Constraint Handling Rules

Theorie und Praxis

Thom Frühwirth
Institut für Informatik
Ludwig-Maximilians-Universität München

9. Juli 1998

Anmerkung

Hände nicht in Hosentaschen

Ins Publikum schauen und sprechen, vor allem am Anfang und Ende

Overheadfolie liegen lassen

Nicht vor der Projektionsfläche stehen

Nicht mit Overheadfolien spielen

Entweder auf Folie zeigen und bunt malen oder auf Projektionsfläche zeigen

Constraint-Programmierung

Der Ansatz

- Beispiel Zahlenschloß:

  0 1 2 3 4 5 6 7 8 9

  Größer oder gleich 5.

  Primzahl.
- Deklarative Problemrepräsentation durch Variablen und Constraints:

  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \wedge primzahl(x)$
- Constraint Propagierung sorgt für eine schnelle Reduktion des Suchraumes (Constraintlöser).

  $x \in \{0, 1, \ldots, 9\} \wedge x \geq 5 \rightarrow x \in \{5, 6, 7, 8, 9\}$

Anmerkung

KONTEXT UMFELD MEINER ARBEIT

- Getrennt von Problemlösungsstrategie:
  Konjunktion von atomaren Formeln in Prädikatenlogik erster Stufe mit Gleichheit.
- Logische Folgerungen.

  Sieht trivial aus, Probleme sind aber oft **NP-vollständig**, Algorithmen exponentiell.

Constraints = Einschränkungen, (Wert-, Rand-, Neben-)Bedingungen.

Constraints beschreiben die Eigenschaften und Beziehungen von teilweise unbekannten Objekten.

(Atomare) Constraints: Spezielle Prädikate der Prädikatenlogik erster Stufe.

Constrainttheorie: Nichtleere, konsistente Theorie, die die Constraints beschreibt.

Constraint(problem), Anfrage: Zu lösende, allquantorfreie Konjunktion von atomaren Constraints.

Lösung, Antwort: Wertebelegungen für Variablen, die Anfrage erfüllen.
Allgemeiner: Möglichst starke Vereinfachung (z.B. Normalform) einer Anfrage.

Constraintlöser: Programm, das Constraints vereinfacht und löst.

Constraint-Programm: schickt nach und nach Constraints an Constraintlöser, erwartet (Teil-)Lösungen.

Constraint-Programmierung

Die Anwendungen

Schneller zu robuster, flexibler, wartbarer Software.

- Lufthansa: Personalplanung nach Störungen.
- Hongkong: Container laden und lagern.
- Renault: Kurzfristige Fertigungsplanung.
- Airbus: Layout der Pilotenkabine.
- Siemens: Verifizierung von Schaltkreisen.
- Caisse d'epargne: Portfolio Management.

In Entscheidungsunterstützungssystemen zum Planen, Konfigurieren, für Design, Analyse.

[Frühwirth et al., in *Logic Programming in Action*, Springer LNCS, 1992]
[Frühwirth, Abdennadher, *Constraint-Programmierung*, Springer Verlag, 1997]

KURZ SAGEN UND ZEIGEN Anmerkung

Umsatz 1996 ca. 100 Millionen Dollar, wie Data Mining, 1% von Microsoft. Kommerziellen Anwendungen 1996 ca. 300.

- Ilog (F,USA): Ilog Optimization Suite (incl. C-Plex, Numerica).
- Cosytec (F): CHIP V5.
- Siemens, IF Computer (A,J,D): IF/Prolog V5.1.
- PrologIA (F): Prolog IV.

- Lufthansa: Personalplanung nach Störungen.
  DAYSY, Cosytec CHIP.
- Hongkong International Terminals: Transport und Lagerung von Containern. ICL DecisionPower.
- Renault: Kurzfristige Fertigungsplanung.
  Bull CHIP-Derivat.
- Airbus: Layout der Pilotenkabine.
  Uni Hamburg.
- Siemens: Verifizierung von Schaltkreisen.
  IFProlog.
- Caisse d'epargne: Portfolio Management.
  PrologIA
- Sonst: Nokia, Daussault.
  SNCF 1700 Züge Gare Du Nord Paris Ilog.

Gelbe Seiten Layout: Anzeigen und Einträge.
A+E nahe, Konkurrenten weg, Seite voll.

Zahlenschloß, Kreuzworträtsel, Gleichungslösen.

Musikprobe - Lärmschutzvorschriften.

Constraint-Programmierung

Die Entwicklung

Seit 1962 Constraints-Netzwerke in der KI.

Seit 1972 Logikprogrammierung (Prolog).

Seit 1982 Constraint-Logikprogrammierung.

Seit 1990 Kommerzielle Anwendungen (C++).

Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it.

[Eugene C. Freuder, *Constraints Journal*, 1997]

Keine Anmerkung

Planung von drahtlosen Kommunikationssystemen

Die Herausforderung

ECRC (Siemens, ICL, Bull)
Siemens Private Netzwerke
Siemens Forschung und Entwicklung
RWTH Aachen, Inst. f. Kommunikationsnetze

- Lokale digitale Telekommunikation
- Braucht kleine Sendestationen
- Optimale Plazierung der Sender:
  - Vollständige Abdeckung
  - Minimale Senderanzahl

Anmerkung

PROJEKTLEITER! Industrieforschungszentrum.

1999: Deutschland 5 Millionen drahtlose Telefone verkaufen.

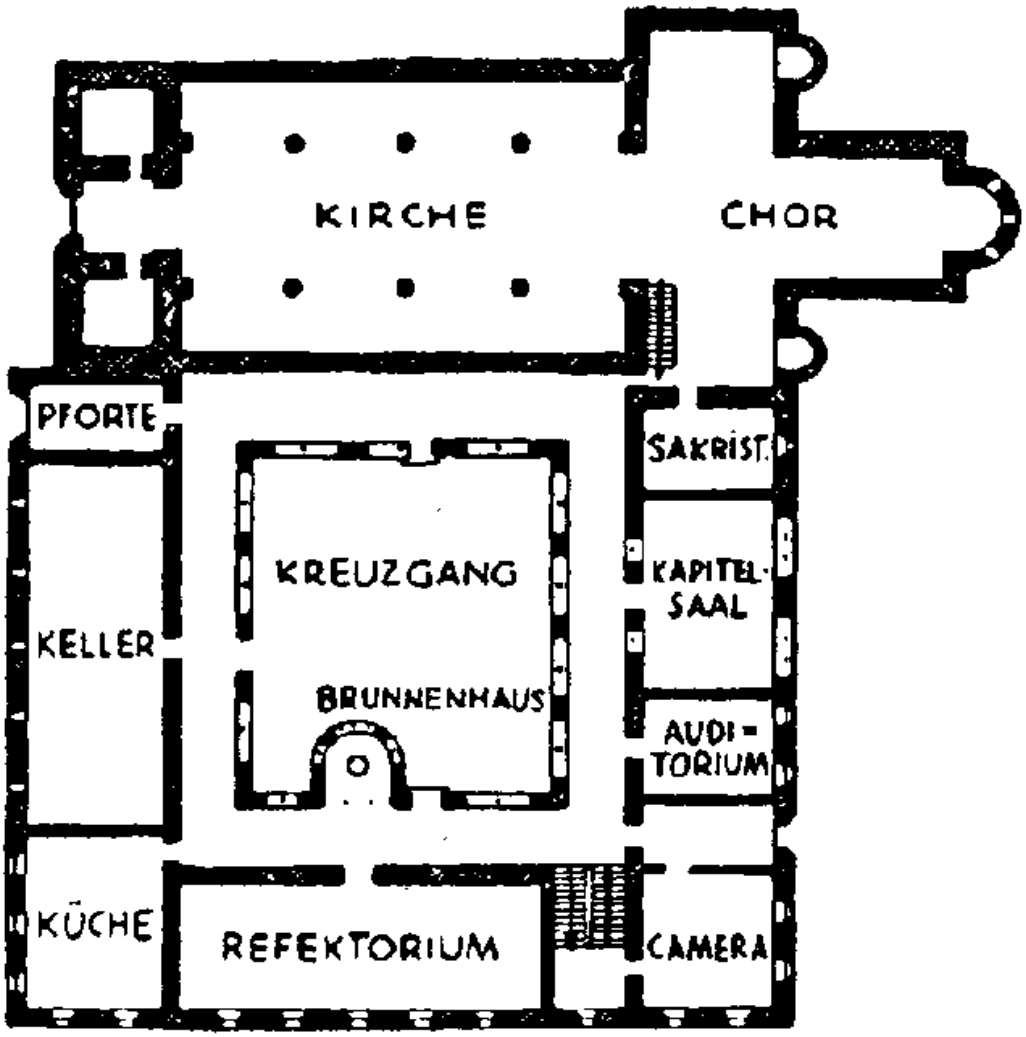
Flexibel, ohne aufwendige Montagearbeiten. Verbesserte Übertragungsqualität bei mehr Abhörschutz.

Ungelöstes Problem

Übliche Constraint-Ansätze funktionieren nicht:

- Unendlicher Wertebereich, mehrdimensional
- Nichtlineare Gleichungen
- Kontinuierliche und diskrete Funktionen
- Disjunktive geometrische Constraints

Die Funkwellenausbreitung im Gebäude muß berücksichtigt werden.


KIRCHE
CHOR
PFORTE
SAKRIST.
KREUZGANG
KAPITEL-
SAAL
KELLER
BRUNNENHAUS
AUDI-
TORIUM
KÜCHE
REFEKTORIUM
CAMERA

Anmerkung

CONSTRAINT: Sender in Funkzelle.

Die Funkwellenausbreitung im Gebäude muß berücksichtigt werden.

Die Funkwellenausbreitung wird beeinträchtigt:

- Dämpfung des Signals je nach Entfernung,
- Abschattung und Beugung durch Hindernisse,
- Mehrwegeausbreitung durch Reflexion.

Das Problem

Constraintlöser ist Blackbox:

Benutzer hat keinen Einblick in Implementierung.

Unterschiedliche Klassen von Anwendungen haben unterschiedliche Constraints.

- Anpassung ineffizient und unflexibel.
- Kombination von Constraintlösern schwierig.
- Fehlersuche und Programmanalyse schwierig.

Mindert Vorteile der Constraint-Programmierung.

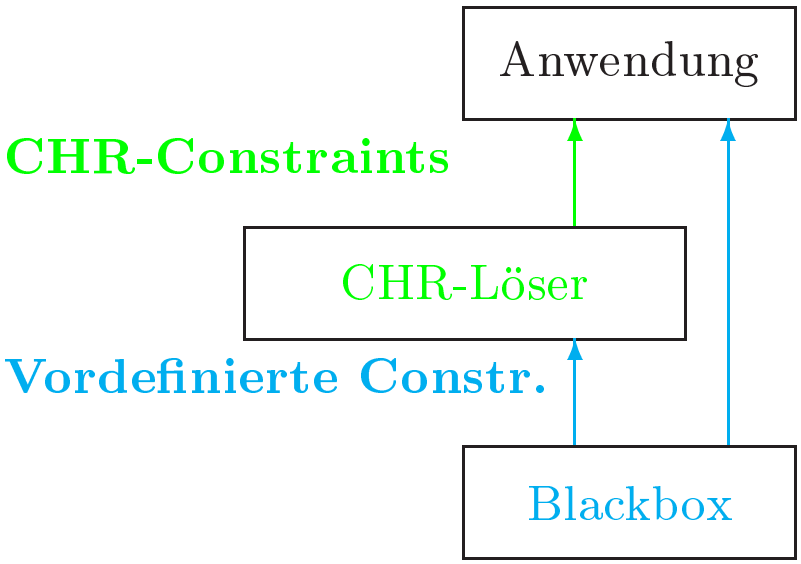
Anmerkung

Beispiel: Anforderungen Hardware-Verifikation versus Personalplanung.

Constraint Handling Rules (CHR)

Frühwirth, seit 1991

Höhere Sprache zur Spezifikation und Implementierung von Constraintlösern.
CHR-Constraintlöser sind offen, deklarativ, anpaßbar, komponierbar, debugbar, analysierbar.



Basissprache (Prolog, Lisp, Java...)

Anmerkung

Vordefinierte Constraints (z.B. auch Tests und Zuweisungen)

Inhalt

Constraint Handling Rules

- Beispiel
- Deklarative Semantik
- Operationale Semantik
- Programmanalyse
- Implementierungen
- Anwendungen
- Zusammenfassung

[Frühwirth, In *Constraint Programming: Basics and Trends*, Springer LNCS, 1995]

Keine Anmerkung

Beispiel Ordnungsrelation $\leq$

```
x≤x ⇔ true              % Reflexivität
x≤y ∧ y≤x ⇔ x=y         % Antisymmetrie
x≤y ∧ y≤z ⇒ x≤z         % Transitivität

a≤b ∧ b≤c ∧ c≤a         Anfrage
                        % Transitivität
   a≤c
                        % Antisymmetrie
         a=c
                        % Antisymmetrie
   a=b

a=b ∧ a=c               Antwort
```

Anmerkung

Constraint aus Zahlenschloß-Beispiel.

Linie zwischen Programm und Anfrage ziehen.

GLEICH BEISPIEL MACHEN!

Constraints sind (allquantorfreie) Konjunktionen von atomaren Constraints.

CHR-Regeln beschreiben wie Constraints sich zu neuen Constraints vereinfachen und wie Constraints andere Constraints propagieren.

Fortgesetzte Vereinfachung und Propagierung von Constraints führt schließlich zur Lösung der Constraints.

Regeln aus Inferenzregeln, Formeln, Ersetzungsregeln gewinnbar.

Hier Abstrakte Syntax verwendet.

Simplifikation entfernt Constraints, Propagierung nicht.

Kein Beweiser, sondern effizient Konjunktionen lösen.

Beispiel: Reflexivität mit Wächter.

Deklarative Semantik

Kopf	K	CHR-Constraints.
Wächter	W	vordefinierte Constraints.
Rumpf	R	vordefinierte und CHR-Constraints.

Simplifikationsregel

$K \Leftrightarrow W \mid R \qquad \forall\ (W \rightarrow (K \leftrightarrow \exists \bar{x}\ R))$

Propagierungsregel

$K \Rightarrow W \mid R \qquad \forall\ (W \rightarrow (K \rightarrow \exists \bar{x}\ R))$

Deklarative Semantik eines CHR-Programm P:

$\mathcal{P}$: Logische Leseweise der CHR-Regeln von P.

$\mathcal{T}$: Theorie für vordefinierte Constraints.

Anmerkung

Meta-Variablen stehen immer für Konjunktionen von Constraints.

Variablen $\bar{x}$ kommen *nur* in R vor.

Constraints als spezielle Prädikate der Prädikatenlogik erster Stufe.
Vordefiniert: *true*, (*false*,) = syntaktische gleichheit.
CHR-Constraints durch Regeln definiert und implementiert.

Der Wächter als Vorbedingung einer Regel impliziert dabei einen logischen Zusammenhang zwischen Kopf und Rumpf der Regel: Im Fall einer Simplifikationsregel ist der Zusammenhang eine logischen Äquivalenz zwischen Kopf und Rumpf, im Fall einer Propagierungsregel eine Implikation.

Mit Wächter:

```
∀X,Y (X=Y→(X≤Y↔true)) % Reflexivität
```

Operationale Semantik

Zustandsübergangssystem:

Zustand: Konjunktion von Constraints.

Anfangszustand = *Anfrage*.

Endzustand = *Antwort*.

Simplifizieren:

$$K' \wedge C \;\longmapsto\; (K = K') \wedge R \wedge C$$

wenn $(K \Leftrightarrow W \mid R) \in P$

und $\mathcal{T} \models \forall\, (C_V \rightarrow \exists \bar{x}((K = K') \wedge W))$

Propagieren:

$$K' \wedge C \;\longmapsto\; (K = K') \wedge R \wedge K' \wedge C$$

wenn $(K \Rightarrow W \mid R) \in P$

und $\mathcal{T} \models \forall\, (C_V \rightarrow \exists \bar{x}((K = K') \wedge W))$

$(K = K')$ steht für paarweises Gleichsetzen der Argumente der atomaren Constraints in K und K'.
C_V steht für die vordefinierten Constraints von C.

TAFEL?! Anmerkung

Endzustand: ($C =$*false* bzw.) keine Reduktionen mehr möglich.

K, W, R, C: Konjunktionen von Constraints.

Jede passende CHR-Regel-Kopie auswählbar.
committed-choice. verpflichtende (Regel-)wahl
Matching anstatt Unifikation (sonst incomplete).
Wächter zur Regelauswahl (da kein backtracking).

$\mathcal{T} \models \forall\ F$: Constraintlöser für vordefinierte Constraints überprüft Gültigkeit der Formel F.
Abkürzung $K = K'$. Parameterübergabe
$\bar{x}$ frische Variablen aus Regelkopie.

Logische Formulierung des MATCHING und WÄCHTER-Tests.

Vereinfachen von vordefinierten Constraints implizit.

Konjunktion assoziativ und kommutativ, nicht idempotent.

$\forall$ Allabschluß.

W im Folgezustand nicht dabei.

Terminierung der Propagierungsregeln nicht dargestellt.

Übereinstimmung der Semantiken

A Anfrage, B Zustand, C Antwort.

Lemma 1 (Äquivalenz von Zuständen) Wenn der Zustand B in einer Ableitung von A vorkommt, dann gilt

$$\mathcal{P}, \mathcal{T} \models \forall\ (A \leftrightarrow \exists \bar{x} B).$$

Satz 1 (Korrektheit) Wenn C eine Antwort von A ist, dann gilt

$$\mathcal{P}, \mathcal{T} \models \forall\ (A \leftrightarrow \exists \bar{x} C).$$

Satz 2 (Vollständigkeit) A habe mindestens eine endliche Ableitung. Wenn $\mathcal{P}, \mathcal{T} \models \forall\ (A \leftrightarrow \exists \bar{x} C)$, dann hat A eine Antwort C', sodaß

$$\mathcal{P}, \mathcal{T} \models \forall\ (\exists \bar{x} C \leftrightarrow \exists \bar{y} C').$$

[Frühwirth, Abdennadher, Meuss, *Constraint Journal*, Kluwer, 1998]

Anmerkung

Ableitung: Folge von Reduktionen.

$\exists \bar{x}, y$: Alle Variablen, die nicht in A vorkommen.

Besser als herkömmliche CLP-Sprachen.

Vollständigkeits- und Korrektheitssätze des CHR-Kalküls basieren vor allem darauf, daß Reduktionen für CHR die logische Äquivalenz von den Zuständen erhalten:

Induktionsbeweis über Länge der Ableitungen plus logische Leseweise der Reduktionsregeln.

Aus diesem Lemma folgt unmittelbar, daß alle Zustände in einer Ableitung einer Anfrage logisch äquivalent sind. Aus dem Lemma folgt auch die Korrektheit.

Der Vollständigkeitssatz gilt nicht, wenn A keine endlichen Ableitungen hat:

$\mathtt{p} \Leftrightarrow \mathtt{p}$

Sei A die Anfrage p.
Es gilt $\mathcal{P}, \mathcal{T} \models \mathtt{p} \leftrightarrow \mathtt{p}$.
Aber A hat nur eine unendliche Ableitung.

Vollständigkeit schwach für Antwort *false*:

$\mathtt{p} \Leftrightarrow \mathtt{q}$
$\mathtt{p} \Leftrightarrow \mathit{false}$

Es gilt $\mathcal{P}, CT \models \neg \mathtt{q}$. Aber q hat nur Antwort q.

Inhalt

Constraint Handling Rules

- Beispiel
- Deklarative Semantik
- Operationale Semantik
- **Programmanalyse**
- Implementierungen
- Anwendungen
- Zusammenfassung

Keine Anmerkung

Programmanalyse

Terminierung

CHR-Constraintlöser terminieren meist mit einfachen Termordnungen (linear polynomial interpretations).

[Frühwirth, *Habilitationsschrift*, 1998]

Konfluenz

Eine Anfrage hat immer die gleiche Antwort, egal in welcher Reihenfolge die Regeln angewandt werden.

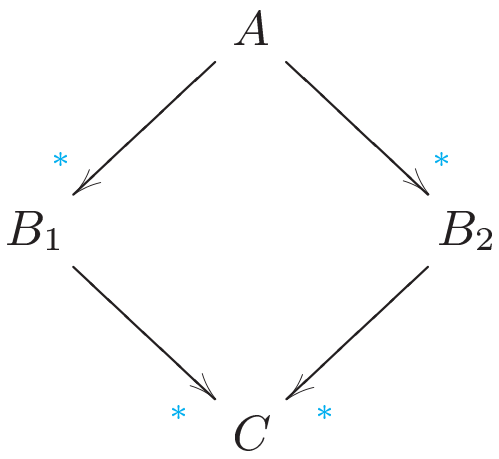
Anmerkung

• Monotonieeigenschaften
Erhalt der möglichen Reduktionen
Kontextunabhängigkeit, Modularität, Analyse, Nebenläufigkeit
• Terminierung
Viel Propagierung: Compileroptimierung stellt Terminierung sicher (z.B. $\leq$).
• Konfluenz
entscheidbares, notwendiges und hinreichendes Kriterium für lokale Konfluenz
in Bool fehlten regeln, gaussche elimination koennte man richten
• Konsistenz
Widerspruchsfreiheit, folgt aus Konfluenz

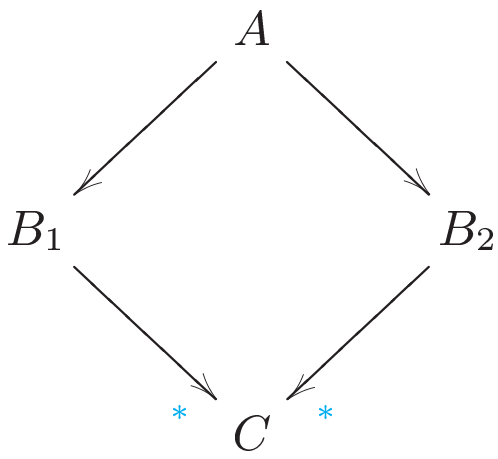
Wie in Termersetzungssystemen? Ja, aber:
• Propagierungsregeln
schwierig, token store, Diss Slim
• Globaler Kontext (Constraintspeicher)
vordefinierte Constraints mitfuehren
• Konjunktion assoziativ und kommutativ
• Lokale Variablen im Rumpf
normalform mit varianten-test, variablen von A mitfuehren
• Mehrfachvorkommen von Variablen in Köpfen
stoert uns nicht, wenn normalform

Konfluenz

Konfluenz



Lokale Konfluenz



Terminierung $\rightarrow$ (Konfluenz $\leftrightarrow$ Lokale Konfluenz)
[Newman, *Annals of Math*, 1942]

Lemma 2 (Monotonie) Wenn die Reduktion $A \longmapsto B$ möglich ist, dann auch $A \wedge D \longmapsto B \wedge D$.
[Frühwirth, Abdennadher, *Constraint-Programmierung*, Springer Verlag, 1997]

Anmerkung

A, B, C, D sind Zustände, d.h. Constraints.

Probleme bei der Adaptation von Ergebnissen aus Termersetzungssystemen:

- ~~Blackbox-Constraintlöser~~
- ~~Propagationsregeln~~

Monotonie entspricht *Verträglichkeit* bei TRS.

Kritisches Paar

Satz 3 Ein CHR-Programm P ist *lokal konfluent* gdw. seine kritischen Paare zusammenführbar sind. [Frühwirth, Abdennadher, Meuss, *Constraint Journal*, Kluwer, 1998]

```
x≤y ∧ y≤x ⇔ x=y,    % Antisymmetrie
x≤y ∧ y≤z ⇒ x≤z     % Transitivität
```

```
x≤y ∧ y≤x
```

```
% Transitivität          % Antisymmetrie
x≤y ∧ y≤x ∧ x≤x          x=y
% Antisymmetrie
x=y ∧ x≤x
```

Anmerkung

Überlappen zweier Regeln ergibt Ausgangszustand.

Ausgangszustand hat die Köpfe der beiden Regeln, wobei ein oder mehrere Kopfconstraints miteinander gleichgesetzt wurden, und die Wächter der beiden Regeln.

Linie zwischen Programm und Anfrage ziehen.

Beispiel: Reflexivität fehlt.

BLITZ

Nutzen des Konfluenztests

Theorie:
Impliziert Konsistenz und verbessert Übereinstimmung der Semantiken.

Praxis:
Antwort auch unabhängig von Reihenfolge der Constraints. Konfluenztest enthüllt Fehler.

Implementierte CHR-Constraintlöser sind meist konfluent.
[Frühwirth, *Habilitationsschrift*, 1998]

Anmerkung

CHR Implementierungen

Eclipse	Interpreter	Frühwirth	ECRC	1991
Lisp	Interpreter	Herbig	DFKI Kaiserslautern	1993
Eclipse	Compiler	Frühwirth, Brisset	ECRC	1994
Oz	Interpreter	Smolka	DFKI Saarbrücken	1995
Sicstus	Compiler	Holzbaur, Frühwirth	LMU, Uni Wien	1997
Java	Emulator	Frühwirth et al.	LMU	1999

Keine Anmerkung

CHR Constraintlöser (Auswahl)

- Terme, Bäume, Listen, Mengen
- Boolesche Algebra
- Endliche Wertebereiche
- Lineare und Nichtlineare (Un-)Gleichungen
- Kanten-, Pfadkonsistenz
- Zeitliches und räumliches Schließen

[Frühwirth, *Journal of Symbolic Computation*, Academic Press, 1996]

[Frühwirth, in *Executable Modal and Temporal Logics*, Springer LNAI, 1995]

- Terminologisches Schließen

[Frühwirth, P. Hanschke, in *Principles and Practice of Constraint Programming*, MIT Press, 1995]

Anmerkung

Dutzende Constraintlöser existieren.

Ca. Hälfte von mir geschrieben.

Datenstrukturen sowie non-standard Löser.

CHR Anwendungen (Auswahl)

LMU, Institut für Informatik

Frühwirth et al

- Münchener Mietspiegel Online
  [Frühwirth, Abdennadher, *Künstliche Intelligenz*, Interdata Verlag, 1997]
- Instituts-Stundenplanung
- BMW Statecharts Simulation
  [LMU, *Constraint-Arbeitsgruppe Faltblatt*, 1998]

Anmerkung

LMU CHR-Code weniger als eine Seite jeweils.

Mietspiegel 10.000e Benutzer.

BMW: Software-Spezifikation, Elektronik bis zu 30% Anteil an Autokosten.

CHR Anwendungen (Auswahl)

Anderswo

A. Frisch, Univ. of York
Verschiedene Resolutionsverfahren

M. Rusinowitch, Univ. Nancy
Lineare und Nichtlineare Gleichungen

A. Wolf, GMD FIRST, Berlin
ConTes System-Konfiguration

E. Sandewall, Univ. Linköping
PMON Implementierung

S. Bressan, MIT
Internet Informations-Integration

P. Stuckey, Univ. Melbourne
Ausführbare Z-Spezifikationen

...etc...

Anmerkung

Nicht-LMU Beispiele alle 1996 PUBLIZIERT (Dumbill 1995, Bressan 1997).

Michael Rusinowitch: CHR auch zur Koordination

Wolf: Innovationsmesse Leipzig 1996

Lintschöping: Zeit-Aktions-Logik

Planung von drahtlosen Kommunikationssystemen

Der Constraint-Ansatz

• Simulation

- Gebäude modellieren: Wände und Material
- Funkwellenausbreitung durch Pfadverfolgung

• Optimierung

- Constraintlösen (Schneiden der Funkzellen)
- Suche (Wahlweises Gleichsetzen von Sendern)
- Optimierung (Branch and Bound)

[Frühwirth, Molwitz, Brisset, *IEEE Expert Magazine*, IEEE Press, 1996]

Anmerkung

Projektleiter!

Aachen: Prof. Dr. Walke, Hr. Hußmann

1999: Deutschland 5 Millionen drahtlose Telefone verkaufen.

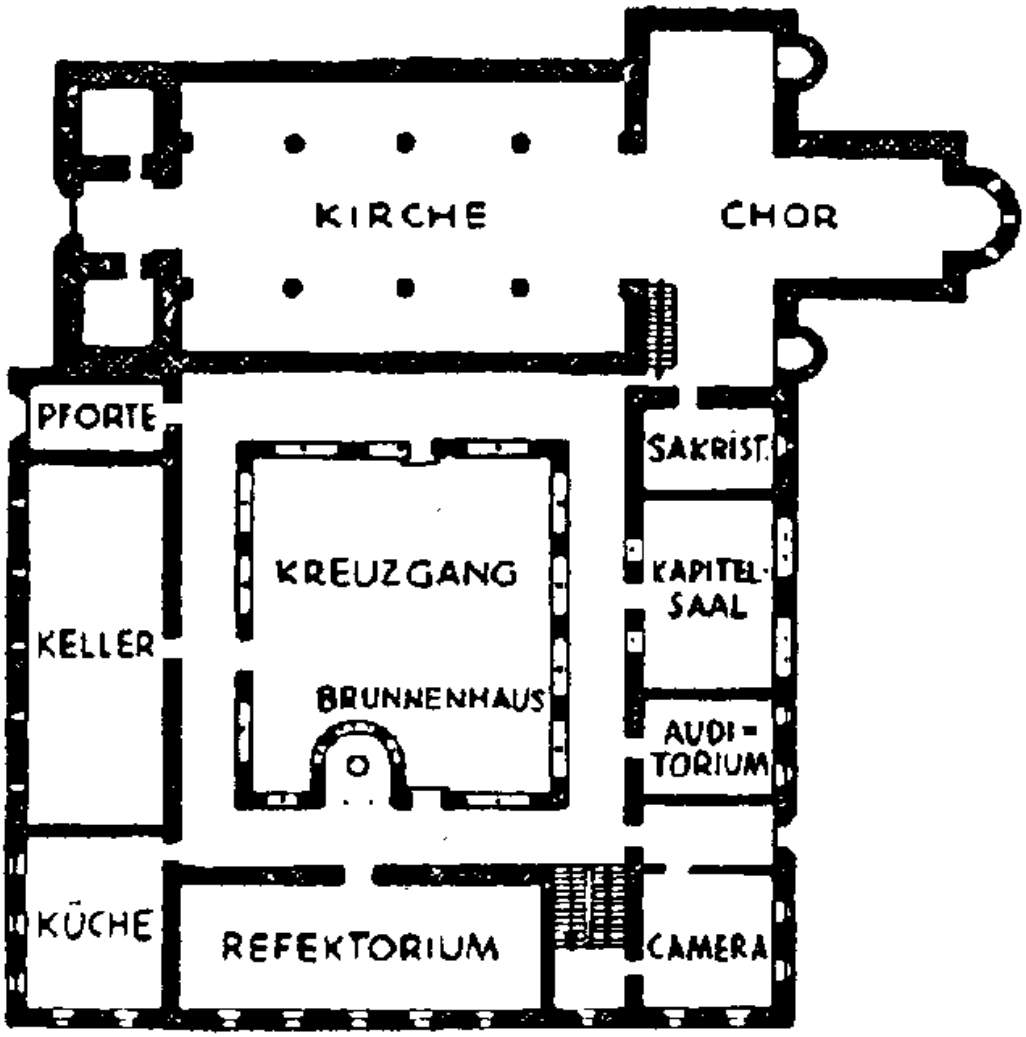
Flexibel, ohne aufwendige Montagearbeiten. Verbesserte Übertragungsqualität bei mehr Abhörschutz.

Weltweit gemeinsam mit AT&T WISE die ersten. IEEE Magazine USA innovative Anwendung in der Telekommunikation ausgewählt. (Technologietransfergespräche mit Hagenuk.)

Jeweils ca. eine Seite Code, weniger als 10 % des gesamten programms.

Das Constraintlogikprogramm für POPULAR ist nur 4000 Zeilen lang, inklusive der Grafik für die Benutzerschnittstelle, die etwa die Hälfte des Programms ausmacht, Rest Simulation. WISE 7500 Zeilen C.

Existierende Constraint-Ansätze funktionieren nicht:
- Unendlicher Wertebereich, mehrdimensional
- Nichtlineare Gleichungen
- Kontinuierliche und diskrete Funktionen
- Disjunktive Constraints


KIRCHE
CHOR
PFORTE
SAKRIST.
KREUZGANG
KAPITEL-
SAAL
KELLER
BRUNNENHAUS
AUDI-
TORIUM
KÜCHE
REFEKTORIUM
CAMERA

Anmerkung

APPROXIMATION durch Rechtecke orthogonal zum Koordinatensystem.

SIMULATION: Dazu wird eine fiktive Funkwelle in in einer ausreichenden Anzahl von Richtungen (mehr als hundert) untersucht. Der Ausbreitungspfad wird durch das gesamte Gebäude hindurch verfolgt.

Entlang des Pfades werden alle Schnittpunkte mit Wänden und Decken festgestellt und die zugehörigen Einfügungsdämpfungen registriert. Dabei werden die Antennendämpfung in Pfadrichtung, die entfernungsabhängige Dämpfung und die registrierten Einfügungsdämpfungen längs des Pfades bis zum zulässigen Grenzwert des Pfadverlustes aufaddiert. Danach werden durch binäre Suche entlang des Pfades ausreichend genau die Koordinaten des zugehörigen Grenzwertpunktes ermittelt. Die Hülle der Grenzpunkte eines Testpunktes beschreibt dann die Funkzelle. Eine Funkzelle umfaßt normalerweise mehrere Räume in mehreren Stockwerken.

Constraintlöser

```
% Sender in Rechteck

nicht_leer @
S in (A,B)#(C,D) ==> A<C, B<D.

durchschnitt @
S in (A1,B1)#(C1,D1), S in (A2,B2)#(C2,D2) <=>
      A = max(A1,A2), B = max(B1,B2),
      C = min(C1,C2), D = min(D1,D2),
      S in (A,B)#(C,D).
```

Anmerkung

Konkrete Syntax! Regelnamen!

Existierende Ansätze funktionieren nicht.

- Unendlicher Wertebereich, mehrdimensional
- Nichtlineare Gleichungen
- Kontinuierliche und diskrete Funktionen
- Disjunktive Constraints

In einer ersten Implementierung des Constraintlösers hat man sich auf ein Stockwerk beschränkt und jede Funkzelle durch ein eingeschriebenes Rechteck approximiert. Rechtecke sind orthogonal zum Koordinatensystem und jeweils durch den linken unteren und rechten oberen Eckpunkt definiert. Für jede Funkzelle wird das Constraint `Sender in Rechteck` eingeführt, wobei `Sender` geometrisch gesehen ein Punkt ist, der im Rechteck `Rechteck` liegen muß. Die Koordinaten für einen Punkt haben die Form `(X,Y)`.

Die `durchschnitt`-Regel ersetzt zwei Constraints, die besagen, daß derselbe Sender in zwei verschiedenen Rechtecken liegt, durch ein Constraint, wobei das neue Rechteck der Schnitt der beiden ursprünglichen Rechtecke ist. `A<C, B<D` stellt sicher, daß ein Rechteck nicht leer ist.

```
% CHR-Constraintloeser

S in nil <=> false.

S in L1, S in L2 <=>
    durchschnitt_quader(L1, L2, L3), S in L3.

 durchschnitt_quader(L1, L2, L3) <=>
    setof(R3,(waehle(R1, L1), waehle(R2, L2),
              durchschnitt_quader(R1,R2,R3)),
          L3).

% Prolog Suche

waehle(X,cons(X,L)):- true.
waehle(X,cons(Y,L)):- waehle(X,L).

sender_gleichsetzen(nil):- true.
sender_gleichsetzen(cons(S,L)):-
    (waehle(S,L) ; true),
    sender_gleichsetzen(L).

% Prolog Optimierung

bb(C,M,L):-
    constraints(C,S), max_sender(S,M),
    sender_gleichsetzen(S), sender_anzahl(S,N-1),
    (bb(C,N-1,L) ; L=S).
```

Anmerkung

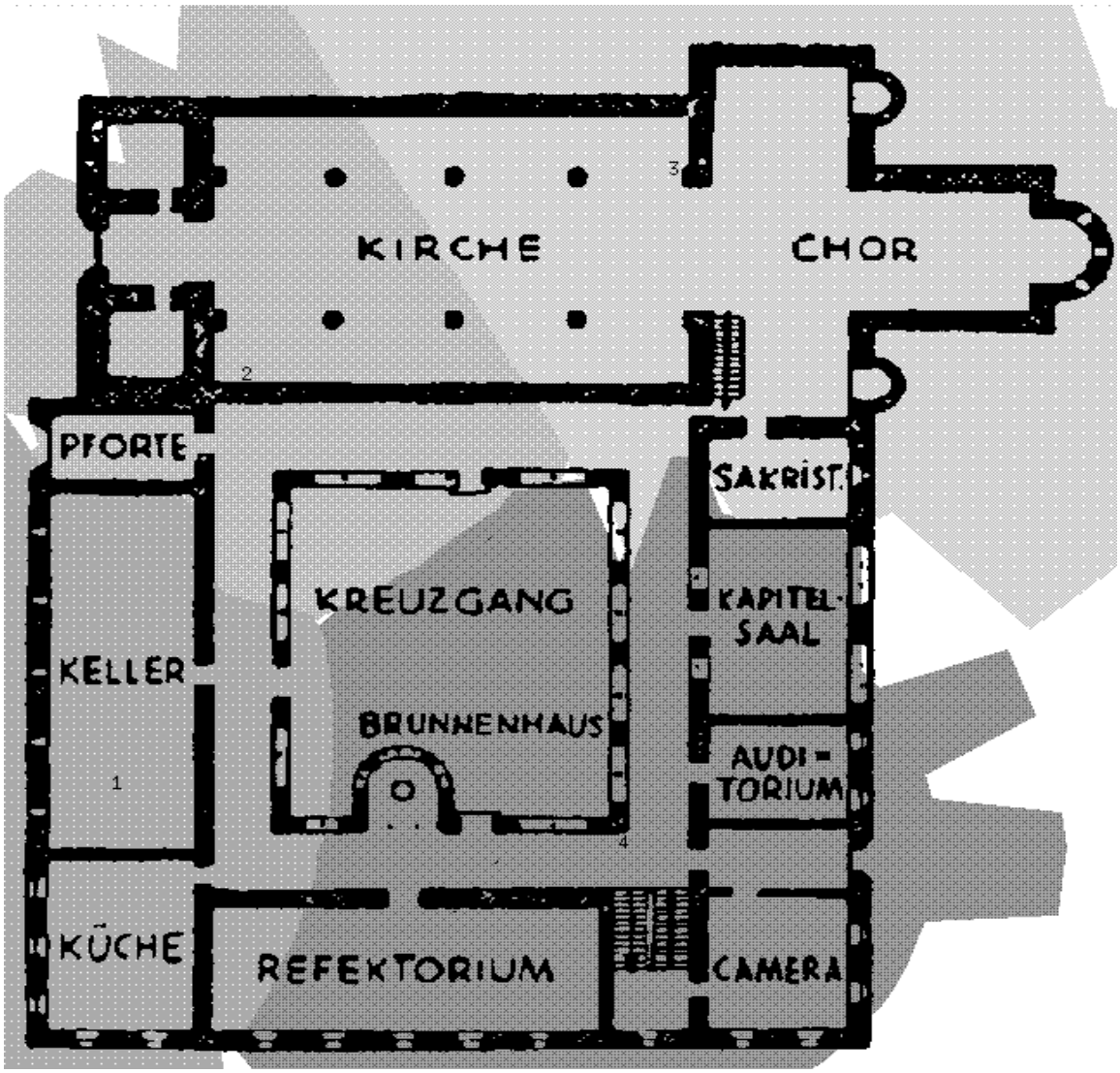
PROLOG code mit :- !

Listen mit cons/1 und nil/0!

In 10 Minuten konnte man diesen Constraintlöser so erweitern, daß er mit Vereinigungen von Rechtecken funktioniert. Damit können die Funkzellen mit beliebiger Genauigkeit approximiert werden. Die Vereinigung von Rechtecken entspricht dem disjunktiven Constraint `S in (R1` $\vee$ `S in R2` $\vee$ `...` $\vee$ `S in Rn`. Es wird kompakter als
`S in cons(R1,cons(R2,...,Rn))` geschrieben.

Das Erweitern des Constraintlösers auf den dreidimensionalen Fall mit mehreren Stockwerken ist ähnlich einfach machbar. Jedes Rechteck wird zusätzlich mit einer Stockwerkskennung versehen, und nur Rechtecke in der gleichen Ebene (im gleichen Stockwerk) werden miteinander geschnitten.

Der Constraintlöser kann auch modifiziert werden, so daß er mit anderen geometrischen Objekten als Rechtecken arbeitet, indem man die Listenelemente `(A,B)#(C,D)#(E,F)` durch die Beschreibung der Objekte ersetzt und die Schnittberechnung in
`durchschnitt_rechteck/3` entsprechend adaptiert.


KIRCHE
CHOR
PFORTE
SAKRIST.
KREUZGANG
KAPITEL-
SAAL
KELLER
BRUNNENHAUS
AUDI-
TORIUM
KÜCHE
REFEKTORIUM
CAMERA
1
2
3
4

Anmerkung

Das Beispiel zeigt das Ergebnis der Versorgung eines mittelalterlichen Klosters mit vier Sendern. Ein Abdeckungsgebiet ist jeweils dem stärksten Sender zugeordnet.

Zusammenfassung

Constraint Handling Rules - Fortschritt in Theorie und Praxis des Constraintlösens:

• Ausführbare Spezifikation und Rapid Prototyping. Extrem kurze, effiziente Programme.

• Weitgehende Übereinstimmung zwischen deklarativer und operationaler Semantik.

• Entscheidbarer Konfluenztest enthüllt potentielle Fehler und garantiert Konsistenz.

• Flexibler als Blackbox Constraintlöser und Spezialwerkzeuge. Constraintlöser können modifiziert und kombiniert werden.

• Ermöglicht auch neue Anwendungen mit neuen Constraints (z.B. Senderplazierung).

Keine Anmerkung

Mit anderen Worten...

For the theoretician meta-theorems can be proved and analysis techniques invented once and for all; for the implementor different constructs can be implemented once and for all; for the user only one set of ideas need to be understood, though with rich (but disciplined) variations (constraint systems).

[P. van Hentenryck, V.A. Saraswat, Strategic Directions in Constraint Programming, *ACM Computing Surveys*, 1996]

Keine Anmerkung

UND WEITERE FOLIEN KOMMEN JETZT:

(in schwarz-weiß)

Anmerkung

Andere Lösungsvorschläge

- Demons, Forward Rules, v. Hentenryck, 1988
- Constr.-Konstruktoren, v. Hentenryck, 1991
- Meta-Constraints, Older, 1992
- Attributierte Variablen, Holzbaur, 1992
- Index-Constraints, Codognet, 1994

Vorteile Constraint Handling Rules

- Höhere Programmiersprache
- Deklarative Semantik
- Theoretische Resultate
- Konkrete Anwendungen
- Programmanalyse
- Neue Constraints möglich

...

Anmerkung

Problem: Keine neuen Constraintlöser schreibbar, nur begrenzte Erweiterungen.

Constraintsystem fix, meist endliche wertebereiche!

Attributierte Variablen: Assembler-artig.

Guarded Rules, Elan, Maude NACH CHR!

CC, TRS, OPS5: Keine deklarative Semantik!

Verwandte Arbeiten

• Forward Rules, van Hentenryck, 1988
einkoepfig, keine waechter
• Guarded Rules, Smolka, 1993
einkoepfige simplifikation, erweitern CLP
• CC-Framework, Saraswat, 1989
einkoepfig, allgemeineer sprachansatz
• Termersetzungssysteme
nicht wirklich einbettbar, siehe konfluenz
- Betrand, Leler, 1988
- Elan, Kirchner, 1992
- Maude, Meseguer, 1992
• Produktionsregeln (OPS5)
zu allgemein, Propagierung nicht direkt

Beispiel Terminierung

Gerade Zahlen:

$$\texttt{gerade(X)} \Leftrightarrow \texttt{X=s(Y)} \mid \texttt{Y=s(Z)} \wedge \texttt{gerade(Z)}$$

Anfragen und Antworten:

`gerade(N) ?`
`gerade(N)`

`gerade(s(N)) ?`
`N=s(N1)` $\wedge$ `gerade(N1)`

`gerade(N)` $\wedge$ `gerade(s(N)) ?`
`gerade(N)` $\wedge$ `N=s(N1)` $\wedge$ `gerade(N1)`
`gerade(s(N1))` $\wedge$ `gerade(N1)`
...ad infinitum...

Anmerkung

`?` symbolisiert Anfrage.

Vergleich

CHR	Prolog	TRS
Spracherweiterung	Allg. Sprache	Allg. Sprache
Relational	Relational	Funktional
Simplifikationsregeln	Klauseln	Ersetzungsreg.
Propagierungsregeln	-	-
Mehrköpfige Regeln	-	(-)
committed-choice	backtracking	committed-choice
Matching	Unifikation	Matching
Wächter	-	(Bedingungen)

Anmerkung

committed-choice bedingt:

Matching anstatt Unifikation (sonst incomplete).

Wächter zur Regelauswahl (da kein backtracking).

Constraint-Programmierung

```
%    D: Darlehen, derzeitige Schuld
%    M: Dauer der Rueckzahlung in Monaten
%    Z: Zinssatz pro Monat
%    R: Monatliche Rueckzahlungsrate
%    S: Rest-Schuld nach M Monaten

darlehen(D, M, Z, R, S):-
      M = 0,
      D = S.

darlehen(D, M, Z, R, S):-
      M  > 0,
      M1 = M - 1,
      D1 = D + D*Z - R,
      darlehen(D1, M1, Z, R, S).
```

Anmerkung

Das Ziel `darlehen(100000,360,0.01,1025,S)` liefert die Antwort `S=12625.90`. Man kann sich nun fragen, welchen Betrag man in diesem Zeitraum zu diesen Konditionen vollständig zurückzahlen kann: Das Ziel `darlehen(D,360,0.01,1025,0)` liefert `D=99648.79`, einen nur geringfügig niedrigeren Betrag (Zinseszinseffekt).

Umgekehrt können wir uns fragen, wieviel Monate lang wir die `100000` zurückzahlen müssen. Das Ziel `S=<0, darlehen(100000,T,0.01,1025,S)` liefert `T=374` und `S=-807.96`.

Wie verhalten sich allgemein Darlehenshöhe und Rückzahlungsbetrag zueinander, wenn man zu obigen Konditionen nach 30 Jahren schuldenfrei sein will? Das Ziel `darlehen(D,360,0.01,R,0)` liefert die intensionale Antwort `R=0.0102861198*D`.

Beliebig weit läßt sich aber die Verallgemeinerung des Zieles nicht treiben: Sollte ein Ziel den Zinssatz `Z` offen lassen, so wird nach mehr als einem Rekursionschritt die Gleichung `D1 = D + D*Z - R` nicht mehr linear, da das neue `D` dann ebenfalls nicht mehr determiniert ist.

Marktüberblick nach Branchen

• Transportwesen (British Airways, Lufthansa, Delta Airlines, SNCF)
• Automobil- und Flugzeugindustrie (Ford, Renault, Peugeot Citroën, Daimler-Benz, Michelin, Lockheed Martin, Airbus)
• Elektronik (IBM, Alcatel, Philips, Hewlett-Packard, Whirlpool, Bosch, Motorola, Thomson, Siemens, Nokia)
• Telekommunikation (France Telecom, AT&T)
• Finanzwesen (Banken und Versicherungen wie Citicorp, National Westminster Bank)
• Energieversorgung (Shell)
• Nahrungsmittelindustrie (Uncle Ben's)
• Handel (El Corte Ingles)
• Chemie (Rhone Poulenc)
• Militär (Dassault)

Keine Anmerkung

Anwendungsgebiete

- Modellierung,
- (ausführbare) Spezifikation,
- Design,
- Synthese,
- Simulation,
- Verifikation,
- Fehlerdiagnose

von elektronischen, elektrischen, mechanischen und Software-Komponenten sowie industriellen Abläufen.

Keine Anmerkung

Anwendungsgebiete

- Produktions-,
- Personal-,
- Finanz-,
- Verkehrs-,
- Netzwerk-
- Ressourcen-
- -planung,
- -logistik,
- -management,

- Transport- und Plazierungsoptimierung,
- Design,
- Konfiguration,
- Layoutgenerierung.

Keine Anmerkung

Boolesche Algebra

Wahrheitswerte 0 und 1, logische Verknüpfungen.

```
und(X,Y,Z) <=> X=0 | Z=0.
und(X,Y,Z) <=> X=1 | Y=Z.
und(X,Y,Z) <=> Z=1 | X=1,Y=1.
und(X,Y,Z) <=> X=Y | Y=Z.
```

E. Dumbill (A. Frisch), Univ. of York, 1995: Automatisches Beweisen, Varianten des Davis Putnam Verfahrens.

Anmerkung

KONKRETE SYNTAX!

Boolesche Constraints leiten sich aus der Booleschen Algebra (auch: Schaltalgebra) ab. Die Constraints des Constraintsystems B sind logische Formeln über Variablen, die nur die Werte 1 oder 0 annehmen können, z.B. $(X \sqcup Y =_B 1)$. Diese Booleschen Variablen werden in der Literatur auch als binäre (Schalt-)Variablen bezeichnet.

Die Signatur von B enthält die (Booleschen) Konstanten 1 und 0 (sie stehen logisch für die Wahrheitswerte *wahr* und *falsch* bzw. für die Spannungsstufen *H(igh)* und *L(ow)* digitaler Schaltungen), logische Operatoren als Funktionssymbole, z.B. die einstellige Negation $-$ und die zweistelligen Verknüpfungen $\sqcap$ (und), $\sqcup$ (oder) und $\oplus$ (exklusives Oder), sowie die Constraintsymbole $=_B$, $true$ und $false$.

Terminologisches Schließen

Terminologie: Konzepte und Rollen mit logischen Verknüpfungen.

```
parent isa human and every child is human.
father isa parent and man.
```

```
I:C  <=> (C isa S), I:S.
I:S and T  <=> I:S, I:T.
I:nota S, I:S  <=> false.
I:every R is S, (I,J):R  ==> J:S.
```

A. Wolf, GMD FIRST, Berlin, 1996:
ConTeS: Konfiguration technischer Prozesse mit TRLC: ausführbare Spezifikationssprache.

Anmerkung

Pfadkonsistenz

Approximationsverfahren aus der Künstlichen Intelligenz.

```
c(I,K,C1),c(K,J,C2) ==>
      composition(C1,C2,C3),  c(I,J,C3).
c(I,J,C1),c(I,J,C2) <=>
      intersection(C1,C2,C3), c(I,J,C3).
```

F. Toledo, Univ. Castellon, Spanien, 1996:
Qualitatives Räumliches Schließen nach Freksa und Zimmermann (oben, links vorne, rechts unten).

Anmerkung

Zeitliches Schließen

Absolute und relative Position von Zeitpunkten und Zeiträumen (5 Uhr, 5 Stunden lang, vor, während, zwischen, in 5 Minuten).

```
% Instanz der Pfadkonsistenz
```

M. Bjareland (E. Sandewall), Univ. Linköpping, 1996:
PMON, Logik zur Modellierung dynamischer Systeme (Beobachtungen, Aktionen, Zeitconstraints).

Anmerkung

Endliche Wertebereiche

Lineare Arithmetik über endliche Ausschnitte der ganzen Zahlen. Z.B. Intervallpropagierung.

```
X::N1..M1, X::N2..M2 <=>
   X::max(N1,N2):min(M1,M2).
X≤Y, X::NX..MX, Y::NY..MY ==>
   X::NX:MY, Y::NX:MY.
X+Y=Z, X::NX..MX, Y::NY..MY ==>
   Z::NX+NY:MX+MY, X::NZ-MY:MZ-NY,...
```

M. Duerst, ETH Zürich, 1996:
Analyse und Generierung von Kanji-Schriftzeichen.

Anmerkung

(Nicht-)Lineare (Un)gleichungen

Variablenelimination.

```
[A1*X|P1]=B1, P=B2 <=>
   % Kommt Variable X in P vor?
   kommt_vor_in(X,P) |
   % Entferne A2*X aus P, erhalte so P2
   entferne(A2*X,P,P2),
   % loese lokal nach erster Variablen X auf
   multipliziere_poly_konst(B1-P1,A2/A1,B3-P3),
   % ersetze P3-B3 fuer X in zweiter Gleichung
   addiere_poly_poly(B3-P3,P2-B2,B4-P4),
   % lasse  erste Gleichung unveraendert
   [A1*X|P1]=B1,
   % neue zweite Gleichung ohne Variable X
   P4=B4.
```

E. Monfroy (M. Rusinowitch), Nancy, 1996:
GroAk: lineare (CHR) mit nichtlinearer Variablenelimination (Gröbner Basen).

Anmerkung

Die Variablen `P, P1, P2, P3, P4` stehen für Polynome, `B, B1, B2, B3, B4` für Konstanten und `A1`, `A2` für Koeffizienten.

In der Regel `eliminiere` gibt es zwei Gleichungen mit Vorkommen von X, $a_1 * X + p_1 = b_1$ und $a_2 * X + p_2 = b_2$. Man macht X in der ersten Variable explizit und multipliziert es mit a_2 ($a_2 * X = (b_1 - p_1) * (a_2/a_1)$), damit es in die zweite Gleichung eingesetzt werden kann: $(b_1 - p_1) * (a_2/a_1) + (p_2 - b_2) = 0$.

Wiederholte Anwendung von `eliminiere` entfernt wieder jedes Vorkommen einer "führenden" Variablen bis auf eines und führt so zur gewünschten Normalform.

```
                  [1*X,1*Y]=7, [1*X,-1*Y]=3
↦eliminiere       [1*X,1*Y]=7, [2*Y]=4
↦eliminiere       [1*X]=5, [2*Y]= 4   ◇
```

Constraintbasierte Stundenplanung

Für das Institut der Informatik an der LMU.

Wie wird geplant?

- Informationen sammeln: neue Angebote und Wünsche der Lehrer
- Entwurf des ersten Vorschlags auf Grundlage des Plans vom Vorjahr:
  - alte Angebote bleiben fest
  - freigewordene Plätze werden bevorzugt vergeben
- Bewertung des Entwurfs durch Lehrer und Sammlung neuer Wünsche, usw.
- Planung ist inkrementeller Prozess
- Maxime: *So wenig wie möglich ändern!*
  $\rightarrow$ vereinfacht Planung und sichert Akzeptanz des neuen Plans

Anmerkung

Noch keine Raumplanung!

Stundenplan Constraints

- *nicht überschneiden*
  - Keine Überschneidungen für Veranstaltungen eines Lehrers!
  - Keine Überschneidungen von Veranstaltungen einer LFE!
  - Keine Überschneidungen von Veranstaltungen eines Semesters des Grundstudiums!
- *(nicht) zwischen*
  - Veranstaltungen zwischen 9 und 18 Uhr
- *mindestens ein Tag (eine Stunde) dazwischen, nicht am gleichen Tag*
- *anschließend*

Unterscheidung von *harten* und *weichen* Constraints durch Gewichte.

Anmerkung

Alle Constraints der Anwendung können durch gewichtete Intervall-Constraints ausgedrückt werden.

Gewichtete Intervall-Constraints

- **domain**(C, S, D): Veranstaltung C beginnt um S aus D
- **in**(C, L, W):
  - $W = 1.0$: Veranstaltung C darf zu keiner Stunde nicht in L beginnen
  - $W < 1.0$: Erhöhung der Bewertung für die Stunden in L
- Initialisierung: Für jede Veranstaltung ein **domain**-Constraint mit

$$D = [0 : 0.0, \ldots, 167 : 0.0]$$

(7 Tage zu 24 Stunden)

Keine Anmerkung

Modellgenerierung mit Constraints

Ausgangspunkt: Satchmo

- Bereichsbeschränkte Klauseln
- Vorwärtsschließen
- Splitting

CPUHR-Kalkül kann auch Theorien behanden, die Formeln folgender From haben:

$$\forall \overline{X}(b_1 \wedge \cdots \wedge b_l \rightarrow \exists \overline{Y}(h_1 \vee \cdots \vee h_m \vee c_1 \vee \cdots \vee c_n))$$

- b_i und h_j sind Atome ohne Constraintsymbole,
  c_k sind Constraintsatome.
- l, m, n können 0 sein.
- Jede freie Variable der Konsequenz kommt in der Bedingung der Implikation vor.

Keine Anmerkung

Vorteile

- Skolemisierung nicht mehr nötig
- Behandlung von realistischen Modellgenerierungsproblemen

Anwendungen

- Datenbanken: Konsistenzüberprüfung von Integritätsbedingungen (referential integrity constraints)
- Disjunktive Constraint-Logikprogrammierung mit Vorwärtsschließen

Keine Anmerkung

Münchener Mietspiegel

Berechnung der ortsüblichen Vergleichsmiete einer Wohnung. Detailierte Fragen zu Größe, Alter, Lage und Ausstattungsmerkmalen.

Bisher:

Von Hand. Ein Wochenende bzw. halbe Stunde mit Experten, falls alle Fragen beantwortet.

Implementierungs-Rezept:

Tabellen, Regeln und Formeln in einer implementieren, als ob alle Fragen beantwortet.

Dann Formeln als Constraints betrachten und passenden Constraintlöser mit CHR hinzufügen.
Programm bleibt praktisch unverändert.

Anmerkung

260 Städte.

Broschüre, ein Qudratmeter Mieterverein Zettel.

Höhe der Kachelung im Bad über 180cm. (Grünfläche unwichtig).

Benutzer kann oder will nicht alles über sich bekanntgeben. (Internet-Sicherheit).

Unschärfe durch statistischen Ansatz selten berücksichtigt.

Rechnen mit Bandbreiten.

Constraint-Logikprogrammiersprache mit CHR. Zeitaufwand wenige Wochen. Wenige Seiten Code. Inkl. Internetanbindung durch speziellen Web-Server.

Münchener Mietspiegel Online

Entscheidungsunterstützungssystem im WWW.
Über 10000 Anfragen seit Anfang 1996.

- Jederorts und jederzeit zugänglich.
- Statt ein Wochenende wenige Minuten.
- Korrekter Umgang mit unvollständigen Angaben und statistische Unschärfe.

→ Mietspiegel nun auch für Wohnungssuche.

- Wartung und Anpassung an neue Mietspiegel.
- Software-Wiederverwendung durch Klonen.

Anmerkung

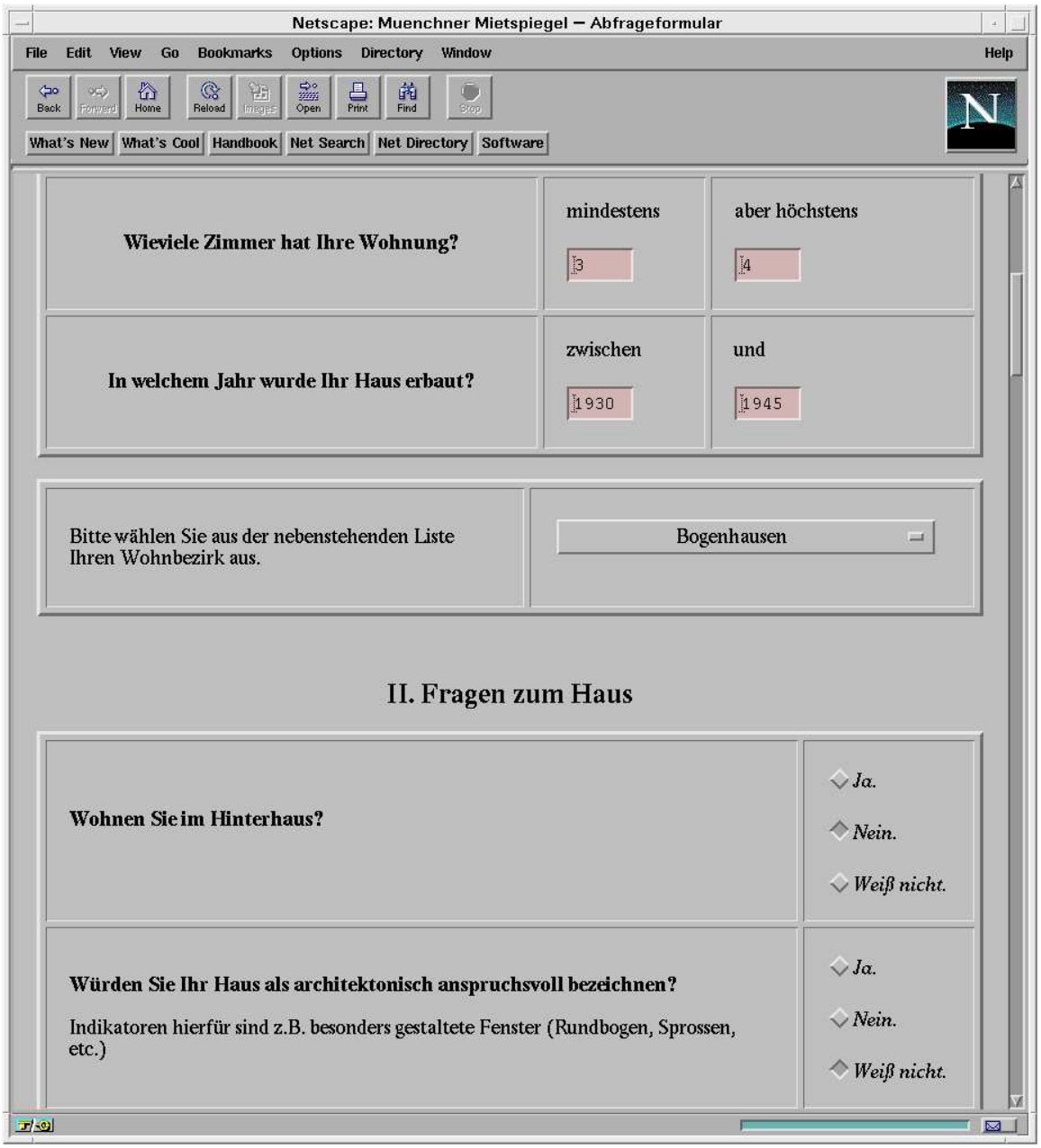
Mit Constraint-Programmierung und CHR:
Aufwand wenige Wochen. Wenige Seiten Code.
Inkl. Internetanbindung durch Web-Server.

Berechnet ortsübliche Vergleichsmiete.

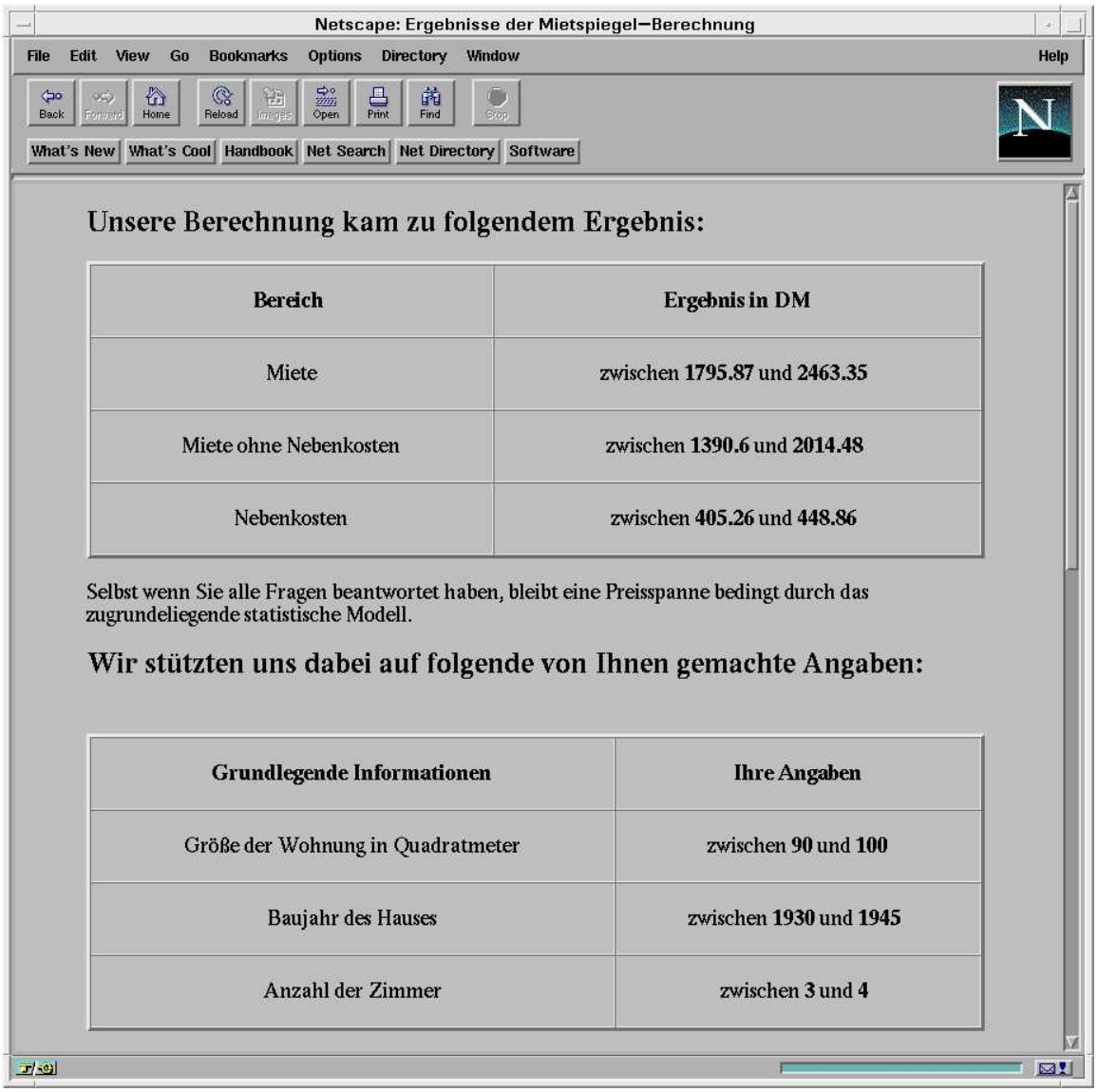
Systems 96+97.

McKinsey.

CP'97 Konferenz in Linz, Workshop Internet und Constraints.


Netscape: Muenchner Mietspiegel – Abfrageformular
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
What's New What's Cool Handbook Net Search Net Directory Software
Wieviele Zimmer hat Ihre Wohnung?
mindestens
3
aber höchstens
4
In welchem Jahr wurde Ihr Haus erbaut?
zwischen
1930
und
1945
Bitte wählen Sie aus der nebenstehenden Liste Ihren Wohnbezirk aus.
Bogenhausen
II. Fragen zum Haus
Wohnen Sie im Hinterhaus?
Ja.
Nein.
Weiß nicht.
Würden Sie Ihr Haus als architektonisch anspruchsvoll bezeichnen?
Indikatoren hierfür sind z.B. besonders gestaltete Fenster (Rundbogen, Sprossen, etc.)
Ja.
Nein.
Weiß nicht.

Anmerkung


Netscape: Ergebnisse der Mietspiegel–Berechnung
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
What's New What's Cool Handbook Net Search Net Directory Software
Unsere Berechnung kam zu folgendem Ergebnis:
Bereich
Ergebnis in DM
Miete
zwischen 1795.87 und 2463.35
Miete ohne Nebenkosten
zwischen 1390.6 und 2014.48
Nebenkosten
zwischen 405.26 und 448.86
Selbst wenn Sie alle Fragen beantwortet haben, bleibt eine Preisspanne bedingt durch das zugrundeliegende statistische Modell.
Wir stützten uns dabei auf folgende von Ihnen gemachte Angaben:
Grundlegende Informationen
Ihre Angaben
Größe der Wohnung in Quadratmeter
zwischen 90 und 100
Baujahr des Hauses
zwischen 1930 und 1945
Anzahl der Zimmer
zwischen 3 und 4

Anmerkung

Klonen

Jeder Teil des Formulars kann in einer anderen Webseite wiederverwendet werden, einfach durch kopieren:

• Formular beliebig umgestalten (inkl. Eingabefelder)

• Formular beliebig kürzen (außer Form-Deklaration)

• Formular beliebig vergrössern (neue Eingabefelder werden ignoriert)

• Eingabefelder vorbelegen und verstecken

Anmerkung

Software-Wiederverwendung.

Mietpreisniveau-Bestimmung: Keine Detailfragen, Bad ja, aber Kachelungshöhe egal.

Zusammenfassung

Constraint-Programmierung
Innovativer Ansatz zum Modellieren und Lösen schwieriger Probleme in Entscheidungsunterstützungssystemen.

Constraint Handling Rules
Sprache zum Schreiben von Constraintlösern.
Optimale Anpassung an Anwendungen.

Flexible und wartbare intelligente Software.
Schnelle Anpassung an veränderte Problemstellungen.

Anmerkung

ENDE