Einführung in die KI

Prof. Dr. sc. Hans-Dieter Burkhard
Vorlesung Winter-Semester 2005/06

2. Constraints:

Grundbegriffe
Constraint Propagierung
Anwendungsbeispiele
(Bildinterpretation, zeitliches Schließen)

Beschränkungen (Constraints)

Allgemeine Problemstellung:

Einschränkende Bedingungen („Constraints“)
für variable Parameter sollen gleichzeitig erfüllt werden

- Produktionsplanung mit
  - Anforderungen an Produkt
  - Anforderungen an Verfahren
  - Anforderungen an Kosten, Zeit, ...
  - Abhängigkeiten zwischen Parametern
- Stundenplanung
- Scheduling
- Landkarten einfärben, ...

Lösung durch Suche (Probieren)

Mögliches Verfahren: Zustandsraum-Suche

Sukzessive die Variablen mit zulässigen Werten belegen,
Backtracking bei Verletzung von Einschränkungen

Zustände: Teil-Belegungen der Parameter mit Werten
Zustandsübergang: Festlegung eines Parameterwertes
(soweit ohne Verletzung von Constraints möglich)

Problem der Suchverfahren: Kombinatorische Explosion
Verbesserungsmöglichkeiten:
Zwischenergebnisse bzgl. Beschränkungen testen

Beispiel: Ziege-Wolf-Kohlkopf

Lokale heuristische Zustandsraum-Suche

Zustände:	Belegungen der Parameter mit Werten
Zustandsübergang:	Änderung eines Parameterwertes
Heuristik:	Parameterwert so ändern, dass Konflikte minimiert werden
Konflikt:	Anzahl verletzter Constraints

Gute Resultate selbst für schwere Probleme,
wenn Lösungen überall im Lösungsraum „dicht“ verteilt sind.
Andernfalls z.B. Evolutionäre Algorithmen anwenden.

Beispiel: Queens-Problem

Propagierung von Beschränkungen

Beschränkungen nutzen zur Reduktion des Suchraums:

- geschicktes Kombinieren von Bedingungen

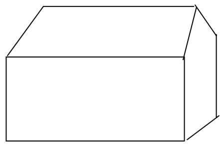
Weiterreichen (Propagieren) lokaler Einschränkungen („Constraint-Propagation“)

Willkürliches Einschränken des Suchraumes (mit Möglickeit zum Backtracking)

```
    S E N D
+   M O R E
= M O N E Y
```


Beispiel aus Bereich ,,Bildverstehen"

Zweidimensionales Abbild (Linien-Zeichnung) eines 3-dimensionalen Körpers interpretieren



Bildinterpretation (Beispiel)

Reale Szene: 3D

Elektronisches Abbild: 2D (Pixel-Matrix)

Filter (Vorverarbeitung)

Segmentierung

Identifikation von Linien

Interpretation von Linien (Beschriftung)

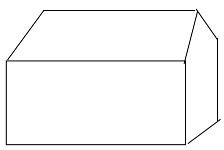
Identifikation von Objekten

Beziehungen zwischen Objekten

Szenen-Interpretation

(Re-)konstruktion von Informationen

Zweidimensionales Abbild eines 3-dimensionalen Körpers unter speziellen Voraussetzungen interpretieren



Keine Schatten oder Bruchlinien.

Verdeckte Kanten sind nicht sichtbar.

Alle Eckpunkte sind Schnittpunkte genau dreier aufeinandertreffender Flächen.

(Die Spitze der Cheops-Pyramide ist nicht erlaubt. -- Zur Vereinfachung, ansonsten komplexere Verfahren.)

,,Allgemeiner Beobachtungspunkt" wird verlangt: Bei geringen Ortsveränderungen des Beobachters darf kein Schnittpunkt seinen Typ wechseln.

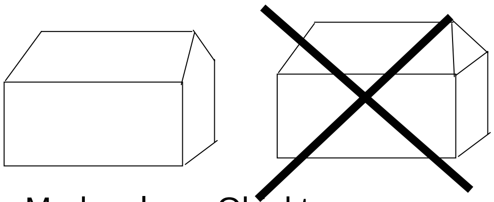
Objekte identifizieren

Erkennung eines Objektes durch Interpretation der Linien

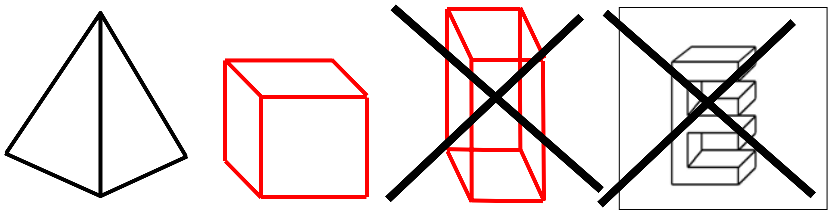
Konvex

Konkav

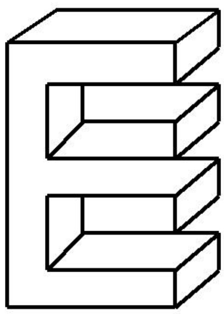
Begrenzung



Zuordnungen: Linien-Merkmale zu Objekten



Interpretationen im Wettstreit



Begrenzungslinien

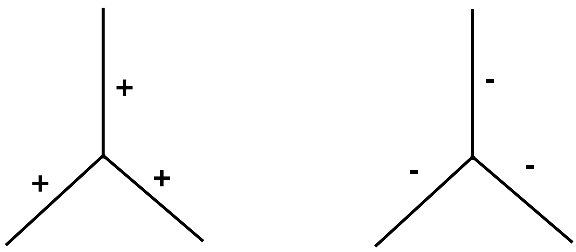
- bilden die äußeren Kanten des Objektes.
  Gekennzeichnet werden sie mit einem Pfeil ,,→". Die Pfeile der Begrenzungslinien sind so gerichtet, daß die Körperfläche immer rechts liegt.

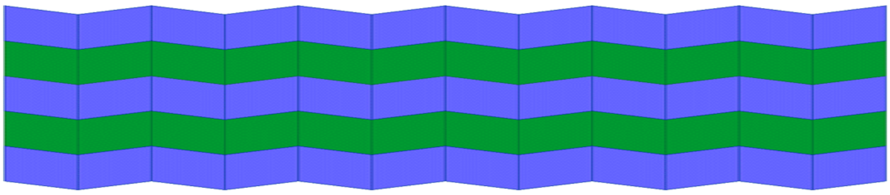
Innenlinien

- sind die Kanten im Inneren des Objektes.
  Es werden zwei Arten unterschieden:
  - **Konvexe Linien:**
    - Beide Grenzflächen sind vom Beobachter abgewandt, wie bei einem Würfel. In der Zeichnung sind sie mit einem ,,+" versehen.
  - **Konkave Linien:**
    - Beide Begrenzungsflächen schließen den Beobachtungsstandpunkt ein. Ein Beispiel wäre der Blick in ein geöffnetes Buch. Sie sind mit einem ,,–" markiert.

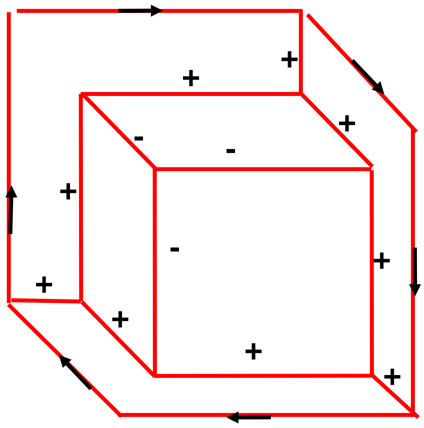
Konkav oder konvex?

Linieninterpretationen sind abhängig vom Kontext



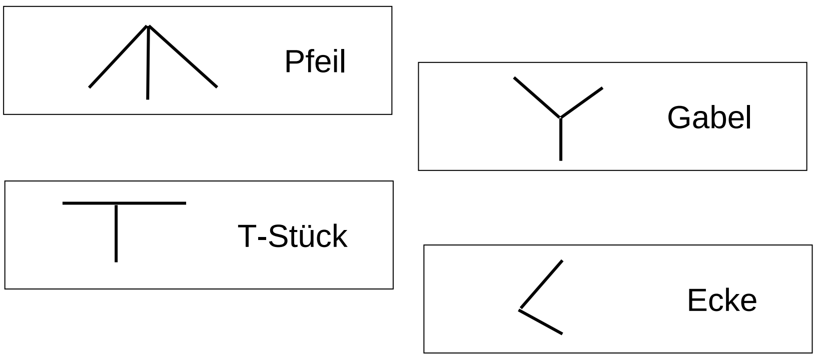


Würfel



4 Typen von Schnittpunkten

(aufgrund der Voraussetzungen)

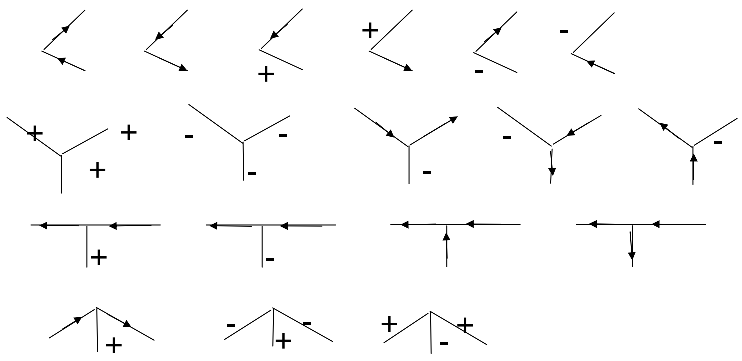


Beschriftete Schnittpunkte

Bei Kanten vier Möglichkeiten ,,←", ,,→", ,,–" und ,,+".

Folglich: insgesamt $4^3+4^3+4^3+4^2=208$ Möglichkeiten.

Davon aber nur 18 physikalisch möglich (Constraints!)



Beschriftungsverfahren

Gegebene 2-D-Zeichnung *konsistent* beschriften, d.h.:

Beschriftung der Ecken (*Variable*) mit Werten aus den zulässigen (18) Typen

Constraints: längs einer Kante darf sich der Kantentyp nicht ändern

oder:

Beschriftung ist eine Belegung der Kanten (*Variable*) mit Werten aus {„←“, „→“, „–“ , „+“}

Constraints: an den Ecken dürfen nur zulässige Typen auftreten

In komplexeren Bildern weitere Constraints durch Licht/Schatten.

Suchverfahren zur Beschriftung

Auswahl einer Ecke, diese beschriften

Fortlaufend Nachbarecken beschriften - solange dies möglich ist. Dabei Auswahl aus mehreren Alternativen:

- ausgewählte Ecke
- ausgewählte Beschriftung der Ecke

3. Beim Auftreten von Widersprüchen: Backtracking zu weiteren Beschriftungsalternativen unter 2.

Zweckmäßig:
Kein simples ,,chronologisches Backtracking" verwenden

Verfahren von Waltz

Stammvater für „Constraint Propagation“

Initial:

Die Ecken mit allen möglichen Beschriftungen versehen

Zyklus:

Solange noch Änderungen möglich sind:

Paare von Nachbarecken vergleichen
(Constraints: Konsistenz entlang der Linien)
Inkonsistente Beschriftungstypen an Ecken entfernen.

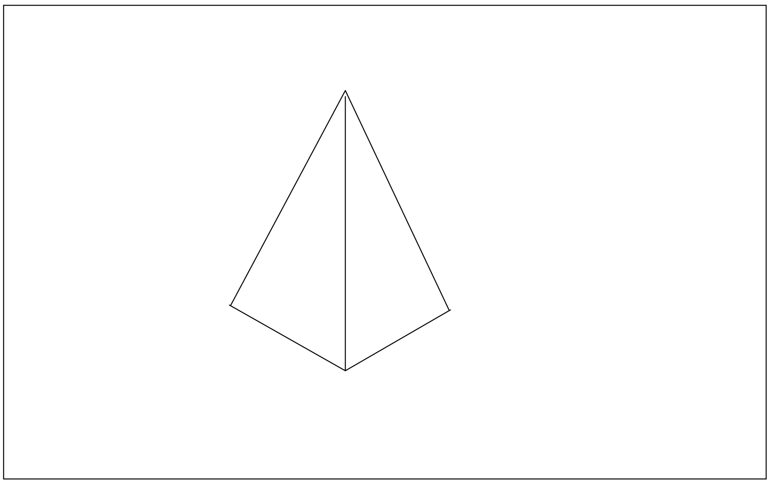
Abschluss:

Beschriftungen festlegen

(z.B. Suche über den verbliebenen Beschriftungen).

Durch Verringerung der Beschriftungen entstehen neue Inkonsistenzen, die wiederum zum Entfernen von Beschriftungen führen: „Constraint Propagation“

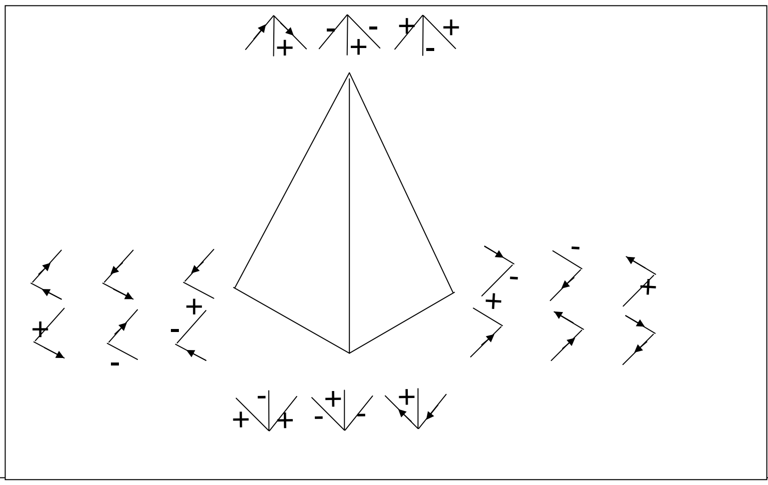
Verfahren von Waltz



H.D.Burkhard, HU Berlin
Winter-Semester 2005/06

Vorlesung Einführung in die KI
Constraints

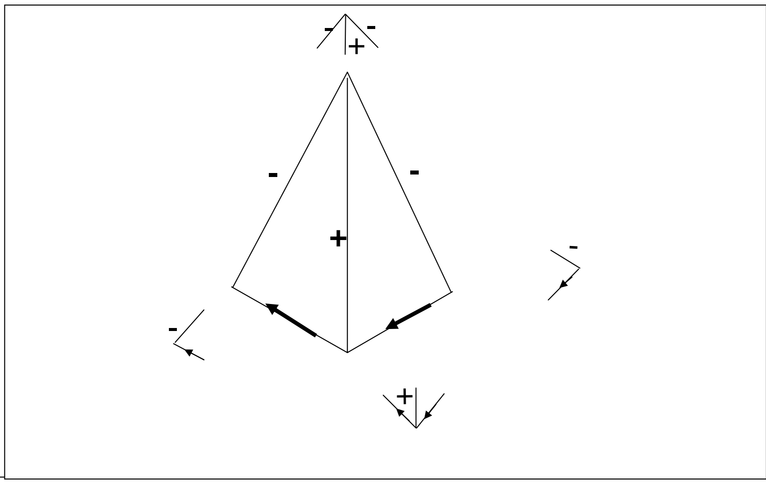
Verfahren von Waltz



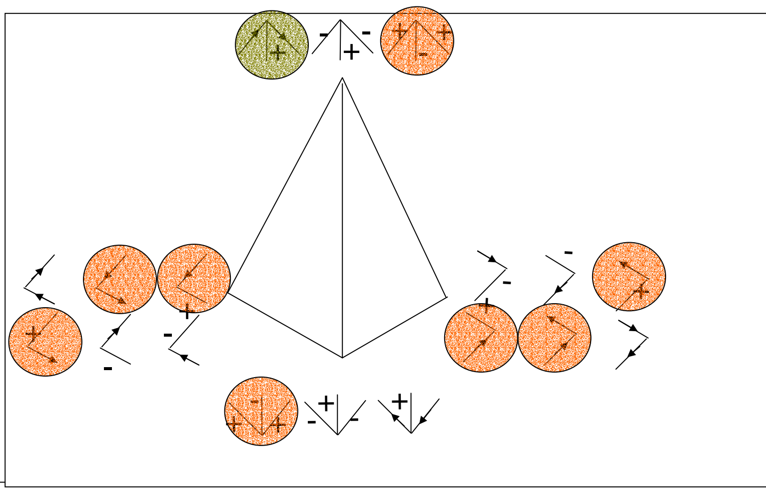
H.D.Burkhard, HU Berlin
Winter-Semester 2005/06

Vorlesung Einführung in die KI
Constraints

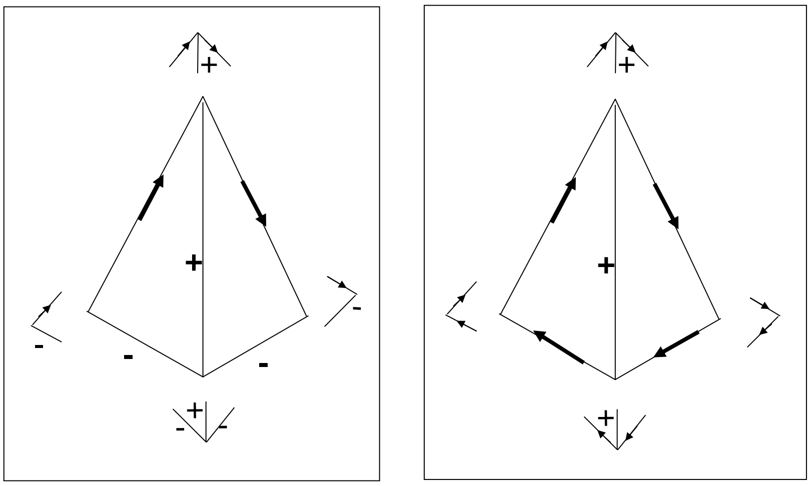
Resultat



Backtracking: Streichen anderer Werte



2 weitere Resultate



Beschriftungsverfahren

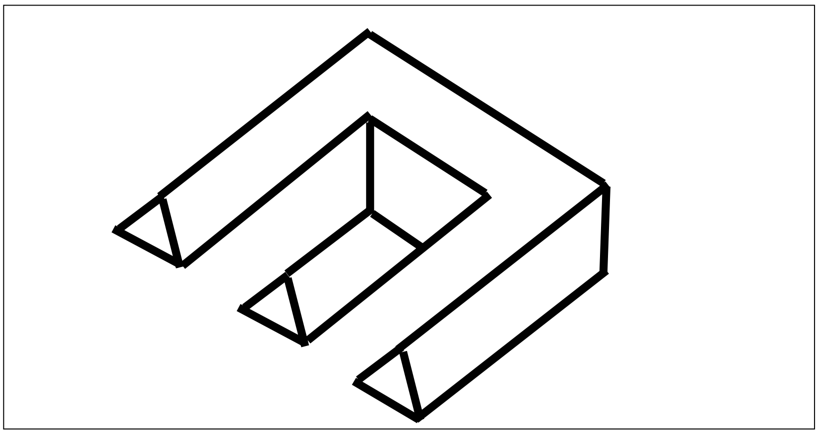
Für realistische Bilder existiert stets (mindestens) eine konsistente Beschriftung.

Es gibt manchmal auch konsistente Beschriftungen bei unrealistischen Bildern.

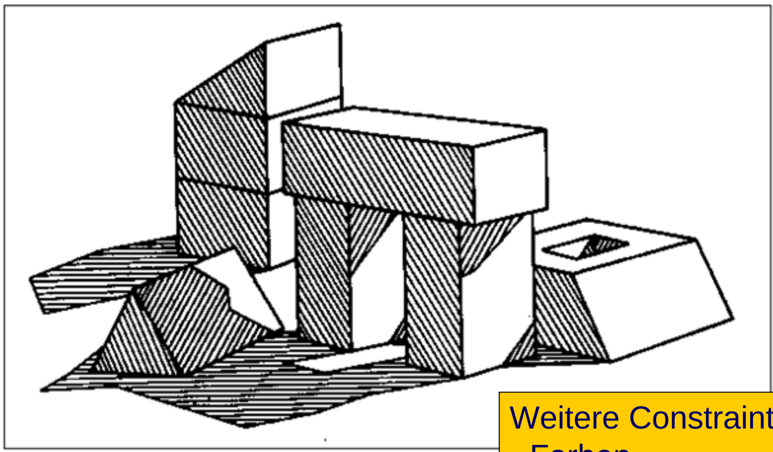
Grund: Die Konsistenz wird nur lokal verlangt

Beschriftungsverfahren

Gibt es konsistente Beschriftung?



Weitere Constraints: Schatten, ...



Weitere Constraints:
- Farben
- Texture (Oberfläche)
- Wissen über Szene

Constraint Propagation

Analog in anderen Problemen anwendbar

Schritte:

A) Wertebereiche einschränken gemäß Constraints

B) Wertebereiche willkürlich einschränken (mit Backtracking)

C) Suche über verbliebenen Werten

A und B können mehrfach und abwechselnd angewendet werden

oder in einem gemeinsamen Schritt:

Größere Einschränkung des Wertebereichs als durch Constraint verlangt

Definitionen: Constraints

Gegeben:

Variablenmenge $V = \{ v_1, ..., v_n \}$ (Parameter)

mit Wertebereichen $Dom(v_i)$, $i=1,...,n$.

Gesamtbereich: $Dom(V) := Dom(v_1) \times ... \times Dom(v_n)$.

__Belegung__ β ordnet jedem v_i einen Wert aus $Dom(v_i)$ zu:

$$\beta = [\ \beta(v_1), ..., \beta(v_n)\ ] \in Dom(v_1) \times ... \times Dom(v_n)$$

Definitionen: Constraints

Constraint C definiert die „zugelassenen Belegungen" über einer Variablen-Teilmenge $V_C = \{v^C_1, ..., v^C_m\} \subseteq V$:

$$C \subseteq Dom(v^C_1) \times ... \times Dom(v^C_m)$$

Constraint-Netz $\boldsymbol{C}$ über V ist eine Menge $\boldsymbol{C} = \{C_1, ..., C_k\}$, wobei jedes C_j ein Constraint über einer Menge $V_{Cj} \subseteq V$ ist.

Definitionen: Constraints

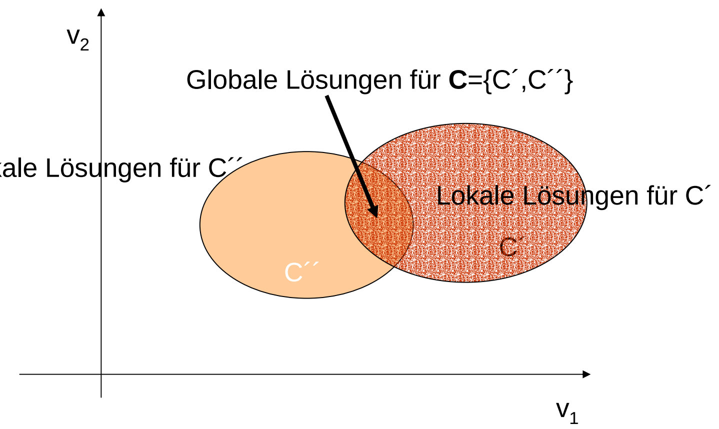
Belegung $\beta = [\ \beta(v_1), ..., \beta(v_n)\ ]$
erfüllt Constraint C über Menge $V_C = \{v^C_1, ..., v^C_m\} \subseteq V$,
falls $[\ \beta(v^C_1), ..., \beta(v^C_m)\ ] \in C$.

β heißt *lokale konsistente Belegung* für C oder *lokale Lösung* für C.

Belegung β *erfüllt das Constraint-Netz* $\boldsymbol{C} = \{C_1, ..., C_k\}$,
falls β alle $C_j \in \boldsymbol{C}$ erfüllt.

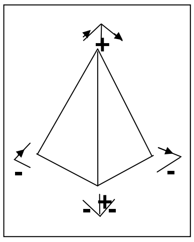
β heißt *global konsistente Belegung* für $\boldsymbol{C}$ oder *globale Lösung* für $\boldsymbol{C}$.

Definitionen: Constraints



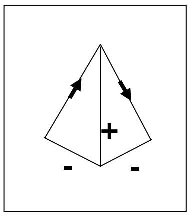
Bildinterpretation

Alternativen:
Variable sind Ecken
Wertebereiche die Ecktypen
Constraints: Konsistenz entlang Linien
Lösung: Beschriftete Ecken



oder

- Variable sind Linien
- Wertebereiche die Linientypen
- Constraints: Konsistenz gemäß Ecktypen
- Lösung: Beschriftete Kanten



Constraint-Erfüllung (Constraint Satisfaction)

Gegeben: Constraint-Netz $C = \{C_1, ..., C_k\}$

Gesucht: globale Lösung β

– Aufgabenarten:
  - existiert Lösung ?
  - finde Lösung β ?

CSP = Constraint Satisfaction Problem

Beispiele

Finde zwei natürliche Zahlen x und y,
so daß $x+y=7$ unter der Vorausetzung $x > y > 2$.

Als Constraint-Problem: $V=\{x,y\}$ $Dom(x)=Dom(y)=\mathbb{N}$

$C_1 \subseteq Dom(y)$ $C_1=\{y \in \mathbb{N}: y > 2\}$
$C_2 \subseteq Dom(x) \times Dom(y)$ $C_2=\{[x,y] \in \mathbb{N}\times\mathbb{N}: x > y \}$
$C_3 \subseteq Dom(x) \times Dom(y)$ $C_3=\{ [x,y] \in \mathbb{N}\times\mathbb{N}: x+y=7 \}$

$\boldsymbol{C}=\{C_1,C_2,C_3\}$

Eine *lokale Lösung* bezüglich C_1 ist $x=1$ und $y=3$.

Für eine *globale Lösung* müssen alle drei Bedingungen erfüllt werden. Eine passende Belegung β ist $x=4$ und $y=3$.

Beispiele

$V=\{x,y,z\}$, $Dom(x)=[0,1]$, $Dom(y)=[0,1]$, $Dom(z)=[0,1]$

$C_1=\{[x,y] : x > y\}$ für $V_1 = \{x,y\}$
$C_2=\{[y] : y > 0.5\}$ für $V_2 = \{y\}$
$C_3=\{[x,z] : x+z=1\}$ für $V_3 = \{x,z\}$
$C_4=\{[x,z] : x < z\}$ für $V_4 = \{x,z\}$

Eine *lokale Lösung* bezüglich C_3 ist $[0.5, 0.7, 0.5]$.

Eine *globale Lösung* für $\boldsymbol{C}=\{C_1,C_2,C_3,C_4\}$ existiert nicht:

Die gegebenen Voraussetzungen sind inkonsistent.

Aus C_1,C_2,C_4 folgt $z > x > y > 0.5$. C_3 steht dazu im Widerspr.

Für $\boldsymbol{C}=\{C_1,C_2,C_3\}$ ist $[0.7, 0.6, 0.3]$ *globale Lösung*.

Constraint-Optimierung (Constraint Optimization)

Gegeben: Constraint-Netz $C = \{C_1, ..., C_k\}$
Kostenfunktion $c: Dom(V) \rightarrow R$
Gesucht: optimale globale Lösung β^*

Aufgabenarten:

$c(\beta^*) \geq r$? für gegebenes $r \in R$
$\rightarrow$ modellierbar als zusätzliches Constraint,
Behandlung als Constraint-Erfüllungsproblem

finde $c(\beta^*)$.
$\rightarrow$ Behandlung als „ $c(\beta^*) \geq r$?“ mit unterschiedlichen r
finde β^* .

Lösungsverfahren für Constraints

Satz:
Es gibt kein universelles Lösungsverfahren für beliebige Constraint-Probleme

Beispiel:

Diophantische Gleichungen

Endliche Constraint-Probleme (Dom(V) endlich)

sind oft NP-vollständig

Constraint-Propagierung

Idee: *Geeignete* Einschränkungen D_i der Wertebereiche $Dom(v_i)$ finden („zusätzliche Constraints")

Geeignet: Es gehen keine Lösungen verloren.

Propagierung: Fortlaufende Einschränkungen durch abwechselnde Betrachtung unterschiedlicher Constraints führen für Wertebereiche $Dom(v_i)$ ($i=1,..,n$) zu Folgen

$$D_i^l \subseteq \ldots \subseteq D_i^5 \subseteq D_i^4 \subseteq D_i^3 \subseteq D_i^2 \subseteq D_i^1 \subseteq Dom(v_i)$$

Suche in $D^l := D_1^l \times \ldots \times D_n^l$ nach Lösung

Probleme z.B.:

- evtl. unendliche Folge der D^l
- evtl. D^l noch zu umfangreich für Suche

Constraint-Propagierung: Einschränkungen

Gegeben: Constraint-Netz **C**=$\{C_1,...,C_k\}$ über V=$\{v_1,...,v_n\}$

„Einschränkung" (der Wertebereiche):

D = $D_1 \times ... \times D_n$ für Teilmengen $D_i \subseteq Dom(v_i)$, i=1, ...,n

z.B. als Intervall im R_n

D heißt *lokal konsistente Einschränkung* bzgl. C_j, falls gilt:

$\forall i \in \{1,...,n\}\ \forall d_i \in D_i\ \exists \beta=[a_1,...,a_{i-1},d_i,a_{i+1},...,a_n] \in D$:

β lokale Lsg. für C_j

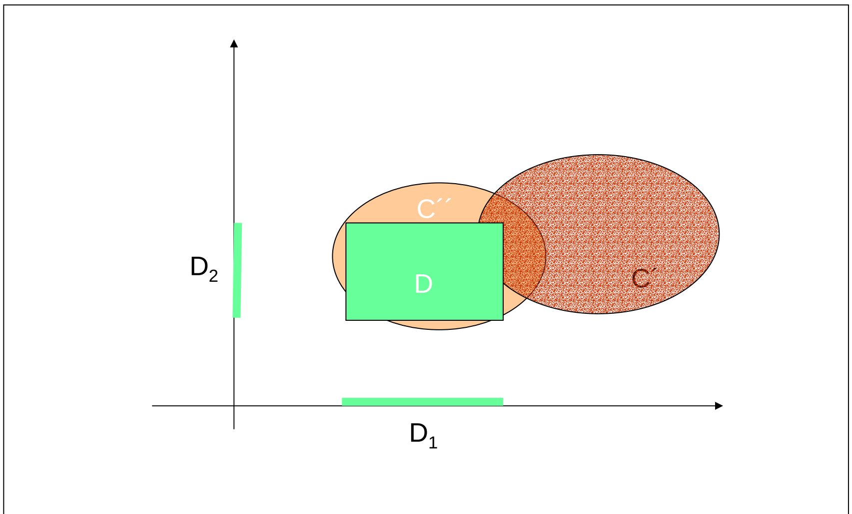
D heißt *global konsistente Einschränkung*, falls gilt:

$\forall i \in \{1,...,n\}\ \forall d_i \in D_i\ \exists\ \beta=[a_1,...,a_{i-1},d_i,a_{i+1},...,a_n] \in D$:

β globale Lsg. für **C**

D heißt *inkonsistent*, wenn D keine globale Lösung enthält.

Lokal konsistente Einschränkung



Constraint-Propagierung: Einschränkungen

Ebenfalls Einschränkung

Innerhalb einer gegebenen Menge $D = D_1 \times \ldots \times D_n$

gibt es genau eine *maximale global konsistente Einschränkung*

$D^{max/global} = D_1^{max/global} \times \ldots \times D_n^{max/global}$ mit

$D_i^{max/global} := \{ d_i \in D_i \mid \exists\, \beta = [a_1,\ldots,a_{i-1},d_i,a_{i+1},\ldots,a_n] \in D : \beta$ globale Lsg.$\}$

Innerhalb einer gegebenen Menge $D = D_1 \times \ldots \times D_n$

gibt es genau eine *maximale bzgl. C_j lokal konsist. Einschränk.*

$D^{max/Cj\text{-}lokal} = D_1^{max/Cj\text{-}lokal} \times \ldots \times D_n^{max/Cj\text{-}lokal}$ mit

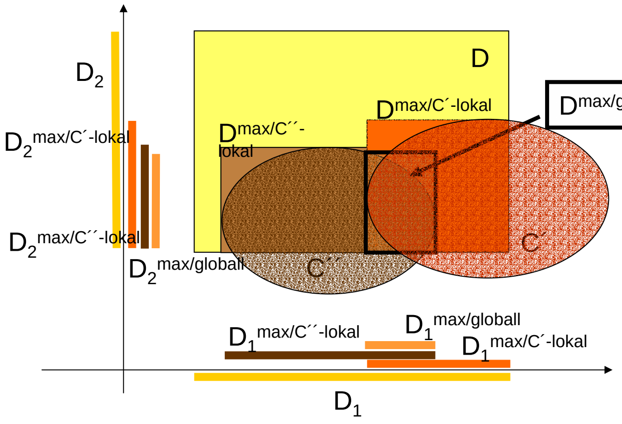
$D_i^{max/Cj\text{-}lokal} := \{ d_i \in D_i \mid \exists\, \beta = [a_1,\ldots,a_{i-1},d_i,a_{i+1},\ldots,a_n] \in D : \beta$ lok. Lsg. für $C_j \}$

Constraint-Propagierung: Einschränkungen

Die Mengen $D_i^{max/global}$ (bzw. $D_i^{max/Cj\text{-}lokal}$) sind die Projektionen der in D enthaltenen globalen (bzw. lokalen) Lösungen auf die Wertebereiche $Dom(v_i)$.

Es gilt für beliebiges $j=1,\ldots,k$: $D_i^{max/global} \subseteq D_i^{max/Cj\text{-}lokal}$.

Constraint-Propagierung: Einschränkungen



Constraint-Propagierung: Einschränkungen

Globale Lösung existiert in D

gdw. nicht-leere global konsistente Einschränkungen von D existieren

gdw. $D^{max/global} \neq \emptyset$

Es gibt keine globale Lösung in D gdw. $D^{max/global} = \emptyset$

Lokale Lösung für C_j existiert in D

gdw. nicht-leere für C_j lokal konsistente Einschränkungen von D existieren

gdw. $D^{max/Cj\text{-}lokal} \neq \emptyset$

Es gibt keine lokale Lösung für C_j in D gdw. $D^{max/Cj\text{-}lokal} = \emptyset$

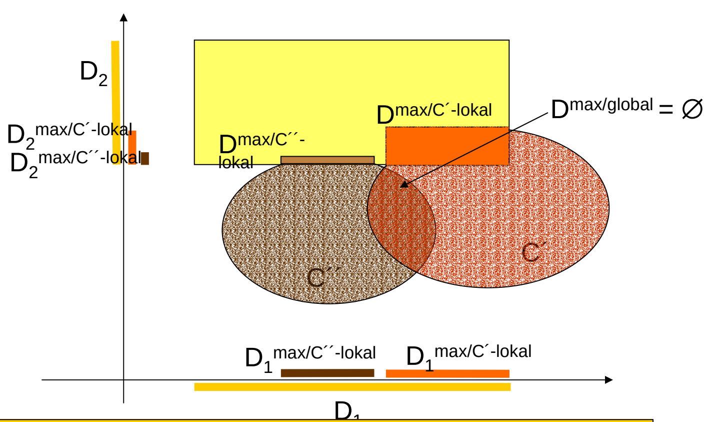
Constraint-Propagierung: Einschränkungen

Zusammenhang
globale Lösung/lokal konsistente Einschränkungen ?

Falls $D^{max/Cj\text{-}lokal} = \emptyset$ für ein Constraint C_j ,
(falls $D_i^{max/Cj\text{-}lokal} = \emptyset$ für ein Constraint C_j und eine Variable v_i)
so gilt $D^{max/global} = \emptyset$ und es gibt keine globale Lösung in D

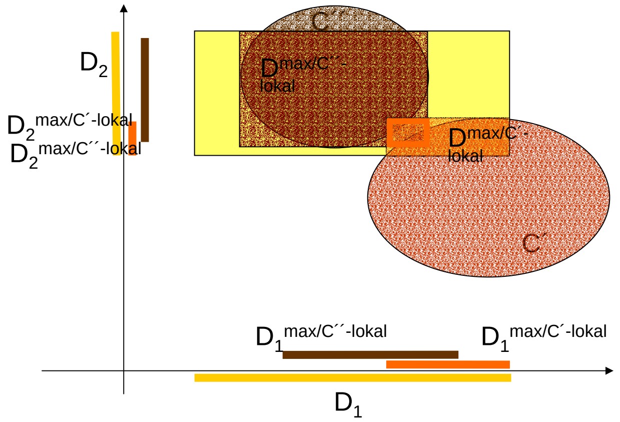
Umkehrung gilt nicht:
Auch wenn für alle Constraints C_j gilt: $D^{max/Cj\text{-}lokal} \neq \emptyset$,
kann $D^{max/global} = \emptyset$ gelten und es gibt keine globale Lösung in D

Constraint-Propagierung: Einschränkungen



Frage: Wäre $\bigcap D^{max/Cj\text{-}lokal} \neq \emptyset$
hinreichend für $D^{max/global} \neq \emptyset$
und die Existenz globaler Lösungen in D ?

Constraint-Propagierung: Einschränkungen



Frage: Wäre $\bigcap D^{max/Cj\text{-}lokal} \neq \emptyset$

hinreichend für $D^{max/global} \neq \emptyset$

und die Existenz globaler Lösungen in D ?

47

Constraint-Propagierung: Einschränkungen

Als *maximale global konsistente Einschränkung*

des Constraint-Netzes $C = \{C_1, ..., C_k\}$

wird die maximale global konsistente Einschränkung

$E=E_1\times...\times E_n$ von $Dom(V)=Dom(v_1)\times...\times Dom(v_n)$ bezeichnet.

Dabei gilt:

$E_i =\{ d_i \in Dom(v_i) \mid \exists\beta=[a_1,..,a_{i-1},d_i,a_{i+1},..,a_n] : \beta$ globale Lsg. von C$\}$

E enthält alle globalen Lösungen, insbesondere gilt:

Globale Lösungen existieren gdw. $E \neq \emptyset$.

Für jeden Constraint C_j gilt: :

Falls $E \subseteq D$ für eine Einschränkung D, so $E_i \subseteq D_i^{max/Cj\text{-}lokal}$.

Schema: Constraint-Propagierung

Propagierung: Fortlaufende Einschränkungen durch abwechselnde Betrachtung unterschiedlicher Constraints führen für Wertebereiche $Dom(v_i)$ $(i=1,..,n)$ zu Folgen

$$D_i^l \subseteq \ldots \subseteq D_i^5 \subseteq D_i^4 \subseteq D_i^3 \subseteq D_i^2 \subseteq D_i^1 \subseteq Dom(v_i)$$

Suche in $D^l := D_1^l \times \ldots \times D_n^l$ nach Lösung

Schema: Constraint-Propagierung

1. Wähle eine Menge $D(0) \subseteq Dom(V)$ als initiale Einschränkung, $s := 1$.

2. Wähle ein (aussichtsreiches) $C(s) \in \mathbf{C}$.

Wahl-Möglichkeit

Wahl-Möglichkeit

3. Wähle eine neue Einschränkung $D(s) \subseteq D(s-1)$, so dass $D(s)$ eine lokal konsistente Einschränkung bezüglich $C(s)$ ist.

Nicht notwendig maximal – Möglichkeiten zum Backtracking

Schema: Constraint-Propagierung

4. Zwischenauswertung. Es können folgende Fälle auftreten:
   (a) Globale Lösung wird in D(s) gefunden (z.B. durch Suche).
   (b) Es ist keine weitere Einschränkung zu erwarten, da kein weiteres C(s) $\in$ **C** gefunden werden konnte, so dass D(s) ≠ D(s-1).
   (c) D(s) ist global inkonsistent (gemäß Suche, insbesondere bei D(s) = ∅).

Daraus ergeben sich folgende Möglichkeiten zur Weiterarbeit:

- o Weiter bei 2. mit s:=s+1 , falls keiner der drei Fälle zutrifft.
- o Abbruch bei 4a)
- o Backtracking zu 2. oder 3. eines Schrittes s'<s für 4b) oder 4c): Dort alternative Einschränkung wählen. Abbruch, wenn Backtracking nicht aussichtsreich.

nicht notwendig chronologisch

kleinerer Bereich oder: anderer Bereich

Schema: Constraint-Propagierung

Es ergibt sich Folge von Einschränkungen

$$\mathrm{Dom}(V) \supseteq D(0) \supseteq D(1) \supseteq D(2) \supseteq D(3) \ldots \supseteq D(s) \ldots$$

Dabei D(s) lokal konsistente Einschränkung von D(s-1) bzgl. C(s) ,
i.a. aber nicht bzgl. C(s-1), C(s-2), C(s-3), ...

→ Wiederholte (evtl. sogar unbegrenzte) Verwendung der C_j aus C kann möglich sein .

Schema: Constraint-Propagierung

Kein eigentlicher Algorithmus (prinzipielle Unlösbarkeit).

Suchverfahren in der gefundenen Einschränkung (4) .

Wahl-Möglichkeit in (3):

Kleine (nicht maximal konsistente) D(s):

- Suchverfahren einfacher.
- evtl. wird Lsg. verfehlt:

  (nicht-chronologisches) Backtracking notwendig.

Schema: Constraint-Propagierung

Satz:

Voraussetzungen:

- D(0) = Dom(V)
- D(s) jeweils maximale lokale Einschränkung von D(s-1)

Behauptungen:

- D(s) enthält stets die

  maximale global konsistente Einschränkung E von **C** :

  $E \subseteq D(s)$
- Falls ein $D(s)= \emptyset$, so existiert keine Lösung.

Visualisierung: Constraint-Graphen

Constraint C_j über $V_j \subseteq V$ heißt *binär*, falls $card(V_j) = 2$.

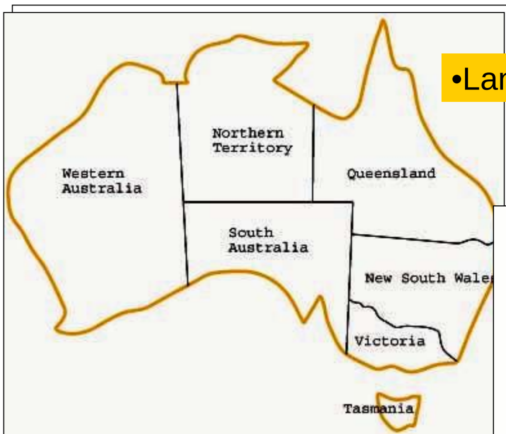
Constraint-Netz $\mathcal{C} = \{C_1, ..., C_k\}$ aus binären Constraints C_j ist darstellbar als Graph $G=[V, \mathcal{C}]$ mit

Knoten: Variable v_i (mit Wertebereichen $Dom(v_i)$)

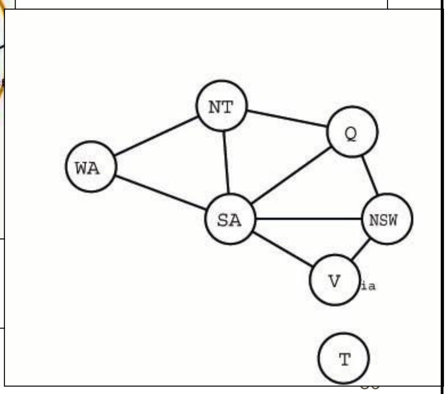
Kanten: Binäre Constraints C_j

Beispiele:
- Landkarten-Färbung
- Eckenbeschriftungen bei Bildinterpretation

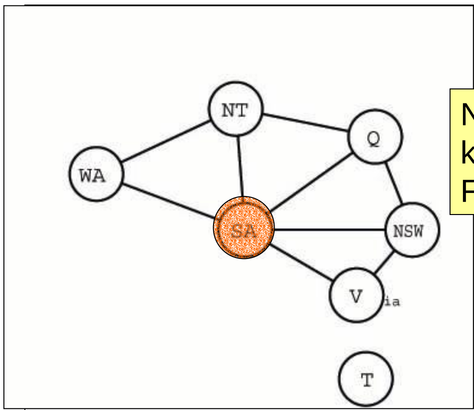
Visualisierung: Constraint-Graphen



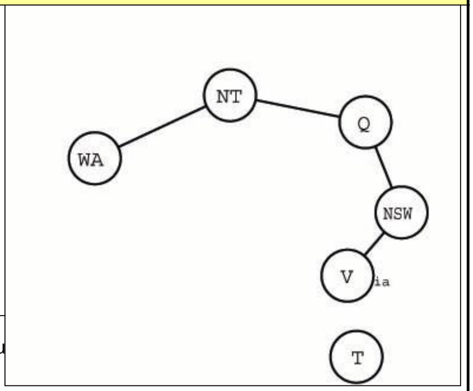
•Landkarten-Färbung



Vereinfachende Zerlegungen des Constraint-Graphen



Nach Einfärbung von SA können andere mit verbleibenden Farben versehen werden



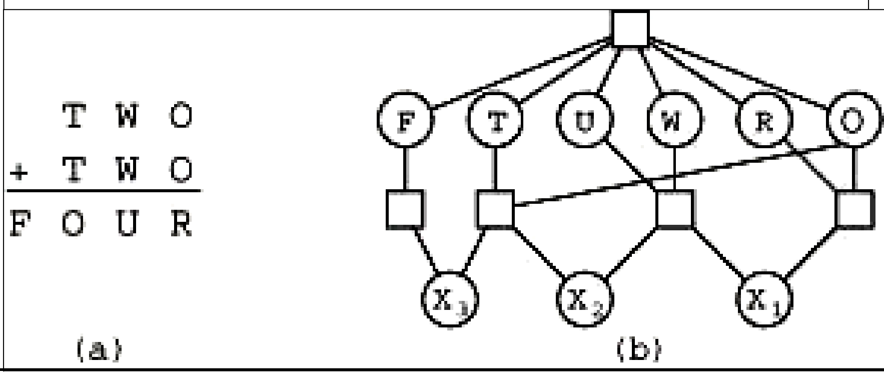
Visualisierung: Constraint-Hyper Graphen

Knoten: Variable v_i

Hyper-Knoten: Constraints C_j über $V_j \subseteq V$

Kanten: verbinden Hyperknoten für Constraints C_j

mit Knoten für Variable v_i V_j



Visualisierung: Constraint-Graphen

Jedes Constraint-Netz in binäres Constraint-Netz überführbar.

Unäre Constraints ($card(V_j) = 1$) in $Dom(v_i)$ integrieren.

Viele Verfahren sind auf binäre Constraint-Netze beschränkt.

z.B. Vereinfachende Zerlegungen des Constraint-Graphen.

Übergang zu Binärem Constraint-Netz

Gegeben Constraint-Netz $\mathcal{C} = \{C_1, ..., C_k\}$

mit Constraints C_i über $V_{Ci} = \{v^{Ci}_1, ..., v^{Ci}_m\}$

$C_i \subseteq Dom(v^{Ci}_1) \times ... \times Dom(v^{Ci}_m)$

Konstruieren binäres Constraint-Netz

mit Variablen $w_1, ..., w_k$

mit $Dom(w_i) = C_i \subseteq Dom(v^{Ci}_1) \times ... \times Dom(v^{Ci}_m)$

d.h. Werte der neuen Variablen w_i sind die lokal konsistenten Belegungen der Variablen $v^{Ci}_1, ..., v^{Ci}_m$ des Constraints C_i

Übergang zu Binärem Constraint-Netz

Variable $w_1,...,w_k$

mit $Dom(w_i) = C_i \subseteq Dom(v^{Ci}_1)\times...\times Dom(v^{Ci}_m)$

$x\in\{1,2\}$
$y\in\{2,4\}$
$z\in\{4,5\}$

C_1: x+y=z

C_2: x<y

{(1,4,5), (2,2,4)}

u

v

{(1,2), (1,4),(2,4)}

Übergang zu Binärem Constraint-Netz

Belegungen der neuen Variablen müssen Gleichheiten der ursprünglichen Variablen respektieren:

Binäre Constraints: $B^v_{i,j} \subseteq Dom(w_i) \times Dom(w_j)$

definiert für die Variablen $v \in V_{Ci} \cap V_{Cj}$:

$B^v_{i,j} := \{ [b_i, b_j] \mid b_i \in Dom(w_i) \wedge b_j \in Dom(w_j) \wedge \pi_v(b_i) = \pi_v(b_j) \}$

dabei bezeichnet $\pi_v(b_i)$

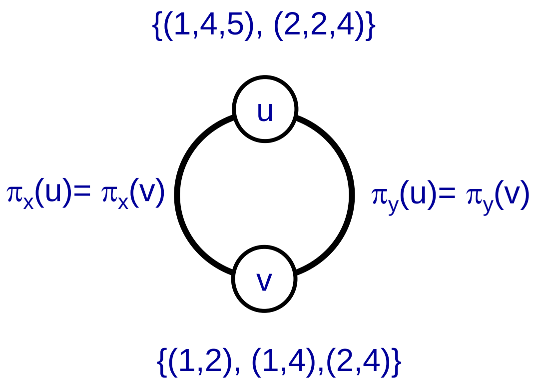
die Projektion von b_i

an der Stelle von v

Übergang zu Binärem Constraint-Netz

Belegungen der neuen Variablen müssen Gleichheiten der ursprünglichen Variablen respektieren:

$x \in \{1,2\}$
$y \in \{2,4\}$
$z \in \{4,5\}$
$x+y=z$
$x<y$



Constraint-Propagation im Allen-Kalkül

Allen-Kalkül: Modell für Zeitliches Schließen

Mögliche Ausgangspunkte:
- Zeitpunkte
- Zeitintervalle
- Ereignisse/Abläufe

Weitere Modelle für zeitliche Abläufe:
- Situationenkalkül
- Algorithmische Logik, z.B. CTL*
- Automaten, Petri-Netze, ...

Beispiel Allen-Kalkül

$$\text{HOLDS}(\text{offside-punishable}(p), \langle max(s_j, s_m), min(e_l, e_m)\rangle) \Leftrightarrow$$
$$\exists j, k, l, m, p_2 :$$
$$\text{OCCUR}(\text{kick}(p_2), j) \wedge \text{HOLDS}(\text{offside-position}(p), k) \wedge$$
$$\text{HOLDS}(\text{ball-free}, l) \wedge \text{HOLDS}(\text{approaching}(p, ball), m) \wedge$$
$$starts(j, l) \wedge in(j, k) \wedge contemporary(l, m) \wedge \text{team}(p) = \text{team}(p_2).$$

starts, in, contemporary bezeichnen Beziehungen zwischen Intervallen

(aus Dissertation Andrea Miene - Bremen, 2003)

Allen-Kalkül

Logisch formaler Zugang für die Repräsentation von zeitlichen Abläufen auf der Basis von Zeit-Intervallen

- Syntax
- Semantik

Ausdrucksmöglichkeiten für

- Intervalle
- Beziehungen zwischen Intervallen (Relationen)
- Stattfinden von Ereignissen (OCCUR)
- Gültigkeit von Fakten (HOLDS)
- ...

Allen-Relationen für Zeitintervalle

Syntax:

– Ausdrucksmöglichkeiten für binäre Relationen

$s(x,y)$, $s_i(x,y)$, $f(x,y)$, $f_i(x,y)$, $d(x,y)$, $d_i(x,y)$,

$b(x,y)$, $b_i(x,y)$, $o(x,y)$, $o_i(x,y)$, $m(x,y)$, $m_i(x,y)$, $e(x,y)$

– Logische Ausdrucksmöglichkeiten (AK, PK1, ...)

– Verbindung zu Intervallen (HOLDS, OCCURS, ...)

Allen-Relationen für Zeitintervalle

Semantik(Modelle): Intervallstrukturen

Eine *Intervallstruktur* ist ein Paar $[I,\{\subseteq, <\}]$, mit

1. I nichtleere Menge (von Intervallen),
2. $\subseteq$ partielle Ordnung (transitiv, reflexiv, antisymmetrisch) über I (Teilmengenrelation)
3. $<$ strikte partielle Ordnung (transitiv, irreflexiv) über I (Präzedenz)

Präzedenz trifft zu, wenn ein Zeitintervall *vollständig vor* einem anderem liegt.
Ein trennendes Zwischenintervall ist nicht erforderlich.

Beabsichtigte Bedeutungen

STARTS(x,y)

$s(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge x < z) \wedge \neg\exists z\, (z \subseteq y \wedge z < x)$

FINISHES(x,y)

$f(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge z < x) \wedge \neg\exists z\, (z \subseteq y \wedge x < z)$

DURING(x,y)

$d(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge x < z) \wedge \quad \exists z\, (z \subseteq y \wedge z < x)$

BEFORE(x,y)

$b(x,y) \leftrightarrow x < y \wedge \exists z\, (x < z \wedge z < y)$

OVERLAPS(x,y)

$o(x,y) \leftrightarrow \exists z\, (z \subseteq x \wedge z < y) \wedge \exists z\, (z \subseteq x \wedge z \subseteq y) \wedge \exists z\, (z \subseteq y \wedge x < z)$

MEETS(x,y)

$m(x,y) \leftrightarrow x < y \wedge \neg\exists z\, (x < z \wedge z < y)$

EQUALS(x,y)

$e(x,y) \leftrightarrow (x \subseteq y \wedge y \subseteq x)$

Beabsichtigte Bedeutungen

STARTS(x,y)

$s(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge x < z) \wedge \neg\exists z\, (z \subseteq y \wedge z < x)$

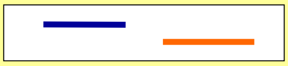
FINISHES(x,y)

$f(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge z < x) \wedge \neg\exists z\, (z \subseteq y \wedge x < z)$

DURING(x,y)

$d(x,y) \leftrightarrow x \subseteq y \wedge \exists z\, (z \subseteq y \wedge x < z) \wedge \quad \exists z\, (z \subseteq y \wedge z < x)$

BEFORE(x,y)

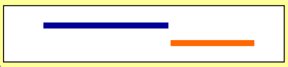


$b(x,y) \leftrightarrow x < y \wedge \exists z\, (x < z \wedge z < y)$

OVERLAPS(x,y)

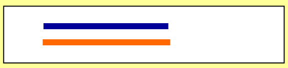
$o(x,y) \leftrightarrow \exists z\, (z \subseteq x \wedge z < y) \wedge \exists z\, (z \subseteq x \wedge z \subseteq y) \wedge \exists z\, (z \subseteq y \wedge x < z)$

MEETS(x,y)



$m(x,y) \leftrightarrow x < y \wedge \neg\exists z\, (x < z \wedge z < y)$

EQUALS(x,y)

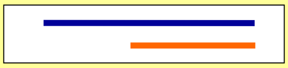


$e(x,y) \leftrightarrow (x \subseteq y \wedge y \subseteq x)$

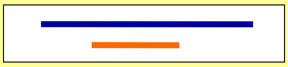
Beabsichtigte Bedeutungen (Inverse)

Inverse von STARTS(x,y)
$s_i(x,y) \leftrightarrow s(y,x)$

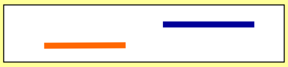
Inverse von FINISHES(x,y)
$f_i(x,y) \leftrightarrow f(y,x)$



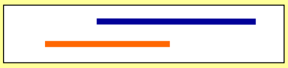
Inverse von DURING(x,y)
$d_i(x,y) \leftrightarrow d(y,x)$



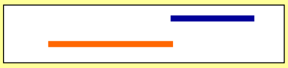
Inverse von BEFORE(x,y)
$b_i(x,y) \leftrightarrow b(y,x)$



Inverse von OVERLAPS(x,y)
$o_i(x,y) \leftrightarrow o(y,x)$



Inverse von MEETS(x,y)
$m_i(x,y) \leftrightarrow m(y,x)$



Axiomatik für Allen-Relationen

- Für jedes Intervall t und jede der 13 Relationen r gibt es ein Intervall t′ mit $r(t,t')$ bzw. $r(t',t)$:

$$\forall r \; [\; (\forall t_1 \exists t_2 : r(t_1,t_2)) \wedge (\forall t_1 \exists t_2 : r(t_2,t_1)) \;]$$

- Axiome zum gegenseitigen Ausschluß der Relationen:

$$o(x,y) \rightarrow \neg m(x,y) \quad \text{usw.}$$

- Axiome bzgl. der Inversen:

$$s(x,y) \rightarrow s_i(y,x) \quad \text{usw.}$$

- Axiome zur Beschreibung des „transitiven“ Verhaltens:

z.B.: $m(t_1,t_2) \wedge d(t_2,t_3) \rightarrow o(t_1,t_3) \vee d(t_1,t_3) \vee s(t_1,t_3)$
(siehe Tabelle, mit $con \leftrightarrow d_i \vee s_i \vee f_i$ und $dur \leftrightarrow d \vee s \vee f$)

ySz / xRy	b	b_i	d	d_i	o	o_i	m	m_i	s	s_i	f	f_i
before b	b	no info	$b\ o\ m\ d\ s$	b	b	$b\ o\ m\ d\ s$	b	$b\ o\ m\ d\ s$	b	b	$b\ o\ m\ d\ s$	b
after b_i	no info	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	$b_i\ o_i\ m_i\ d\ f$	b_i	b_i	b_i
during d	b	b_i	d	no info	$b\ o\ m\ d\ s$	$b_i\ o_i\ m_i\ d\ f$	b	b_i	d	$b_i\ o_i\ m_i\ d\ f$	d	$b\ o\ m\ d\ s$
contains d_i	$b\ o\ m_i\ d\ f$	$b_i\ o_i\ d_i\ m_i\ s_i$	$o\ o_i\ dur\ con\ =$	d_i	$o\ d_i\ f_i$	$o_i\ d_i\ s_i$	$o\ d_i\ f_i$	$o_i\ d_i\ s_i$	$d_i\ f_i\ o$	d_i	$d_i\ s_i\ o_i$	d_i
overlaps o	b	$b_i\ o_i\ d_i\ m_i\ s_i$	$o\ d\ s$	$b\ o\ m\ d_i\ f_i$	$b\ o\ m$	$o\ o_i\ dur\ con\ =$	b	$o_i\ d_i\ s_i$	o	$d_i\ f_i\ o$	$d\ s\ o$	$b\ o\ m$
overlapped by o_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ o_i\ dur\ con\ =$	$b_i\ o_i\ m_i$	$o\ d_i\ f_i$	b_i	$o_i\ d\ f$	$o_i\ b_i\ m_i$	o_i	$o_i\ d_i\ s_i$
meets m	b	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	b	b	$o\ d\ s$	b	$f\ f_i\ =$	m	m	$d\ s\ o$	b
met-by m_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	b_i	$o_i\ d\ f$	b_i	$s\ s_i\ =$	b_i	$d\ f\ o_i$	b_i	m_i	m_i
starts s	b	b_i	d	$b\ o\ m\ d_i\ f_i$	$b\ o\ m$	$o_i\ d\ f$	b	m_i	s	$s\ s_i\ =$	d	$b\ m\ o$
started-by s_i	$b\ o\ m\ d_i\ f_i$	b_i	$o_i\ d\ f$	d_i	$o\ d_i\ f_i$	o_i	$o\ d_i\ f_i$	m_i	$s\ s_i\ =$	s_i	o_i	d_i
finishes f	b	b_i	d	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	$b_i\ o_i\ m_i$	m	b_i	d	$b_i\ o_i\ m_i$	f	$f\ f_i\ =$
finished-by f_i	b	$b_i\ o_i\ m_i\ d_i\ s_i$	$o\ d\ s$	d_i	o	$o_i\ d_i\ s_i$	m	$s_i\ o_i\ d_i$	o	d_i	$f\ f_i\ =$	f_i

Axiomatik für Allen-Relationen

Modelle der Axiome sind Intervall-Strukturen für eine

- nicht-verzweigende
- in beiden Richtungen unbeschränkte Zeit.

Im Prinzip ist Reduktion möglich auf MEETS:

- Alle Relationen mittels MEETS definierbar,
- Axiome für MEETS.

Zeitliche Inferenzen

Es sind Beziehungen zwischen Intervallen gegeben

Weitere Beziehungen sollen abgeleitet werden (z.B. genaue Reihenfolge festlegen)

Allen-Axiome können als Constraints verwendet werden (insbesondere Tabelle)

Beispiel: (A, B, C, D sind Zeitintervalle)

A beginnt während B

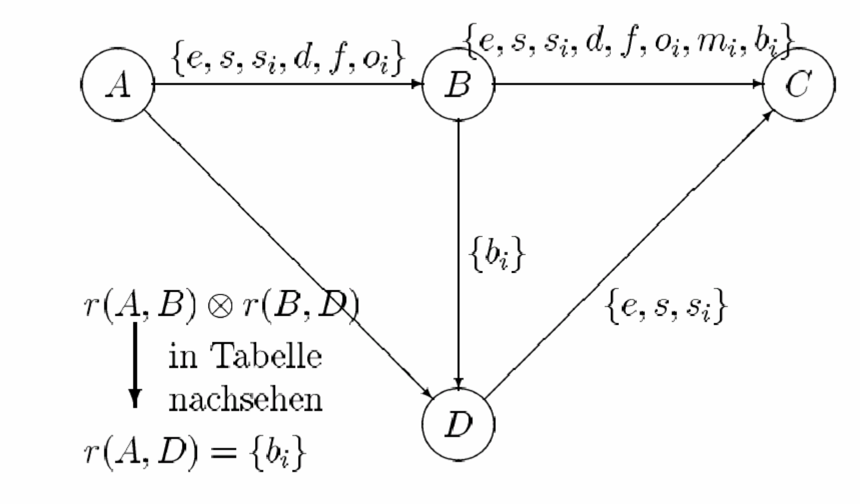
B beginnt nicht vor C

B liegt zeitlich völlig nach D

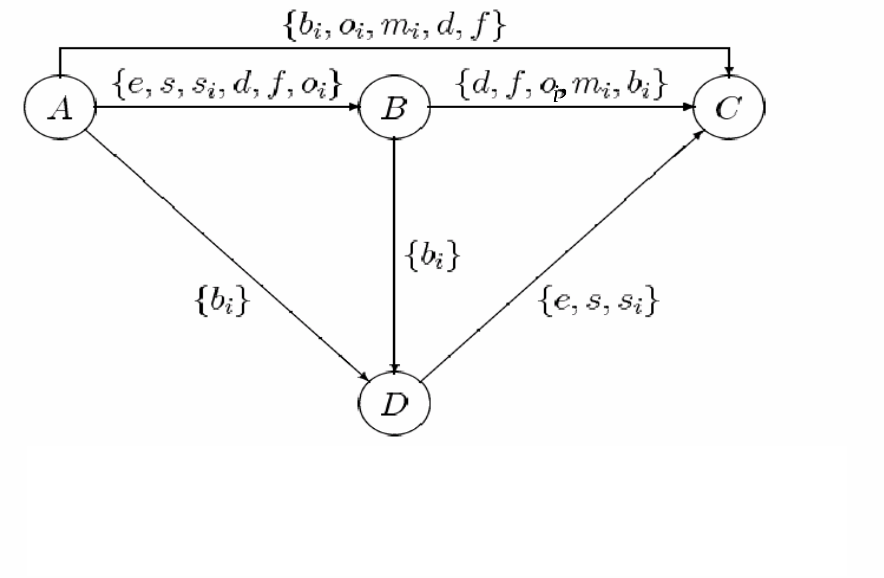
C und D beginnen gleichzeitig

- Was kann über Beziehungen zwischen A und D gesagt werden ?
- Können dafür Aussagen (Constraints) bzgl. A und B sowie B und D geeignet kombiniert werden?
- Führen solche Kombinationen zu verschärften Aussagen über die ursprünglichen Constraints?

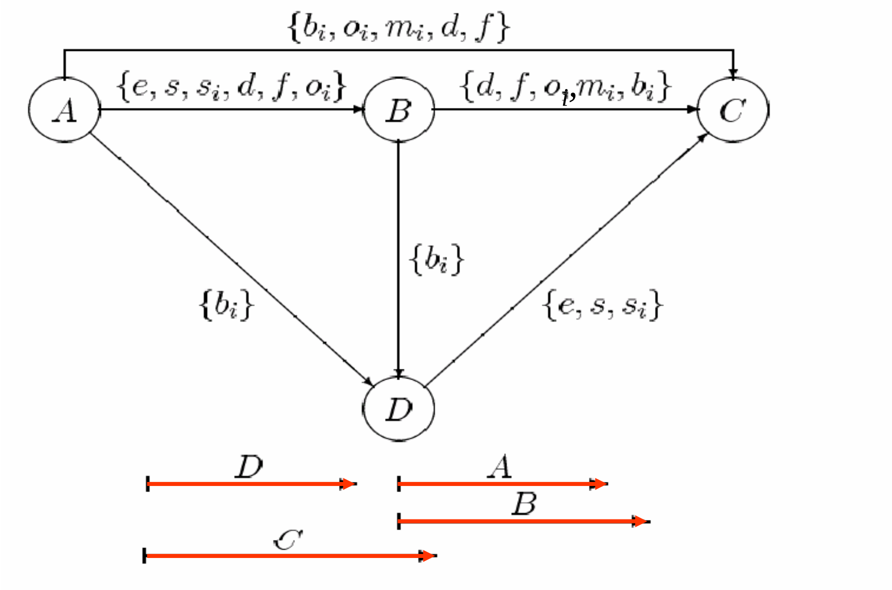
Darstellung als Netz



Eintragen der fehlenden Relationen



Auswertung



Allen-Netz

Ein Allen-Netz ist gegeben durch A = [T, C] mit

- T ist eine Menge von Intervall-Variablen und
- C ist eine Abbildung $C : T \times T \rightarrow 2^{\{s,si,\ldots,e\}}$

  $\{s, s_i, \ldots, e\}$ = Menge der Allen-Relationen.)
- Dabei gilt für alle t_1, t_2 aus T x T die Bedingung:

  $C(t_2, t_1)$ = Inverse der Relationen in $C(t_1, t_2)$

$C(t_1,t_2)$ gibt die zwischen den Intervallen t_1 und t_2 vorgegebenen Beziehungen an.
Ein Allen-Netz lässt sich als Graph („Netz“) mit Knoten T und Kantenbeschriftungen $C(t_1,t_2)$ darstellen.
Entspricht Constraint-Graph für binäre Constraints.

Modell eines Allen-Netzes A = [T, C]

Voraussetzung: Gegebene Intervallstruktur $[I, \{ \subseteq , < \}]$

Ein *Modell* für A = [T, C] ist eine Belegung $\beta: T \rightarrow I$ mit:

Für alle $t_1, t_2 \in T$ gilt:

$\beta(t_1), \beta(t_2)$ stehen in einer durch C zugelassenen Relation

d.h. $r(\beta(t_1), \beta(t_2)) \in c(t_1, t_2)$

(r ist eindeutig aufgrund der Axiome)

Globale Konsistenz:

A = [T, C] ist *global konsistent*, wenn es ein Modell besitzt

(andernfalls: *global inkonsistent*)

Betrachtung als Constraint-Problem

Variable t_i für Intervalle

Wertbereiche $Dom(t_i)$:

Menge I der Intervalle einer Intervallstruktur $[I, \{ \subseteq , < \}]$

Constraints:

– Axiome der Allen-Relationen

z.B. für $o(x,y) \rightarrow \neg m(x,y)$

$C = \{ [t_x,t_y] / o(t_x,t_y) \rightarrow \neg m(t_x,t_y)\}$

– Beziehungen gemäß Problemstellung

z.B. für „A beginnt während B":

$C = \{ [t_A,t_B] \ / \ e(t_A,t_B) \vee s(t_A,t_B) \vee s_i(t_A,t_B)$
$\vee d(t_A,t_B) \vee f(t_A,t_B) \vee o_i(t_A,t_B) \}$

Lokale Konsistenz

Lokale Konsistenz bei Allen-Netzen wird bezogen auf die Intervall-Axiome zur Beschreibung des „transitiven“ Verhaltens:

z.B.: $m(t_1,t_2) \wedge d(t_2,t_3) \rightarrow o(t_1,t_3) \vee d(t_1,t_3) \vee s(t_1,t_3)$
(siehe Tabelle)

Ein Netz $A = [T, C]$ ist *lokal konsistent* an der Stelle $\{t_1,t_2,t_3\}$

falls die Einschränkung $A/\{t_1,t_2,t_3\} = [\ \{t_1,t_2,t_3\}\ ,\ C/\{t_1,t_2,t_3\}\ ]$

von A auf $\{t_1,t_2,t_3\}$ global konsistent ist.

Das Teilnetz $A/\{t_1,t_2,t_3\}$ ist ein „Teildreieck“ von A.

Prüfung auf lokale Konsistenz

Gegeben: A = [T, C].

Überprüft werden „Dreiecke“ $\{t_1,t_2,t_3\}$ auf lokale Konsistenz.

Dabei werden die möglichen „Beschriftungen“ der Kanten (t_i,t_j) sukzessive verringert bis zur Stabilisierung.

Verringerungen ergeben sich aus Inkonsistenzen bzgl. der Intervall-Axiome des „transitiven“ Verhaltens.

Der Algorithmus benutzt

Einen Stack K für aktuell zu prüfende Intervallpaare und

Eine Abbildung

$R : T \times T \rightarrow 2^{\{s,si,\ldots,e\}}$ der aktuellen „Beschriftungen“

Prüfung auf lokale Konsistenz

Initialisierung:

K := Liste der Paare $(t_1, t_2) \in T \times T$,

$R(t_1, t_2) := C(t_1, t_2)$

Äußerer Zyklus:

Falls $K=[\,]$: EXIT(lokal konsistent),

sonst: $(t_1,t_2) := pop(K)$ und inneren Zyklus ausführen.

Innerer Zyklus:

Für alle $t \in T$ die Schritte (a) und (b) ausführen:

(a) Falls $R(t_1,t) \supset R(t_1,t) \cap (R(t_1,t_2) \times R(t_2,t))$:

setze $R(t_1,t) := R(t_1,t) \cap (R(t_1,t_2) \times R(t_2,t))$,

falls dann $R(t_1,t) = \varnothing$: EXIT(lokal inkonsistent) ,

andernfalls: $push(t_1,t)$ (einkellern).

(b) Falls $R(t, t_2) \supset R(t, t_2) \cap (R(t, t_1) \times R(t_1,t_2))$:

setze $R(t, t_2) := R(t, t_2) \cap (R(t, t_1) \times R(t_1,t_2))$,

falls dann $R(t, t_2) = \varnothing$: EXIT(lokal inkonsistent),

andernfalls: $push(t, t_2)$ (einkellern)

Evaluierung des Verfahrens

Der Algorithmus bricht nach maximal $O(n^3)$ Schritten ab.

(n = Anzahl der Knoten im Allen-Netz).

$O(n^2)$ für äußere Schleife:

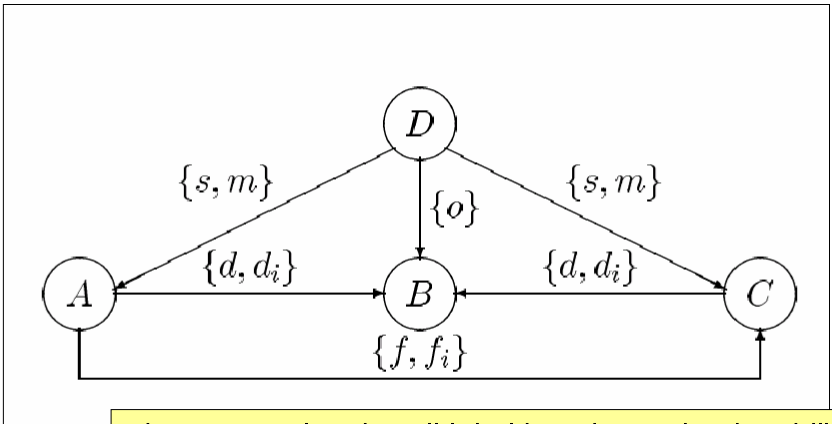
höchstens 13mal (Anzahl der Relationen)

für jedes der n^2 Paare $(t_1, t_2) \in T \times T$,

$O(n)$ für innere Schleife.

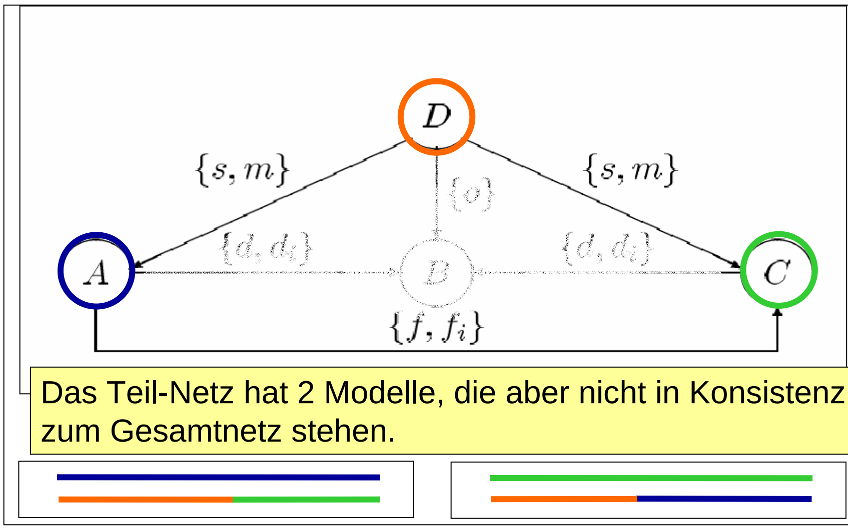
Beim Abbruch wird das korrekte Resultat bzgl. lokaler Konsistenz geliefert.

Lokale vs. Globale Konsistenz



Dieses Netz ist überall lokal konsistent (und stabil)

Lokale vs. Globale Konsistenz



Das Teil-Netz hat 2 Modelle, die aber nicht in Konsistenz zum Gesamtnetz stehen.

Anwendung des Verfahrens

Es gibt lokal konsistente Allen-Netze, die nicht global konsistent sind.

- Verfahren prüft auf lokale Konsistenz
- Wenn diese nicht vorliegt, ist das Netz auch global inkonsistent
- Bei lokaler Konsistenz kann unter den verbliebenen Möglichkeiten nach einer globalen Lösung gesucht werden.

Constraints: Weiteres

- „Harte“ Constraints
- „Weiche“ Constraints

- Verbindung mit logischer Programmierung:

  Constraint-logische Verfahren

Abschluss: Was ist die Idee bei CSP?

Zusammenhänge im Suchraum explizit machen:

Einschränkende Bedingungen für Lösungen

Ziel: Keine Suche in Bereichen, die keine Lösung enthalten können.

Modellierung

Suchraum als Parameterraum.

Constraints:

- Beschränkungen zwischen Werten der Parameter
- Lösung muss allen Beschränkungen genügen

Abschluss: Was ist die Idee bei CSP?

Lösungsraum einschränken:

Zusätzliche Constraints schränken Lösungsmenge ein.

Besonders bequem:

Einschränkungen der Definitions-Bereiche.

Constraint-Propagation:

- Definitions-Bereiche sukzessive einschränken.

Ergebnis ist ein kleinerer Suchraum.

- am Ende steht meist wieder eine Suche.

Verzicht auf Suche in Bereichen,
die keine Lösung enthalten können.

CSP: Zerlegungen des Problemraums

Und-Zerlegung (= Problemzerlegung)
(separierte Constraint-Netze)

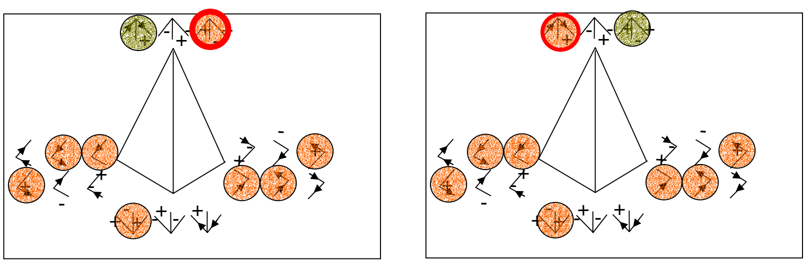
Zerlegung der Variablenmenge
Probleme einzeln lösen,
Gesamtergebnis aus Einzel-Ergebnissen zusammengesetzt

Beispiele:

- Bildverarbeitung: nicht zusammenhängende Objekte
- Allen-Netze: Nicht verknüpfte zeitliche Angaben
- Färbungsproblem für Inseln

CSP: Zerlegungen des Problemraums

Oder-Zerlegung für eine Variable v
mit (möglichst eingeschränktem) Wertebereich W
Aufteilung: Suche in Teilräumen für jeden Wert $w \in W$:



Es genügt, eine Lösung in einem Teilraum zu finden

CSP: Zerlegungen des Problemraums

Kombination von und/oder-Suche.

Beispiel:

Annahme: Constraint-Graph $G=[V,C]$ kann aus zwei Teilgraphen $G_1 =[V_1,C_1]$ und $G_2 =[V_2,C_2]$ kombiniert werden, die nur einen Knoten v gemeinsam haben:

$V_1 \cap V_2 = \{v\}$, W sei Wertebereich von v

Dann kann für jedes $w \in W$ die Lösung in den separierten Graphen G_1 und G_2 gesucht werden.

Oder-Zerlegung: Alternativen für $w \in W$

Und-Zerlegung: Suche in G_1 und G_2

Lokale Suche für CSP

Suchraum: Belegungen der Parameter mit Werten.

Start mit einer (Erfolg versprechenden) Belegung

Sukzessive Werte einer Variablen ändern,

so dass Zahl der Constraint-Verletzungen sinkt

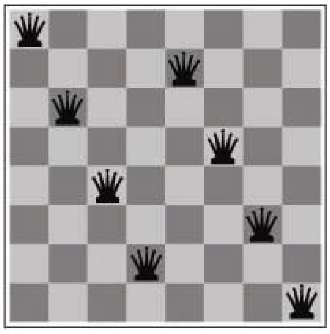
(Heuristik für Suche)

Beispiel: 8-Damen-Problem (s.u.)

Ansatz oft sehr erfolgreich, wenn es viele im Parameterraum gleichmäßig verteilte Lösungen gibt.

Fallstudie: 8-Damen-Problem

8 Damen auf dem Schachfeld so platzieren, dass keine eine andere angreifen kann (im Beispiel nicht erfüllt)



Fallstudie: 8-Damen-Problem

Behandlung als Suchproblem

Formulierung S1:

Zustände: 0...8 Damen in beliebiger Position auf dem Schachbrett
Ausgangszustand: leeres Brett
Zielzustand: 8 Damen auf dem Brett, keine angegriffen
Zustandsübergang: eine Dame auf ein freies Feld stellen

Komplexität:
$64 \cdot 63 \cdot 62 \cdot 61 \cdot 60 \cdot 59 \cdot 58 \cdot 57 \approx 3 \cdot 10^{14}$ Folgen untersuchen

Formulierung S2:

Zustandsübergang: Dame auf nicht-angegriffenes Feld setzen
Sonst wie S1

Fallstudie: 8-Damen-Problem

Behandlung als Suchproblem

Formulierung S3:

Zustände: 0...8 Damen auf dem Schachbrett,
in jeder Spalte höchstens 1 Dame,
Links liegende Spalten belegt, rechte Spalten frei

Zustandsübergang:
Wähle die am weitesten links liegende freie Spalte
Setze eine Dame auf ein nicht angegriffenes Feld in dieser Spalte

Sonst wie S1

Komplexität: 2057

Fallstudie: 8-Damen-Problem

Behandlung als heuristisches Suchproblem

Formulierung S4 (Bergsteigen, „greedy search“):

Zustände: 8 Damen auf dem Schachbrett,
in jeder Spalte genau 1 Dame

Anfangszustand: Beliebig aus dieser Menge

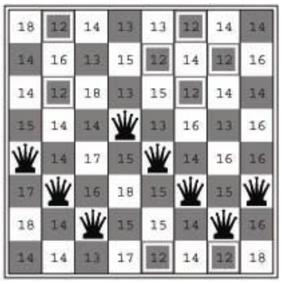
Zielzustand: 8 Damen auf dem Brett, keine angegriffen

Zustandsübergang: Eine Dame in ihrer Spalte verschieben

Heuristik: Anzahl der sich direkt oder indirekt angreifenden
Damenpaare (indirekter Angriff: Es steht eine Figur dazwischen).

Fallstudie: 8-Damen-Problem

Heuristik h: Anzahl der sich direkt oder indirekt angreifenden Damenpaare (indirekter Angriff: Es steht eine Figur dazwischen).

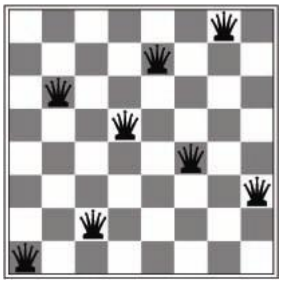


Angabe der h-Werte

Bei Verschieben einer Dame

Innerhalb einer Spalte

Lokales Minimum (h=1)



Fallstudie: 8-Damen-Problem

Anwendung genetischer Algorithmen

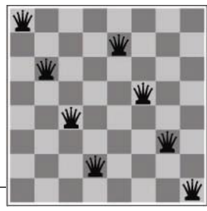
Formulierung G

Individuen: 8 Damen auf dem Schachbrett,
in jeder Spalte genau 1 Dame

Fitness: Paare von Damen, die sich nicht gegenseitig angreifen
(f=28 für Lösung)

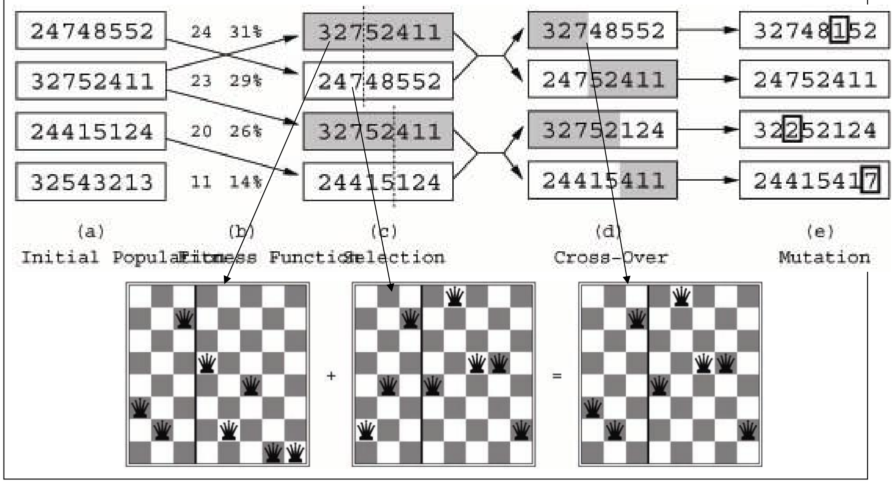
Genetische Kodierung eines Individuums durch 8 Ziffern:
Angabe der Zeilen, in denen die Damen stehen.

86427531



Fallstudie: 8-Damen-Problem

Populationen verändern durch Mutation, Kreuzung



Fallstudie: 8-Damen-Problem

Behandlung als Constraint-Problem

Formulierung C:

Variable $v_1,...,v_8$ mit Wertebereichen $Dom(v_i)=\{1,..,8\}$

($v_i = j$ bedeutet: Dame in der Spalte i steht auf Zeile j)

Constraints C_{kl} für Variablenpaare $[v_k,v_l]$, $1\leq k<l\leq 8$:

C_{kl} = „Dame in Spalte k greift Dame in Spalte l nicht an“

Fallstudie: 8-Damen-Problem

Behandlung als Constraint-Problem mit lokaler Suche

Formulierung CL:

Variable $v_1,\ldots,v_8$ mit Wertebereichen $Dom(v_i)=\{1,..,8\}$

($v_i = j$ bedeutet: Dame in der Spalte i steht auf Zeile j)

Constraints C_{kl} für Variablenpaare $[v_k,v_l]$, $1 \leq k < l \leq 8$:

C_{kl} = „Dame in Spalte k greift Dame in Spalte l nicht an“

Zustandsraum: Variablenbelegungen (d.h. Stellungen wie in S3)

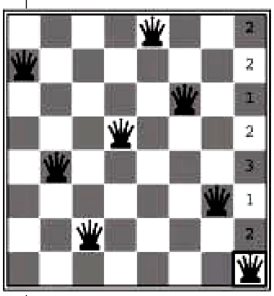
Ausgangszustand: Eine Variablenbelegung

Zustands-Übergang: Änderung des Wertes einer Variablen

Heuristik: Constraint-Verletzungen minimieren

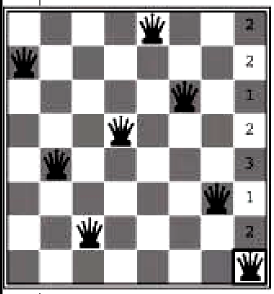
Fallstudie: 8-Damen-Problem

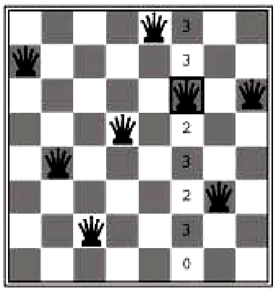
Behandlung als Constraint-Problem mit lokaler Suche



Fallstudie: 8-Damen-Problem

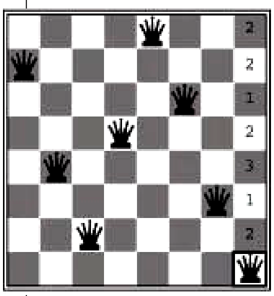
Behandlung als Constraint-Problem mit lokaler Suche

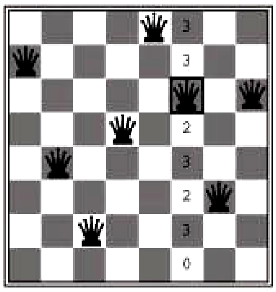


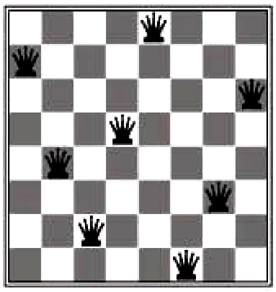


Fallstudie: 8-Damen-Problem

Behandlung als Constraint-Problem mit lokaler Suche







Lösung wird schnell gefunden
Selbst noch 1.000.000-Damen-Probleme in ca. 50 Schritten!