE;
65
66 /* Non Terminals */
67 non terminal      expr_all;
68 non terminal Expression  expr_list, expr, expr_part;
69
70 /* Precedences */
71 precedence left EQUAL, NOT_EQUAL, GREATER, LOWER, GREATER_EQUAL, LOWER_EQUAL;
72 precedence left PLUS, MINUS;
73 precedence left TIMES, DIVIDE;
74 precedence left UMINUS;
75
76 /* The grammar */
77
78 expr_all ::= expr_list:e
79         {:
80             System.out.println(" => "+e.toString());
81             this.expression = e;
82         :}
83         ;
84
85 expr_list ::= expr:e
86           {: RESULT = e; :}
87           | expr_list:l expr:r
88           {: RESULT = new BinaryOperator(sym.SEMI, ";", l, r); :}
89           ;
90

```

```

91  expr      ::= expr_part:l EQUAL expr_part:r SEMI
92              {: RESULT = new BinaryOperator(sym.EQUAL, "=", l, r); :}
93              | expr_part:l NOT_EQUAL expr_part:r SEMI
94              {: RESULT = new BinaryOperator(sym.NOT_EQUAL, "!=", l, r); :}
95              | expr_part:l GREATER expr_part:r SEMI
96              {: RESULT = new BinaryOperator(sym.GREATER, ">", l, r); :}
97              | expr_part:l LOWER expr_part:r SEMI
98              {: RESULT = new BinaryOperator(sym.LOWER, "<", l, r); :}
99              | expr_part:l GREATER_EQUAL expr_part:r SEMI
100             {: RESULT = new BinaryOperator(sym.GREATER_EQUAL, ">=", l, r); :}
101             | expr_part:l LOWER_EQUAL expr_part:r SEMI
102             {: RESULT = new BinaryOperator(sym.LOWER_EQUAL, "<=", l, r); :}
103             ;
104
105  expr_part ::= NUMBER:n
106             {: RESULT = new Constant(n); :}
107             | LBRACKET FLOAT:lo COMMA FLOAT:hi RBRACKET
108             {: RESULT = new Constant(lo, hi); :}
109             | LBRACKET NUMBER:lo COMMA NUMBER:hi RBRACKET
110             {: RESULT = new Constant(lo.doubleValue(), hi.doubleValue()); :}
111             | LBRACKET FLOAT:lo COMMA NUMBER:hi RBRACKET
112             {: RESULT = new Constant(lo.doubleValue(), hi.doubleValue()); :}
113             | LBRACKET NUMBER:lo COMMA FLOAT:hi RBRACKET
114             {: RESULT = new Constant(lo.doubleValue(), hi.doubleValue()); :}
115             | VARIABLE:v
116             {: tmpVariable = (Variable)actionVariablesMap.get(v);
117              if (tmpVariable == null) {
118                  // Variable mit leerer Domaene instantiieren:
119                  tmpVariable = new Variable(v, new Domain());
120                  actionVariablesMap.put(v, tmpVariable);
121              }
122              RESULT = tmpVariable;
123             :}
124             | expr_part:l PLUS expr_part:r
125             {: RESULT = new BinaryOperator(sym.PLUS, "+", l, r); :}
126             | expr_part:l MINUS expr_part:r
127             {: RESULT = new BinaryOperator(sym.MINUS, "-", l, r); :}
128             | expr_part:l TIMES expr_part:r
129             {: RESULT = new BinaryOperator(sym.TIMES, "*", l, r); :}
130             | expr_part:l DIVIDE expr_part:r
131             {: RESULT = new BinaryOperator(sym.DIVIDE, "/", l, r); :}
132             | MINUS expr_part:e
133             {: RESULT = new UnaryOperator(sym.UMINUS, "-", e); :}
134             %prec UMINUS
135             | LPAREN expr_part:e RPAREN
136             {: RESULT = e; :}
137             ;

```

Anhang C

API-Dokumentation

Nachfolgend ist die API-Dokumentation der in Kapitel 5 auf Seite 23 ff. beschriebenen Implementierung aufgeführt. Attribute, Standardkonstruktoren und *private*-Methoden werden nicht aufgelistet, ebenso werden, um diese Übersicht kompakt zu halten, geerbte Methoden von übergeordneten Klassen nicht separat ausgewiesen (außer sie werden durch eine spezielle Funktionalität überschrieben). Abstrakte Methoden und Implementierungen von Methoden aus Interfaces werden in den Unterklassen ebenfalls nicht nochmals aufgeführt. Deren Dokumentation kann den jeweils übergeordneten Klassen entnommen werden.

C.1 Package ...

[...]

C.1.1 Interface ...

[...]

Deklaration:

- `public interface ...`

Methoden:

- `public(...)`
[...]
- `public(...)`
[...]
- `public(...)`
 `throws ...Exception`
[...]

- `public(...)`
 `throws ...Exception`
 `[...]`

C.1.2 Klasse ...

`[...]`

Deklaration:

- `public class ...`
 `implements ...`

Konstruktoren:

- `public ...()`
 `[...]`
- `public ...(...)`
 `[...]`

Methoden:

- `public(...)`
 `[...]`
- `public(...)`
 `[...]`
- `public(...)`
 `throws ...Exception`
 `[...]`
- `public(...)`
 `throws ...Exception`
 `[...]`

Anhang D

Glossar

API (Abkürzung für engl. *Application Programming Interface*) Eine *API* ist eine dokumentierte Software-Schnittstelle, mit deren Hilfe ein Programm die Funktionen eines anderen Programms nutzen kann (gleiches gilt für die *API* eines Betriebssystems).

Backus-Naur-Form (BNF) Die Backus-Naur-Form ist eine kompakte und formale Metasyntax, die zur Darstellung von *kontextfreien Grammatiken* eingesetzt wird. Dies betrifft die Syntax gängiger höherer Programmiersprachen. Sie wird auch für die Notation von Befehlssätzen und Kommunikationsprotokollen verwendet. Durch die *Backus-Naur-Form* ist es möglich, die Syntax einer Programmiersprache formal exakt, d. h. ohne die Ungenauigkeiten natürlicher Sprachen, darzustellen.

BSD-Lizenz (Abkürzung für engl. *Berkeley-Source-Distribution-Lizenz*) Die *BSD-Lizenz* ist wie die *GNU General Public License* (\rightarrow *GPL*) eine Lizenz für freie Software.¹ Ursprünglich wurde die Lizenz nur für Software verwendet, die an der Universität von Kalifornien in Berkeley entwickelt wurde, fand aber bald recht weite Verbreitung. Sie ähnelt im Wesentlichen der *GPL*, ist teilweise jedoch liberaler formuliert. So ist es wie in der *GPL* erlaubt, Software beliebig zu kopieren und zu verändern, jedoch darf das Programm auch in kommerzieller Software verwendet werden. Berühmtestes Beispiel ist die Verwendung des Netzwerkmoduls von *BSD* in Microsoft Windows.

Framework (objektorientiertes) Eine Menge kooperierender Klassen, welche die Elemente eines wiederverwendbaren Entwurfs für eine bestimmte Art von Software darstellen. Ein *Framework* bietet eine Architekturhilfe beim Aufteilen des Entwurfs in abstrakte Klassen und beim Definieren ihrer Zuständigkeiten und Interaktionen. Ein Entwickler passt das *Framework* für eine bestimmte Anwendung an, indem er Unterklassen der *Framework*-Klassen bildet und ihre Objekte zusammensetzt (vgl. Gamma et al. 1996, S. 445).

GPL (Abkürzung für engl. *GNU General Public License*) bezeichnet eine Lizenz für „freie“ Software (bzw. genauer für den Programmquellcode), die von *Richard Stallman*, dem

¹<http://www.opensource.org/licenses/bsd-license.php>

Begründer des GNU-Projekts entworfen wurde.² Freie Software bedeutet in diesem Zusammenhang, dass sie von jedem gebraucht, geändert und angepasst werden kann. Besonders wichtig ist, dass jegliche veränderte Software wieder unter die *GPL* gestellt werden muss, d. h. frei weitergegeben werden muss. Andere wichtige Lizenzen sind die *Lesser General Public License* (\rightarrow *LGPL*) und die \rightarrow *BSD-Lizenz*.

HTML (Abkürzung für engl. *Hypertext Markup Language*) Standardisierte Seitenbeschreibungssprache für WWW-Seiten im Internet bzw. Intranet, welche von Charles F. Goldfarb entwickelt wurde und in der ISO-Norm 8879 definiert ist (auch \rightarrow *XML*). Sie definiert sowohl die Gestaltung, den Inhalt und die Grafik der Seite als auch die Hyperlinks zu eigenen oder fremden Seiten.

LGPL (Abkürzung für engl. *GNU Lesser General Public License*) ist eine etwas entschärfte Variante (engl. *lesser*, „weniger“) der \rightarrow *GPL*, deren Hauptunterschied darin liegt, dass die Verwendung von Programmen, die unter dieser Lizenz stehen, nicht dazu führen muss, dass die ganze Software unter dieser Lizenz (und damit frei) herausgegeben werden muss.³ Besonders gut eignet sich diese Lizenz daher für Bibliotheken, was der alte Name, *Library General Public License*, ausdrückt. Dieser wurde jedoch geändert, da diese Lizenz nicht nur auf Bibliotheken beschränkt gelten sollte.

NP Problemklasse für die Menge der mit einem nichtdeterministischen Algorithmus mit polynomialen Aufwand lösbaren Probleme (\rightarrow *P*) (vgl. Claus und Schwill 2001).

NP-hart Ein Problem *X* ist *NP-hart*, wenn jedes Problem aus *NP* polynomial auf *X* reduzierbar ist (vgl. Claus und Schwill 2001).

NP-vollständig Ein Problem ist *NP-vollständig*, wenn es *NP-hart* ist, und wenn es zu *NP* gehört (vgl. Claus und Schwill 2001).

P Bezeichnung für die Menge aller Probleme, die ein deterministischer Algorithmus mit polynomialen Zeitaufwand löst (\rightarrow *NP*) (vgl. Claus und Schwill 2001).

Polymorphie benennt die Eigenschaft objektorientierter Programmiersprachen, dass in einer Klasse die geerbten Methoden redefiniert, d. h. *überschrieben* oder *überladen* werden können. *Überschreiben* bedeutet, dass in der abgeleiteten Klasse eine Methode mit dem gleichen Namen und der gleichen Signatur wie in der Oberklasse definiert wird. Die Signatur ist durch den Rückgabewert und die Parameterliste der Methode definiert. Man spricht von einer *Überladung*, wenn eine Methode mit gleichem Namen aber unterschiedlicher Parameterliste eingeführt wird.

XML (Abkürzung für engl. *eXtensible Markup Language*) *XML* ist eine Methode zur Repräsentation strukturierter Daten. *XML* ist (wie auch \rightarrow *HTML*) eine „vereinfachte“ Version der *Standard Generalized Markup Language* (SGML), die es Programmieren von Web-Seiten erleichtert SGML-Anwendungen zu schreiben, und dabei eigene Dokumententypen (DTD) festzulegen.

²<http://www.gnu.org/licenses/gpl.html>

³<http://www.gnu.org/licenses/lgpl.html>

Literaturverzeichnis

Hinweis: Die Zahlen am Ende eines jeden Eintrags kennzeichnen die Seitenzahlen der vorliegenden Arbeit, auf denen die jeweilige Quelle zitiert wird.

Berk 2000

BERK, Elliot: JLex: A lexical analyzer generator for Java / Princeton University, Department of Computer Science. Princeton, New Jersey, USA, 6. September 2000 (Version 1.2.5). – Online Handbuch. – URL <http://www.cs.princeton.edu/~appel/modern/java/JLex/current/manual.html>. – Zugriffsdatum: 14. September 2005 33

Broy 1992

BROY, Manfred: *Informatik, Eine grundlegende Einführung*. Bd. I. Berlin, Heidelberg, New York : Springer Verlag, 1992. – xii + 250 S. – ISBN 3-540-55191-3 13

Claus und Schwill 2001

CLAUS, Volker (Hrsg.) ; SCHWILL, Andreas (Hrsg.): *Duden Informatik, Ein Fachlexikon für Studium und Praxis*. 3. Aufl. Mannheim, Leipzig, Wien, Zürich : Dudenverlag, Bibliographisches Institut, 2001. – 762 S. – ISBN 3-411-05233-3 40

Cunis et al. 1991

CUNIS, Roman (Hrsg.) ; GÜNTER, Andreas (Hrsg.) ; STRECKER, Helmut (Hrsg.): *Das PLAKON-Buch, Ein Expertensystemkern für Planungs- und Konfigurierungsaufgaben in technischen Domänen*. Berlin, Heidelberg, New York : Springer Verlag, 1991 (Informatik-Fachberichte, Subreihe Künstliche Intelligenz 266). – 279 S. – ISBN 3-540-53683-3 42

Freuder 1997

FREUDER, Eugene C.: In Pursuit of the Holy Grail. In: *Constraints, An International Journal* 2 (1997), April, Nr. 1, S. 57–61. – Zugl.: ACM CSUR, 28 (1996), Nr. 4es, Artikel Nr. 63. – ISSN 1383-7133 vii

Gamma et al. 1996

GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software*. 1. Aufl. München : Addison-Wesley, 1996. – xx + 479 S. – ISBN 3-89319-950-0 39

Günter 1991

GÜNTER, Andreas: Expertensysteme für Konstruktionsaufgaben. In: (Cunis et al. 1991), S. 1–3. – ISBN 3-540-53683-3 5, 6

Günter 1992

GÜNTER, Andreas: *Flexible Kontrolle in Expertensystemen zur Planung und Konfiguration in technischen Domänen*. Sankt Augustin : Infix Verlag, 1992 (DISKI 3). – iv + 239 S. – Zugl.: Hamburg, Universität, Dissertation, 1991. – ISBN 3-929037-03-3 8

Güsgen 2000

GÜSGEN, Hans W.: Constraints. In: GÖRZ, Günther (Hrsg.) ; ROLLINGER, Claus-Rainer (Hrsg.) ; JOSEF, Schneeberger (Hrsg.): *Handbuch der Künstlichen Intelligenz*. 3., vollst. überarb. Aufl. München, Wien : Oldenbourg Verlag, 2000, Kap. 8, S. 267–287. – ISBN 3-486-25049-3 13

Hudson 1999

HUDSON, Scott E.: CUP User's Manual / Georgia Institute of Technology, Graphics Visualization and Usability Center. Atlanta, Georgia, USA, Juli 1999 (Version 0.10j). – Online Handbuch. – URL <http://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html>. – Zugriffsdatum: 14. September 2005 34

Kopka 1996

KOPKA, Helmut: *L^AT_EX Einführung*. Bd. 1. 2., überarb. Aufl. Bonn : Addison-Wesley, 1996. – xix + 502 S. – ISBN 3-8273-1025-3

Mackworth 1977

MACKWORTH, Alan K.: Consistency in Networks of Relations. In: *Artificial Intelligence* 8 (1977), Februar, Nr. 1, S. 99–118. – ISSN 0004-3702 15

Neumann 1991

NEUMANN, Bernd: Expertensysteme zur Konstruktion: Anforderungen an ein Werkzeugsystem. In: (Cunis et al. 1991), Kap. 2, S. 12–27. – ISBN 3-540-53683-3 5, 6, 7

Stumptner 1997

STUMPTNER, Markus: An Overview of Knowledge-Based Configuration. In: *AI Communications (AICOM)* 10 (1997), Juli, Nr. 2, S. 111–125. – ISSN 0921-7126 7

Tsang 1993

TSANG, Edward P. K.: *Foundations of Constraint Satisfaction*. London, San Diego, New York : Academic Press, 1993 (Computation in Cognitive Science). – 421 S. – URL <http://cswww.essex.ac.uk/CSP/papers/Tsang-Fcs1993.pdf/>. – Zugriffsdatum: 20. September 2004. – ISBN 0-12-701610-4 13, 14

Abkürzungsverzeichnis

Allgemeine Abkürzungen

aktual.	aktualisiert
Anm.	Anmerkung
Aufl.	Auflage
Bd.	Band
bzgl.	bezüglich
bspw.	beispielsweise
bzw.	beziehungsweise
ca.	circa
engl.	englisch
erw.	erweiterte
et al.	et alii
etc.	et cetera
evtl.	eventuell
f.	folgend
ff.	fortfolgend
gdw.	genau dann wenn
ggf.	gegebenenfalls
Hrsg.	Herausgeber
inkl.	inklusive
d. h.	das heißt
i. A.	im Allgemeinen
i. d. R.	in der Regel
Kap.	Kapitel
max.	maximal
min.	minimal/mindestens
Nr.	Nummer
o. ä.	oder ähnlich
S.	Seite
sog.	sogenannt
u.	und
u. a.	und andere

überarb.	überarbeitet
usf.	und so fort
usw.	und so weiter
u. U.	unter Umständen
vgl.	vergleiche
vollst.	vollständig
vs.	versus
z. B.	zum Beispiel
zit.	zitiert
z. T.	zum Teil
zugl.	zugleich

Fachbegriffliche Abkürzungen

API	Application Programming Interface
BC	Backchecking
BJ	Backjumping
BM	Backmarking
BNF	Backus-Naur-Form
BT	(chronologisches) Backtracking
FC	Forward Checking
FLA	Full Look-Ahead
GT	Generate & Test
MAC	Maintaining Arc Consistency
PLA	Partial Look-Ahead

Organisationen, Institute und Unternehmen

AAAI	American Association for Artificial Intelligence
ACM	Association for Computing Machinery
ERCIM	European Research Consortium for Informatics and Mathematics
GI	Gesellschaft für Informatik
GMD	Gesellschaft für Mathematik und Datenverarbeitung
GNU	GNU's Not UNIX
INRIA	Institut National de Recherche en Informatique et en Automatique
IITB	Fraunhofer-Institut für Informations- und Datenverarbeitung
LIRMM	Le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
MIT	Massachusetts Institute of Technology
TZI	Technologie-Zentrum Informatik

W3C	World Wide Web Consortium
-----	---------------------------

Konferenzen und Workshops

ECAI	European Conference on Artificial Intelligence
IAAI	Conference on Innovative Applications of Artificial Intelligence
IJCAI	International Joint Conference on Artificial Intelligence
KI	Deutsche Jahrestagung Künstliche Intelligenz
OOPSLA	ACM Conference on Object-Oriented Programming Systems, Languages and Applications
PA Java	Conference and Exhibition on Practical Application of Java
PuK	GI Workshop Planen, Scheduling und Konfigurieren, Entwerfen
XPS	Deutsche Expertensystemtagung

Zeitschriften, Serien und Enzyklopädien

AI	Artificial Intelligence
AICOM	AI Communications
AI EDAM	Artificial Intelligence for Engineering Design, Analysis and Manufacturing
CACM	Communications of the ACM
CSUR	ACM Computing Surveys
DISKI	Dissertationen zur Künstlichen Intelligenz
ECS	Encyclopedia of Cognitive Science
ENTCS	Electronic Notes in Theoretical Computer Science
IJAIT	International Journal on Artificial Intelligence Tools
JACM	Journal of the ACM
JAIR	Journal of Artificial Intelligence Research
JFLP	Journal of Functional and Logic Programming
LNCS	Lecture Notes in Computer Science
MITECS	The MIT Encyclopedia of the Cognitive Sciences
PAMI	IEEE Transactions on Pattern Analysis and Machine Intelligence
TOCHI	ACM Transactions on Computer-Human Interaction

Stichwortverzeichnis

A

API 4, 37, 44

B

Backchecking *siehe* BC
 Backjumping *siehe* BJ
 Backmarking *siehe* BM
 Backtracking *siehe* BT
 Backus-Naur-Form *siehe* BNF
 Baumann, Helge 49
 BC 16, 44
 BJ 16, 44
 BM 16, 44
 BNF 39, 44
 Broy, Manfred 13
 BSD-Lizenz 39 f.
 BT 16, 44

D

Daly, Patrick W. 49
 DTD 40

E

Expertensystem 5, 7 f.

F

FC 16, 44
 FLA 16, 44
 Forward Checking *siehe* FC
 Framework
 objektorientiert 39
 Freuder, Eugene C. vii
 Full Look-Ahead *siehe* FLA

G

Generate & Test *siehe* GT
 GMD 44
 GNU 44
 GNU-Projekt 40
 Goldfarb, Charles F. 40
 GPL 39
 GT 16, 44

H

HTML 40

I

Indexeintrag
 Untereintrag 6

J

Java CUP 34
 JLex 33

K

Knappen Jörg 49
 Knuth, Donald E. 49
 Konstruktionssystem 5

L

Lamport, Leslie 49
 LGPL 40
 Linux 49

M

MAC 16, 44
 Maintaining Arc Consistency ... *siehe* MAC

Meta

-syntax 39

N

Nachname, Vorname 5, 21, 23, 25, 27

Neuss, Wolfgang 49

NP-hart 40

NP-vollständig 40

O

Oberdiek, Heiko 49

P

Partial Look-Ahead *siehe* PLA

PLA 16, 44

Polymorphie 40

R

R1/XCON 7

S

SGML 40

Stallman, Richard 39

Syntax 39

T

Technologie-Zentrum Informatik . *siehe* TZI

TZI 44

X

XCON 7

XML 40

Kolophon

Stell dir vor, es geht, und keiner kriegt's hin.

WOLFGANG NEUSS

Diese Arbeit wäre in der vorliegenden Form ohne die Satzbeschreibungssprache \TeX von *Donald E. Knuth*¹ und dem darauf basierenden Makropaket \LaTeX von *Leslie Lamport*² nicht möglich gewesen.

Der Text wurde mit \LaTeX 2_ε aus der 11 Punkt Computer-Modern-Schrift gesetzt. Für den Satz fanden die ec-Schriften³ von *Jörg Knappen* Verwendung.

Die Abbildungen in dieser Arbeit wurden mittels Dia⁴, xfig⁵, OpenOffice.org Draw⁶, Inkscape⁷, Kivio⁸ und Karbon⁹ erstellt.

Das Literaturverzeichnis wurde nach DIN 1505 mit \BibTeX und dem *dinat*-Paket¹⁰ von *Helge Baumann* sowie dem *natbib*-Paket¹¹ von *Patrick W. Daly* erzeugt.

Die Miniaturbildchen („Thumbnails“) für die PDF-Version dieses Dokuments wurden mit dem *thumbpdf*-Paket¹² von *Heiko Oberdiek* generiert und eingebunden.

Der Satz dieses Dokuments erfolgte am 4. April 2011 um 10:17 Uhr auf einer unter Ubuntu Linux 9.04¹³ („Jaunty Jackalope“) betriebenen Intel Core2-Duo-Maschine („T7700“) mit 2.4 Gigahertz Taktung unter Verwendung der Distribution \TeX Live 2007.

Zur vollständigen Auflösung aller Referenzen benötigte \LaTeX insgesamt sechs Läufe.

Alle Eingabedateien brachten eine Summe von 0 Kilobytes auf. Die Größe der Ausgabedateien betrug im DVI-Format 0 Kilobytes, im PS-Format 0 Kilobytes, im PDF-Format 0 Kilobytes und im HTML-Format 0 Kilobytes.

¹<http://www-cs-faculty.stanford.edu/~knuth/>

²<http://research.microsoft.com/users/lamport/>

³<http://www.uni-mainz.de/~knappen/jk007.html>

⁴<http://www.gnome.org/projects/dia/>

⁵<http://www.xfig.org>

⁶<http://www.openoffice.org>

⁷<http://www.inkscape.org>

⁸<http://www.koffice.org/kivio>

⁹<http://www.koffice.org/karbon>

¹⁰<ftp://ftp.dante.de/tex-archive/biblio/bibtex/contrib/german/dinat/>

¹¹<ftp://ftp.dante.de/tex-archive/macros/latex/contrib/natbib/>

¹²<ftp://ftp.dante.de/tex-archive/support/thumbpdf/>

¹³<http://www.ubuntu.com>