Einführung in die KI

Prof. Dr. sc. Hans-Dieter Burkhard
Vorlesung Winter-Semester 2006/07

2. Constraints:
   - Grundbegriffe
   - Constraint Propagation
   - Anwendungsbeispiele
     (Bildinterpretation, zeitliches Schließen)

Beschränkungen (Constraints)

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

- Produktionsplanung
- Stundenplanung
- Scheduling
- Landkarten einfärben, ...

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

Constraint: Definition

Geg.: Variablenmenge $V = \{ v_1, ..., v_s \}$
mit Wertebereichen $Dom(v)$ für $v \in V$

$$Dom(V) := X_{v \in V} Dom(v) = Dom(v_1) \times ... \times Dom(v_s) .$$

Constraint C gibt an,
welche Werte-Kombinationen zugelassen sind:

$$C \subseteq Dom(v_1) \times ... \times Dom(v_s)$$

Constraint-Erfüllungs-Problem (CSP)

Ein Constraint-Erfüllungs-Problem
(Constraint Satisfaction Problem, CSP, Constraint-Netz)
über einer Variablenmenge $V = \{ v_1, ..., v_n \}$

ist gegeben
durch eine Menge $\mathcal{C} = \{C_1, ..., C_k\}$ von Constraints,
wobei die Constraints C_j jeweils über einer Menge $V_{Cj} \subseteq V$
definiert sind.

Belegungen, Lösungen

Belegung $\beta \in Dom(V)$
ordnet jedem $v \in V$ einen Wert $\beta(v)$ aus $Dom(v)$ zu:

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

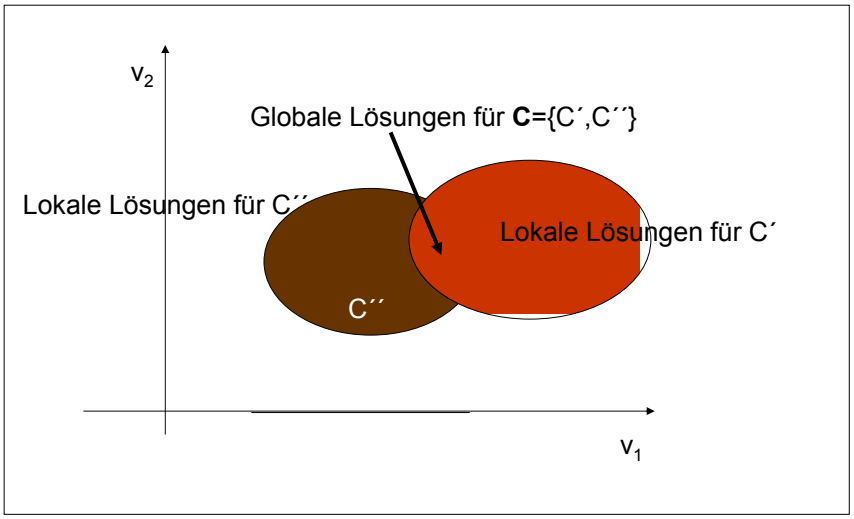
β erfüllt Constraint C über $V_C = \{v^C_1, ..., v^C_m\} \subseteq V$,
falls $[\, \beta(v^C_1), ..., \beta(v^C_m) \,] \in C$

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

Belegung β erfüllt $\boldsymbol{\mathcal{C}} = \{C_1, ..., C_k\}$, falls β alle $C_j \in \boldsymbol{\mathcal{C}}$ erfüllt.

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

Beispiele



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}

lokale Lösung für C_3 ist [0.5, 0.7, 0.5] .

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

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

Constraint-Optimierung (Constraint Optimization)

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

Methode:

Zusätzliche Constraints C(r): = {$\beta \mid \beta \in Dom(V) \wedge c(\beta) \geq r$ }

einführen und als CSP für verschiedene $r \in R$ untersuchen.

Constraint-Graphen

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

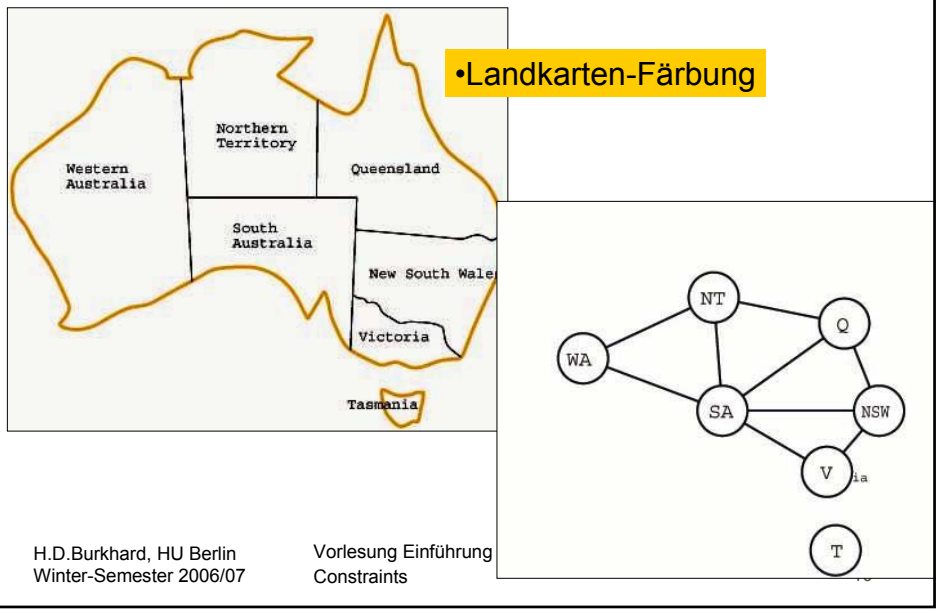
CSP $\mathcal{C} = \{C_1, ..., C_k\}$ aus binären Constraints C_j

ist darstellbar als Graph $G=[V, \mathcal{C}]$ mit

Knoten: Variable v

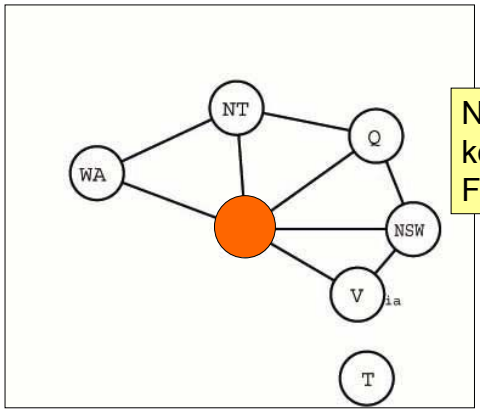
Kanten: Für jedes Constraints $C \in \mathcal{C}$ verbindet eine Kante die beiden durch C beschränkten Variablen

Constraint-Graphen

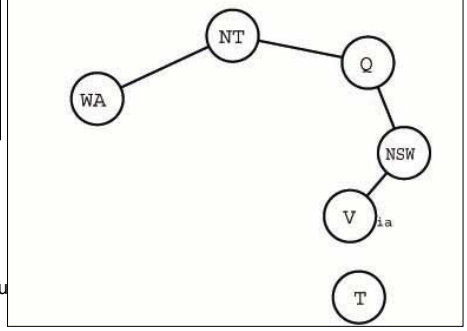


•Landkarten-Färbung

Vereinfachende Zerlegung des Constraint-Graphen



Nach Einfärbung von SA
können andere mit verbleibenden
Farben versehen werden

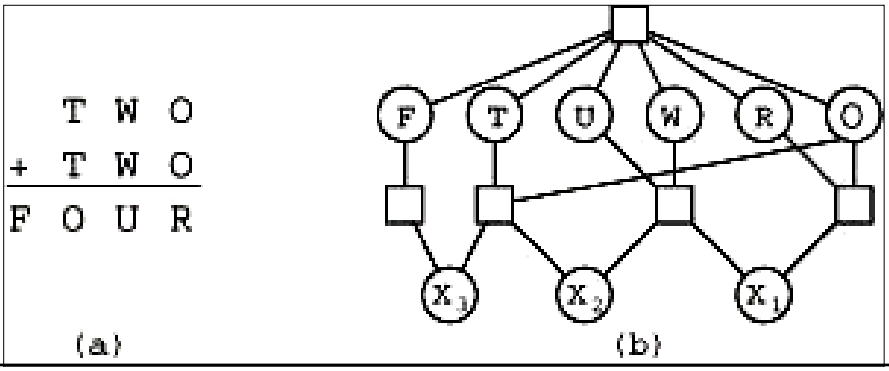


Constraint-Hyper Graphen

Knoten: Variable v

Hyper-Knoten: Constraints C über $V_C \subseteq V$

Kanten: verbinden Hyperknoten für Constraints C
mit Knoten für Variable $v \in V_C$



Constraint-Graphen

Jedes CSP kann in ein binäres CSP überführt werden.

Viele Verfahren sind auf binäre Csp beschränkt.

z.B. Vereinfachende Zerlegungen des Constraint-Graphen.

Übergang zu Binärem CSP

Gegeben CSP $\boldsymbol{C} = \{C_1, ..., C_k\}$

mit Constraints C über $V_C = \{v^C_1, ..., v^C_m\}$

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

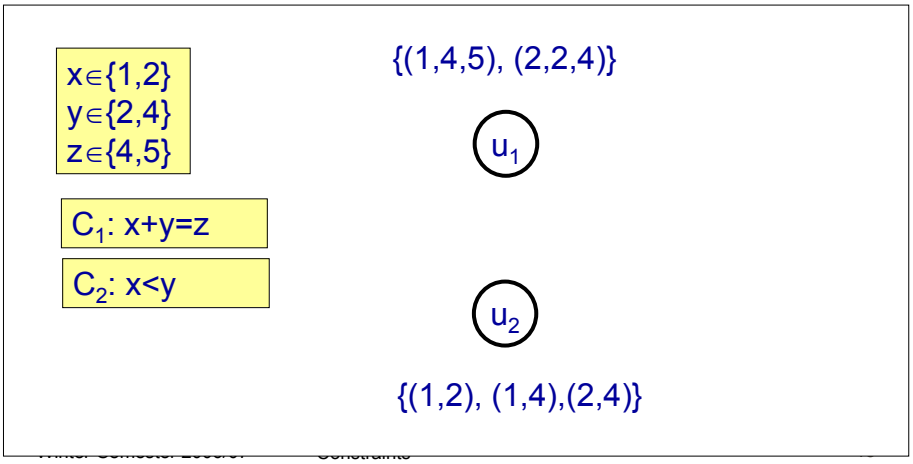
Unäre Constraints ($card(V_C) = 1$) in $Dom(v^C)$ integrieren.

Konstruieren binäres CSP

mit Variablen u_C für jedes Constraint $C \in \boldsymbol{C}$

und $Dom(w_C) = C$

Übergang zu Binärem CSP



Übergang zu Binärem CSP

Belegungen der neuen Variablen müssen Gleichheiten der ursprünglichen Variablen respektieren:

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

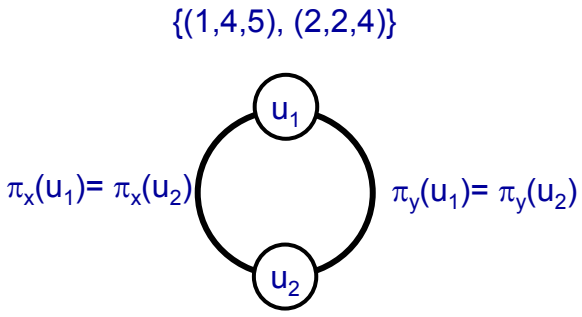
jeweils definiert für die Variablen $v \in V_{Ci} \cap V_{Cj}$:

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

dabei bezeichnet $\pi_v(b_i)$
die Projektion von b_i
an der Stelle von v

Übergang zu Binärem CSP

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



Lösungsverfahren für CSP

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

Beispiel:

Diophantische Gleichungen

Endliche Constraint-Probleme (Dom(V) endlich)
sind oft NP-vollständig

Suchverfahren für Constraints

Zustände: (Teil-)Belegungen der Parameter mit Werten

Zielzustände: $Z_{final} = \{ \beta \mid \beta \text{ erfüllt } \boldsymbol{C} = \{C_1, ..., C_k\} \}$

Zustandsübergang: Festlegung/Änderung der Parameterwerte

Bewertungen

Heuristiken

Backtracking-Verfahren

- Start mit „leerer“ Belegung β
- Für alle Variablen $v \in V = \{ v_1, ..., v_n \}$ der Reihe nach
  - Für alle Werte $w \in Dom(v)$:
    - Falls w konsistent bzgl. aller Constraints mit bisheriger Belegung: $\beta(v) := w$
  - Falls kein konsistenter Wert zuweisbar: Backtracking zu nicht versuchtem Wert der vorherigen Variablen.
- Falls β vollständig: Lösung β gefunden, andernfalls existiert keine Lösung.

Heuristiken

Forward-Checking:

- Bei Auswahl des Wertes w werden bei allen nicht belegten Variablen v‘ die Werte aus Dom(v‘) gestrichen, die bzgl. eines Constraints in Konflikt zu w stehen.
- Vorteile:
  - Suchraumeinschränkung durch Verringerung der Alternativen für spätere Wertzuweisungen.
  - Falls Dom(v‘)=∅ für eine Variable v‘ kann bereits w als Wertzuweisung zurück genommen werden.

Heuristiken für Auswahl der Variablen

Minimum Remaining Values

(most constrained variable, fail-first-Heuristik):

- Variable mit den wenigsten Möglichkeiten für Wertzuweisungen bevorzugen.

Degree-Heuristik:

- Variable mit den meisten Constraints bzgl. nicht belegter Variablen bevorzugen

Lokale heuristische Suche

Zustände:	Belegungen aller Parameter mit Werten Z = Dom(V)
Zielzustände:	Z_{final} = { β | β erfüllt ***C*** = {C_1, ..., C_k} }
Zustandsübergang:	Änderung eines Parameterwertes
Zustandsbewertung:	v(z) := Anzahl der Konflikte (verletzte Constraints) bei z

Implementierung z.B. mittels Bergsteigen
Heuristik: Bewertung v minimieren

Lokale heuristische Suche

Gute Resultate selbst für schwere Probleme, wenn Lösungen überall im Lösungsraum „dicht“ verteilt sind.

Andernfalls z.B. Evolutionäre Algorithmen anwenden.

Propagierung von Beschränkungen

Suchraumreduktion: Einschränkung der Wertebereiche Dom(v) durch Weitergeben von Beschränkungen

bei Forward-Checking: 1-Schritt-Vorausschau

Constraint-Propagation:

Schritte (P),(E),(S) in beliebiger Reihenfolge mit Wiederholungen:

(P) Weiterreichen (Propagieren) lokaler Einschränkungen zur Beschränkung von Wertebereichen

(E) Willkürliches Einschränken von Wertebereichen (nicht nur Zuweisung eines Wertes)

(S) Suche im verbliebenen Suchraum

Bildinterpretation: Ablauf

Reale Szene: 3D

Elektronisches Abbild: 2D (Pixel-Matrix)

Filter (Vorverarbeitung)

Segmentierung

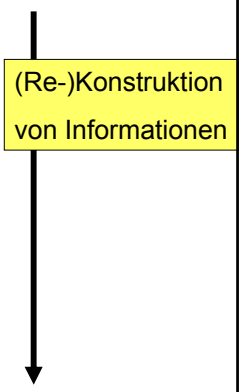
Identifikation von Linien

Interpretation von Linien (Beschriftung)

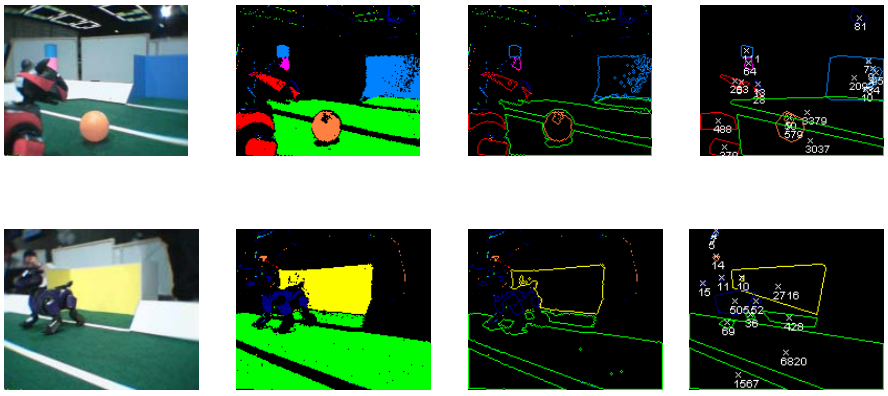
Identifikation von Objekten

Beziehungen zwischen Objekten

Szenen-Interpretation



Farb- und Kantensegmentierung



Kennzeichnung von Linien

Begrenzungslinien

- äußere Kante eines Objekts. Kennzeichnung: ,,→" .
- Pfeile der Begrenzungslinien so gerichtet, daß die Körperfläche rechts liegt.

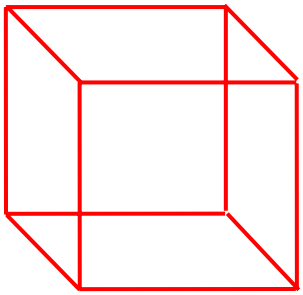
Innenlinien (Kanten im Inneren des Objekts)

– **Konvexe Linien:**
  - Beide Grenzflächen vom Beobachter abgewandt, Kennzeichnung: ,,+" .

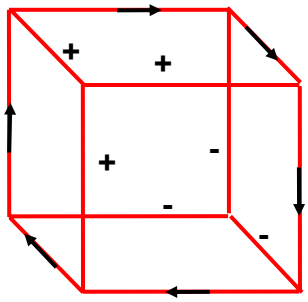
– **Konkave Linien:**
  - Schnittline der Begrenzungsflächen vom Beobachter abgewandt,
  - Kennzeichnung: ,,-"

„Schlecht gestelltes Problem“

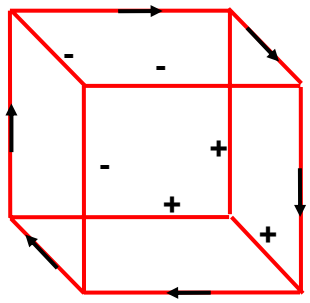
Mehrere 3D-Szenen zu 2D-Bild möglich



Würfel

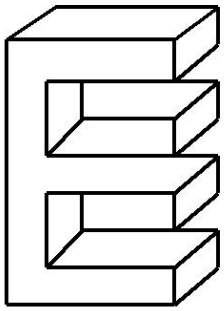


Würfel



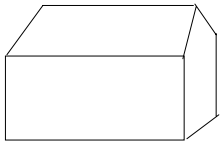
Interpretationen im Wettstreit

Vermischung unterschiedlicher Interpretationen



2D-Bild eines 3D-Körpers interpretieren

Annahmen zur Vereinfachung
(andernfalls komlexere Verfahren,
mehr Constraints, weniger Lösungen (!))



- Keine Schatten oder Bruchlinien.
- Verdeckte Kanten sind nicht sichtbar.
- Alle Eckpunkte sind Schnittpunkte genau dreier aufeinandertreffender Flächen.
- ,,Allgemeiner Beobachtungspunkt":
  Bei geringen Ortsveränderungen des Beobachters darf kein Schnittpunkt seinen Typ wechseln.

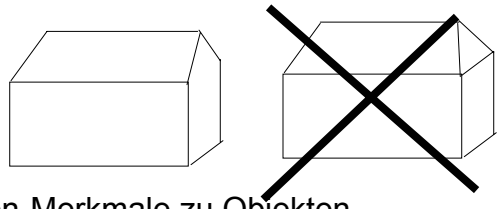
Objekte identifizieren

Erkennung eines Objektes durch Interpretation der Linien

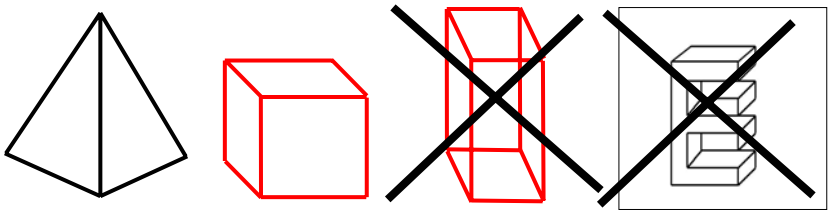
Konvex

Konkav

Begrenzung

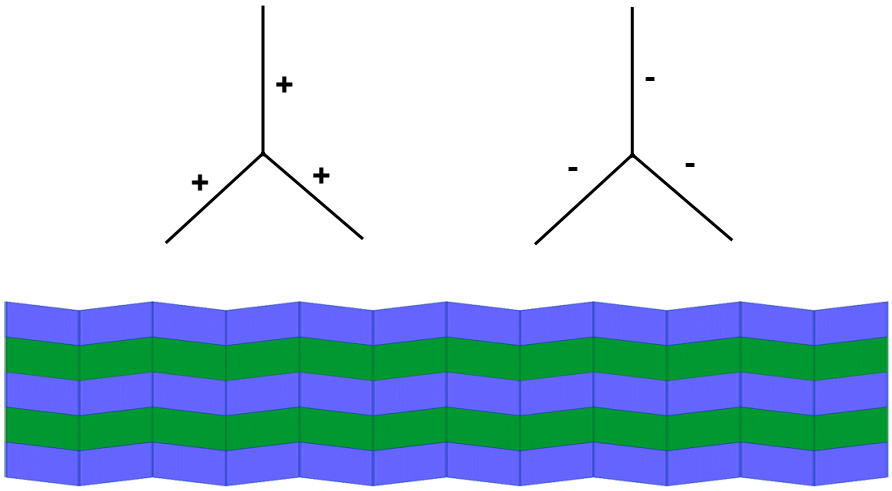


Zuordnungen: Linien-Merkmale zu Objekten

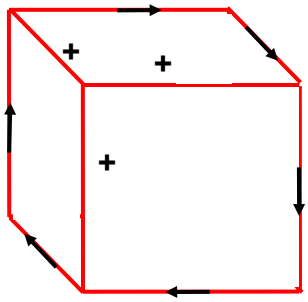


Konkav oder konvex?

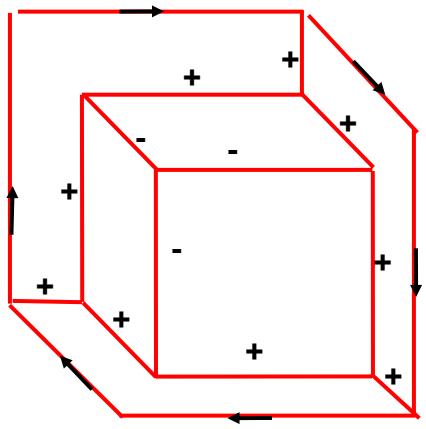
Linieninterpretationen sind abhängig vom Kontext



Würfel



Würfel



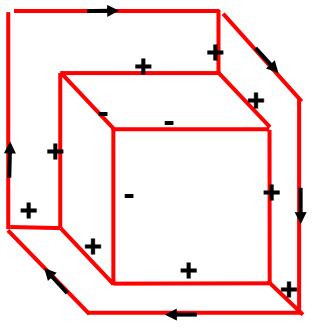
Beschriftung

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

Identifizierung von Objekten anhand ihrer Beschriftungen.

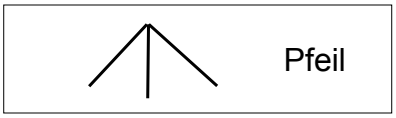
Nicht alle Beschriftungen sind möglich (Constraints).

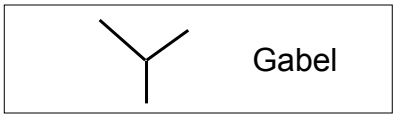
Suche nach zulässigen Beschriftungen

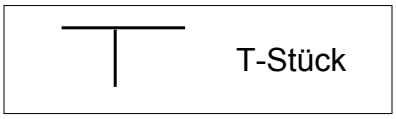


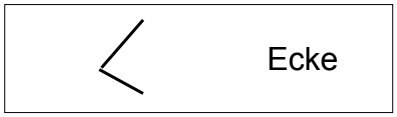
4 Typen von Schnittpunkten

(aufgrund der Voraussetzungen)







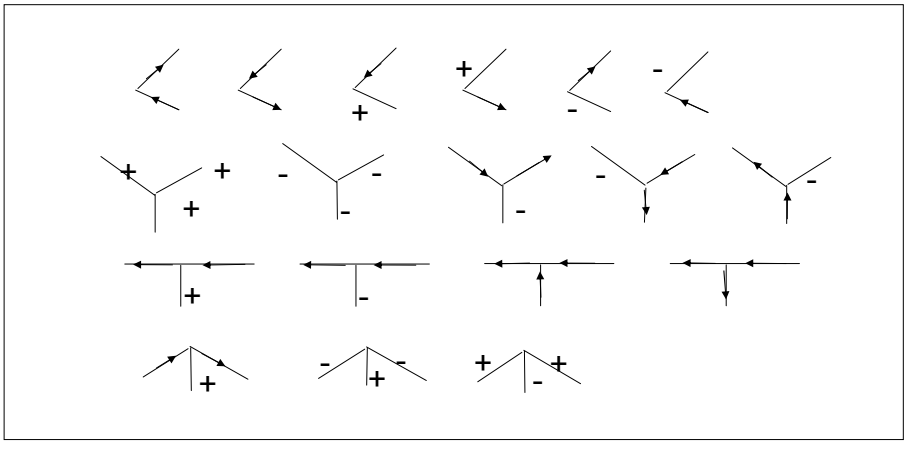


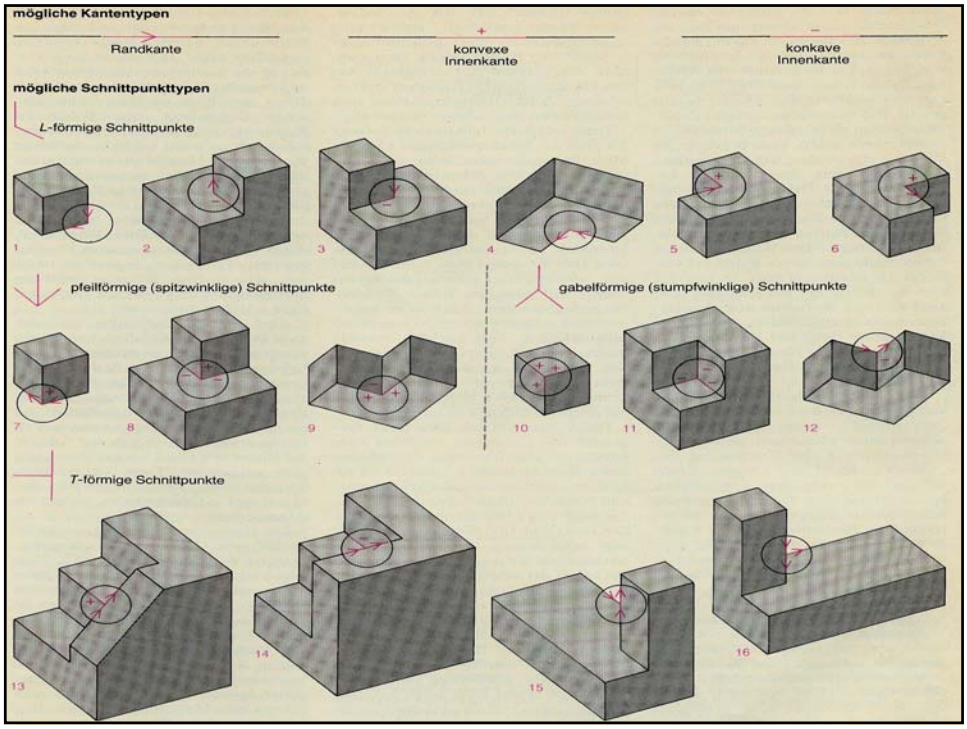
Beschriftung an Ecken

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

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

Davon aber nur 18 physikalisch möglich (**Constraints**)





Suchverfahren für Kanten-Beschriftungen

Kanten (*Variable*) sukzessive mit Werten aus {„←“, „→“, „–“ , „+“} beschriften.

Constraints beachten: an den Ecken dürfen nur zulässige Typen auftreten.

Resultate

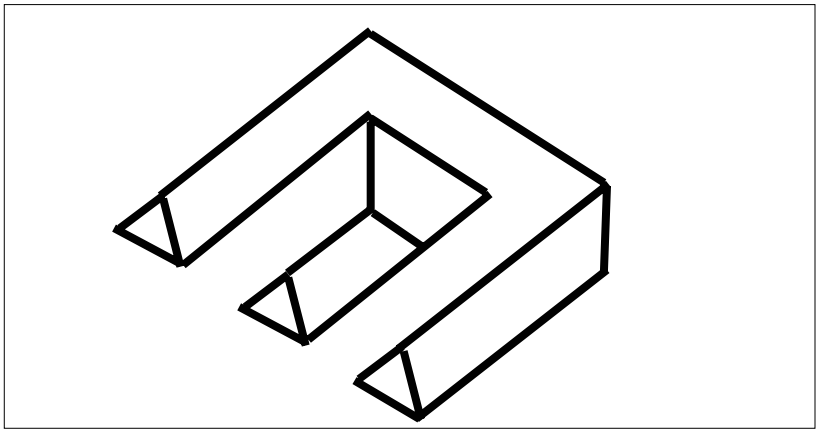
Das Verfahren testet auf lokale Konsistenz.

Wenn keine lokale Konsistenz erreichbar (bei allen Varianten Misserfolg wegen leerer Menge an einer Ecke) kann es keine global konsistente Beschriftung der Kanten geben.

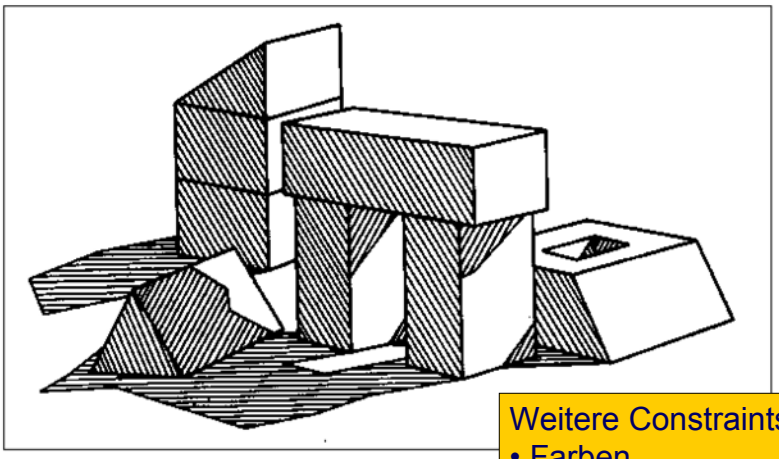
Auch bei lokaler Konsistenz kann es sein, dass keine global konsistente Beschriftung der Kanten existiert.

Beschriftungsverfahren

Gibt es konsistente Beschriftung?



Weitere Constraints: Schatten, ...



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

Algorithmus AC-3

Testet lokale Konsistenz („Kantenkonsistenz“) für binäre Constraintgraphen G=[V, **C**]

- Verwendet Warteschlange Q für Variablen-Paare [v‘, v‘‘] und
- Einschränkungen D(v) $\subseteq$ Dom(v) für Variable v

Initial:

- Q :={ $[v_i,v_j]$ | $\exists C \in \mathbf{C}$: $\{v_i,v_j\} = V_C$ }
  (alle Paare $[v_i,v_j]$, die durch Kante verbunden sind)
- D(v) := Dom(v) für alle Variablen v

Algorithmus AC-3

Führe wiederholt folgende Schritte aus:

- Falls Q leer: EXIT(„lokal konsistent“)
- Entferne das erste Paar [v, v‘] aus Q .
  C sei das zugeordnete Constraint
- Entferne alle Werte x aus D(v) ,
  für die kein Wert $y \in D(v')$ mit $[x,y] \in C$ existiert.
- Falls $D(v) = \varnothing$: EXIT(„inkonsistent“)
- Falls D(v) verkleinert wurde:
  Für alle mit v verbundenen v‘‘ : Füge [v‘‘,v] zu Q hinzu.

Algorithmus AC-3

Für endliche CSP bricht Algorithmus ab mit Resultat:

- lokal inkonsistent.
  Dann ist das CSP nicht lösbar.

oder

- lokal konsistent („Kantenkonsistenz“). Dann gilt:
  Falls das CSP lösbar ist, muss eine global konsistente Lösung die Werte aus den resultierenden Einschränkungen D(V) verwenden.
  (Suchraum wurde verkleinert)

Komplexität: Maximal $n^2 d^3$ Schritte für
$n = card(V)$
$d = card\ (Dom(v)\ )$

Constraint-Propagierung allgemein

Auch für nicht binäre CSP

Schritte für Lösungssuche:

P) Wertebereiche einschränken (Constraint-Propagation)

E) Wertebereiche willkürlich einschränken

(bei Bedarf Backtracking)

S) Suche über verbliebenen Werten

P, E, S können mehrfach und abwechselnd angewendet werden

Constraint-Propagierung: Einschränkungen

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

Betrachten *„Einschränkungen" der Wertebereiche*

$D_i \subseteq Dom(v_i)$, i=1, ...,n

$D = D_1 \times ... \times D_n$ ← z.B. als Intervall im R_n

(„zusätzliche Constraints")

Constraint-Propagierung: Einschränkungen

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

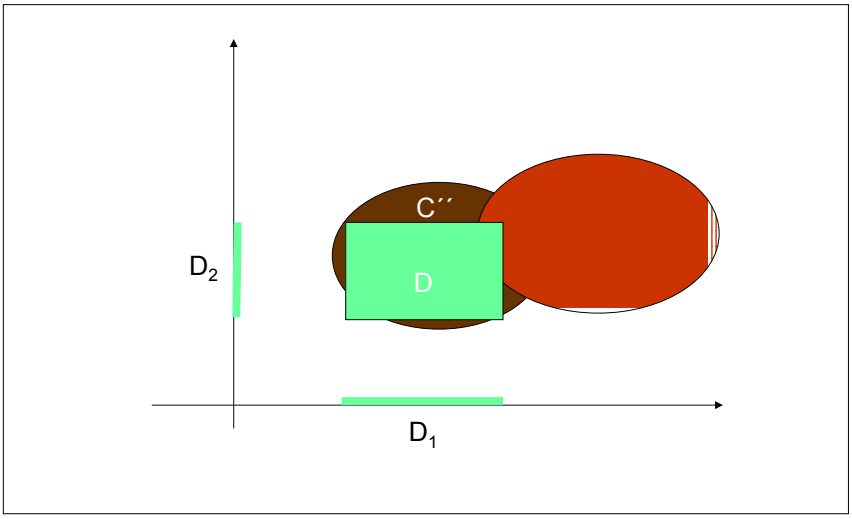
D heißt *global konsistente Einschränkung*, falls gilt:

$\forall i \in \{1,...,n\}\ \forall d_i \in D_i$

$\exists \beta = [a_1,...,a_{i-1},d_i,a_{i+1},...,a_n] \in D$: β globale Lsg. für **C**

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

Eine Lokal konsistente Einschränkung bzgl. C‘‘



Maximale lokal konsistente Einschränkung

Innerhalb einer Menge $D = D_1 \times ... \times D_n$

gibt es für jedes Constraint C_j

genau eine ***maximale*** *lokal konsistente Einschränkung*

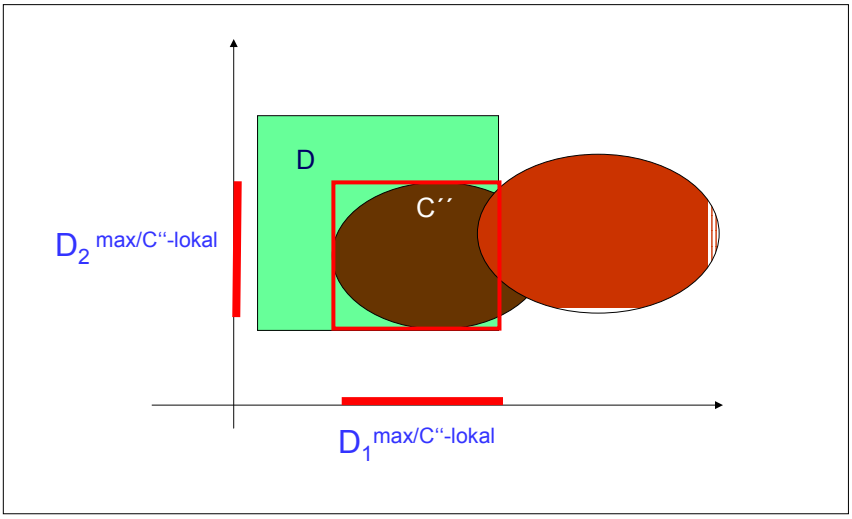
$D^{max/Cj\text{-}lokal} = D_1^{max/Cj\text{-}lokal} \times ... \times D_n^{max/Cj\text{-}lokal}$

mit $D_i^{max/Cj\text{-}lokal}$

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

(Projektion der in D enthaltenen lokalen Lösungen auf den Wertebereich $Dom(v_i)$.)

Maximale lokal kons. Einschränkung bzgl. C‘‘ in D



Maximale lokal kons. Einschränkung bzgl. C‘ in D

Lokale Lösung für C_j existiert in D

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

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

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

Maximale global konsistente Einschränkung

Innerhalb einer Menge $D = D_1 \times ... \times D_n$

gibt es genau eine *maximale global konsistente Einschränkung*

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

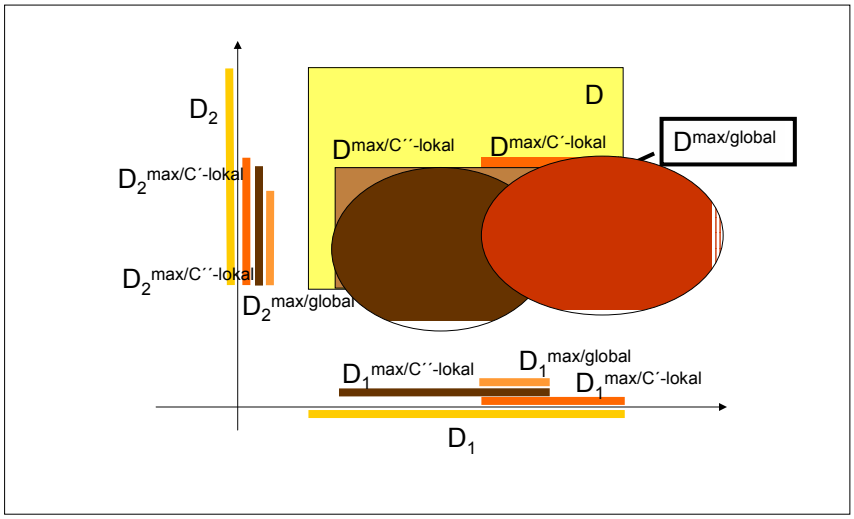
mit $D_i^{max/global}$

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

(Projektion der in D enthaltenen globalen Lösungen

auf den Wertebereich $Dom(v_i)$.)

Maximale konsistente Einschränkung



Maximale global konsistente Einschränkung

Globale Lösung existiert in D

gdw. nicht-leere global konsistente Einschränkungen von D existieren

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

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

$D^{imax/global} \subseteq D^{imax/Cj-lokal}$.

Es gibt keine globale Lösung in D, falls ein $D^{imax/Cj-lokal} = \varnothing$

Max. lokal kons. Einschränkung und globale Lsg.

Falls $D^{max/Cj\text{-}lokal} = \varnothing$ für ein Constraint C_j ,

(falls $D_i^{max/Cj\text{-}lokal} = \varnothing$ für ein Constraint C_j und eine Variable v_i)

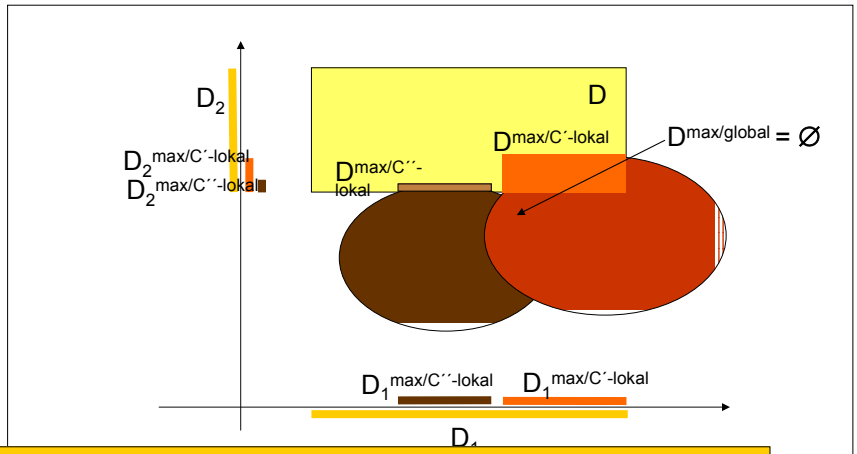
so gilt $D^{max/global} = \varnothing$ und es gibt keine globale Lösung in D

Umkehrung gilt nicht:

Auch wenn für alle Constraints C_j gilt: $D^{max/Cj\text{-}lokal} \neq \varnothing$,

kann $D^{max/global} = \varnothing$ gelten und es gibt keine globale Lösung in D

Alle $D^{max/Cj\text{-}lokal} \neq \varnothing$ und gleichzeitig $D^{max/global} = \varnothing$

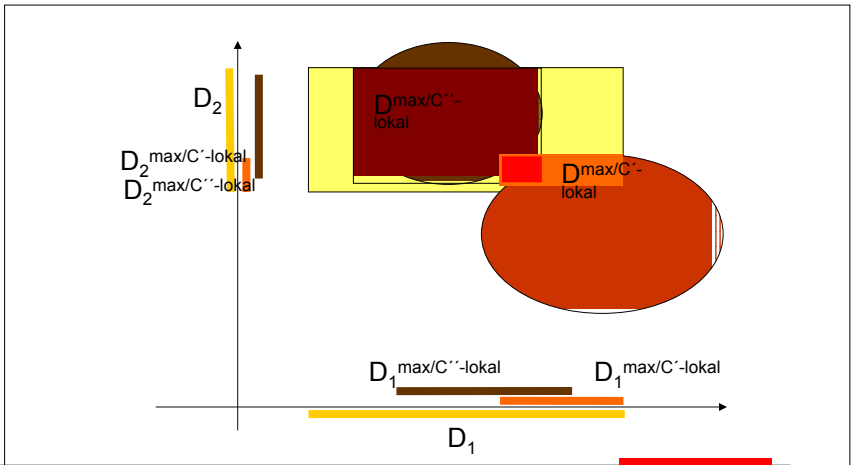


Frage: Wäre $\bigcap D^{max/Cj\text{-}lokal} \neq \varnothing$

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

und die Existenz globaler Lösungen in D ?

$\cap$ Dmax/Cj-lokal $\neq \varnothing$ und Dmax/global = $\varnothing$



Frage: Wäre $\cap$ $D^{max/Cj\text{-}lokal} \neq \varnothing$ NEIN

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

und die Existenz globaler Lösungen in D ?

61

Lokale Konsistenz

Algorithmen für lokale Konsistenz

prüfen notwendige (aber nicht hinreichende) Bedingungen für Existenz globaler Lösungen.

Beispiel:

Lokale Konsistenz, aber keine globale Lösung

$V=\{v_1,v_2,v_3\}$ $Dom(v_1)=Dom(v_2)=Dom(v_3)=\{a,b\}$

Constraints:

Unterschiedliche Variablen mit unterschiedlichen Werten belegen.

$C_{ij} = \{ [a,b],[b,a] \} \subseteq Dom(v_i)\times Dom(v_j)$ für $i\neq j$

Schema: Constraint-Propagierung

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

$$D_i^I \subseteq ... \subseteq D_i^5 \subseteq D_i^4 \subseteq D_i^3 \subseteq D_i^2 \subseteq D_i^1 \subseteq Dom(v_i)$$

Resultat: Einschränkung des Suchraums.

Suche in $D^I := D_1^I \times ... \times D_n^I$ nach Lösung

Schema: Constraint-Propagierung

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

   $s := 1$.

Wahl-Möglichkeiten

2. **Wähle** ein (aussichtsreiches) $C(s) \in \mathbf{C}$.
3. **Wähle** Einschränkung $D(s) \subseteq D(s-1)$, so dass $D(s)$ lokal kons. 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 zusätzliche Suche).

(b) Es ist keine weitere Einschränkung zu erwarten,
(kein weiteres $C(s) \in \mathbf{C}$ mit $D(s) \neq D(s-1)$ gefunden)

(c) D(s) ist global inkonsistent
($D(s) = \varnothing$ bzw. gemäß weiterer Suche).

Schema: Constraint-Propagierung

Möglichkeiten zur Weiterarbeit:

- Abbruch mit gefundener Lösung bei 4a)
- Weiter bei 2. mit s:=s+1 (falls keiner der drei Fälle zutrifft).
- Backtracking bei 4b) oder 4c)
  (alternative Möglichkeiten unter 1.,2.,3. versuchen)
- Suche nach Lösung in D(s)
- Abbruch, wenn Backtracking nicht aussichtsreich.

Schema: Constraint-Propagierung

Es ergibt sich Folge von Einschränkungen

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

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

Satz:

Voraussetzungen:

- D(0) = Dom(V)
- D(s) jeweils maximale lokale Einschränkung von D(s-1) bezüglich des betrachteten Constraints

Behauptungen:

- D(s) enthält stets alle globalen Lösungen (kein Backtracking nötig).
- Falls ein $D(s) = \varnothing$, so existiert keine Lösung.

Schema: Constraint-Propagierung

Kein eigentlicher Algorithmus (prinzipielle Unlösbarkeit).

Wahl-Möglichkeiten und Suchverfahren einbezogen.

Probleme:
- Vollständigkeit (evtl. gehen Lösungen verloren).
- Evtl. unendliche Folge von Einschränkungen.

Beispiele

Keine Verbesserung bei maximalen Einschränkungen:

$V=\{v_1,v_2,v_3\}$
Constraints:
Unterschiedliche Variablen mit unterschiedlichen Werten belegen: $C_{ij} = \{ [a,b],[b,a] \} \subseteq Dom(v_i)\times Dom(v_j)$ für $i\neq j$

a) $Dom(v_1)=Dom(v_2)=Dom(v_3)=\{a,b\}$
b) $Dom(v_1)=Dom(v_2)=Dom(v_3)=\{a,b,c\}$
In beiden Fällen stets $D_i^{max/Cj\text{-}lokal} = Dom(v_i)$

Globale Lösung existiert bei b) aber nicht bei a)

Beispiele

Unendliche Folge bei maximalen Einschränkungen:

$V=\{v_1,v_2\}$, $Dom(v_1)=Dom(v_2)= [0,1]$ (Intervall)

Constraints:

$C_1 = \{ [a,b] / \quad a = b \} \subseteq Dom(v_1)\times Dom(v_2)$

$C_2 = \{ [a,b] / \quad a = 2b \} \subseteq Dom(v_1)\times Dom(v_2)$

Initial: $D(0) = [0,1] \times [0,1]$

mit C_2 : $D(1) = [0,1] \times [0, 0.5]$

mit C_1 : $D(2) = [0,0.5] \times [0, 0.5]$

usw.

Globale Lösung: [0,0]

Bildverarbeitung als „schlecht gestelltes Problem“?

Nicht allgemein ...

Bilder enthalten viel (auch redundante) Information

Pose einer Kamera (Ort und Richtung: 6 Parameter)
i.a. bestimmt durch Abbildung von 3 bekannten Punkten

Information aus Bildern als Constraints

Durch geometrische Beziehungen miteinander verknüpft:

- Positionen von Objekten in der Realität
- Positionen von Objekten im Bild
- Relationen zwischen Objekten (Distanz, Verdeckung…)
- Kamerapose (z.B. als Ort/Richtung des Brennpunktes)

Constraints für Positions-Parameter
Parameterwerte bestimmen: CSP mit vielen Redundanzen

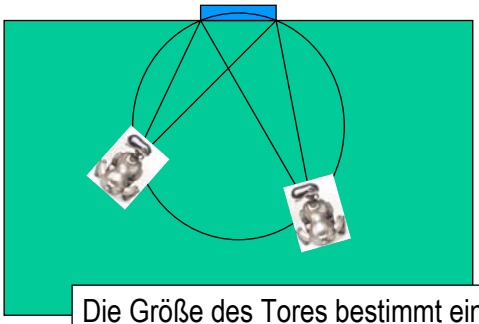
Weitere Constraints:

- Wertebereiche (z.B. für Ort der Kamera eines Roboters)
- Frühere Positionen von Objekten
- Geschwindigkeiten usw.

Zusammenhänge nutzen (Geometrie)

Aufgabe: Wo auf dem Spielfeld (Weltkoordinaten) befinden sich der Roboter (Beobachter) und der Ball?

Zusammenhänge nutzen (Geometrie)

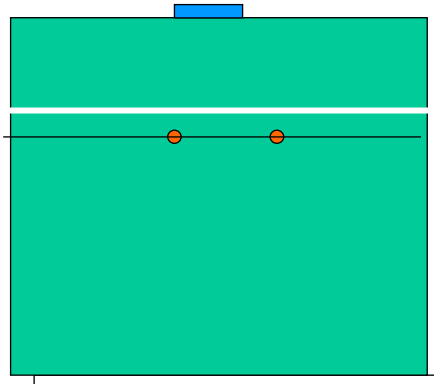


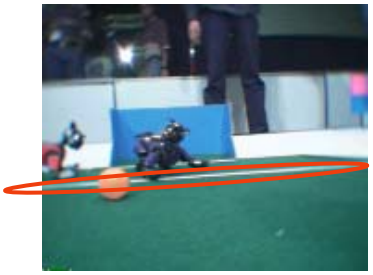
Die Größe des Tores bestimmt einen Kreis für mögliche Positionen des Beobachters.

Zusammenhänge nutzen (Geometrie)

Die Größe des Balles bestimmt einen Kreis für mögliche Positionen des Balles relativ zum Beobachter.

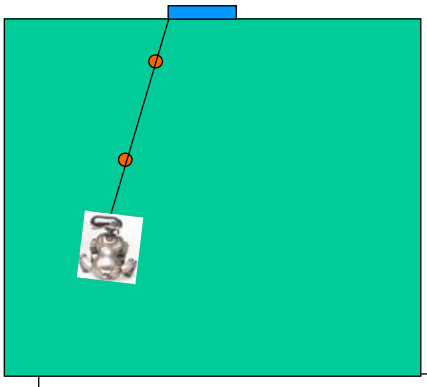
Zusammenhänge nutzen (Geometrie)





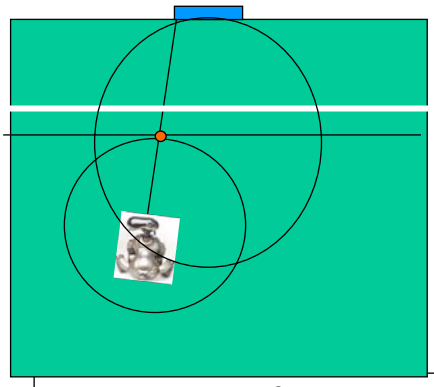
Der Ball liegt auf einer Linie vor der Strafraumgrenze

Zusammenhänge nutzen (Geometrie)



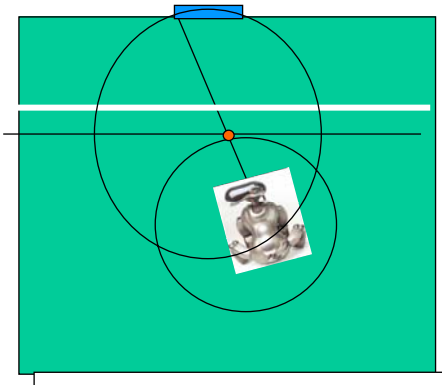
Der Ball liegt auf einer Linie zwischen mir und dem Torpfosten

Zusammenhänge nutzen (Geometrie)



Kombination ergibt 2 mögliche Positionen für Roboter und Ball

Zusammenhänge nutzen (Geometrie)

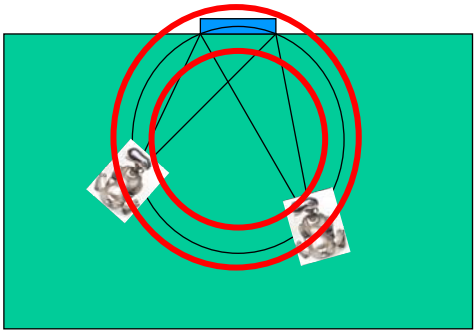


Kombination ergibt 2 mögliche Positionen für Roboter und Ball

Information aus Bildern als Constraints

Behandlung von ungenauer Information:

Constraints erweitern um Toleranz-Bereiche



Constraint-Propagation im Allen-Kalkül

Allen-Kalkül: Modell für **Zeitliches Schließen** auf der Basis von Zeit-Intervallen

Übertragung auch auf räumliches Schließen

Andere Modelle für zeitliche Abläufe:

- Situationenkalkül
- Temporale 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

Ausdrucksmöglichkeiten für

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

Allen-Relationen für Zeitintervalle

Syntax:

– Zeichen 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,{ ⊆ , < }] , mit

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

Präzedenz:
Ein Zeitintervall liegt *vollständig vor* einem anderem.
Ein trennendes Zwischenintervall ist nicht erforderlich.

Bedeutungen (Formalismus in Metasprache!)

STARTS(x,y)

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

FINISHES(x,y)

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

DURING(x,y)

d(x,y) gdw. $x \subseteq y \wedge \exists z (z \subseteq y \wedge x < z) \wedge \exists z (z \subseteq y \wedge z < x)$

BEFORE(x,y)

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

OVERLAPS(x,y)

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

MEETS(x,y)

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

EQUALS(x,y)

e(x,y) gdw. $(x \subseteq y \wedge y \subseteq x)$

Bedeutungen

STARTS(x,y)

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

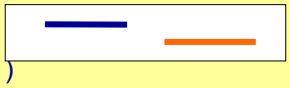
FINISHES(x,y)

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

DURING(x,y)

d(x,y) gdw. $x \subseteq y \wedge \exists z (z \subseteq y \wedge x < z) \wedge \exists z (z \subseteq y \wedge z < x)$

BEFORE(x,y)

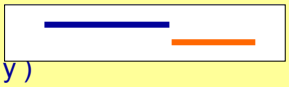


b(x,y) gdw. $x < y \wedge \exists z (x < z \wedge z < y)$

OVERLAPS(x,y)

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

MEETS(x,y)



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

EQUALS(x,y)

e(x,y) gdw. $(x \subseteq y \wedge y \subseteq x)$

Bedeutungen (Inverse Relationen)

Inverse von STARTS(x,y)
$s_i(x,y)$ gdw. s(y,x)

Inverse von FINISHES(x,y)
$f_i(x,y)$ gdw. f(y,x)

Inverse von DURING(x,y)
$d_i(x,y)$ gdw. d(y,x)

Inverse von BEFORE(x,y)
$b_i(x,y)$ gdw. b(y,x)

Inverse von OVERLAPS(x,y)
$o_i(x,y)$ gdw. o(y,x)

Inverse von MEETS(x,y)
$m_i(x,y)$ gdw. 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) :

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

- 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.

Reduktion ist möglich auf MEETS:

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

Zeitliche Inferenzen

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 beginnt während B

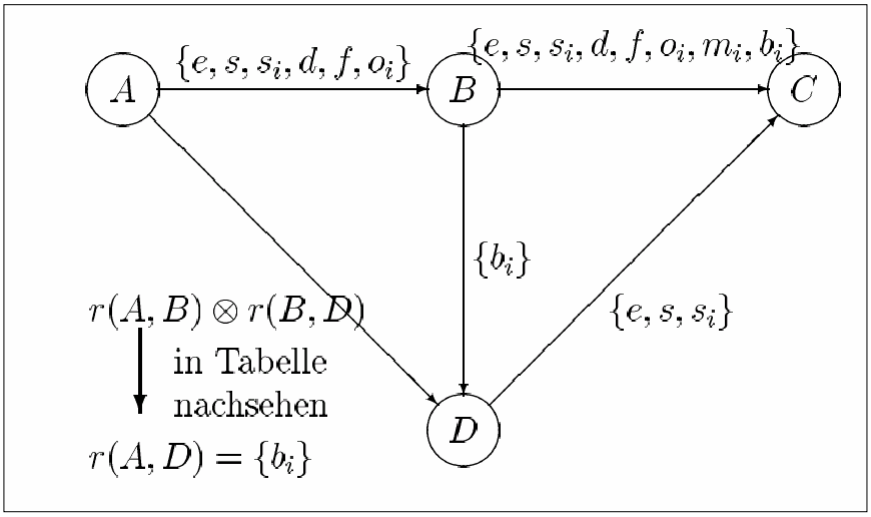
B beginnt nicht vor C

B liegt zeitlich völlig nach D

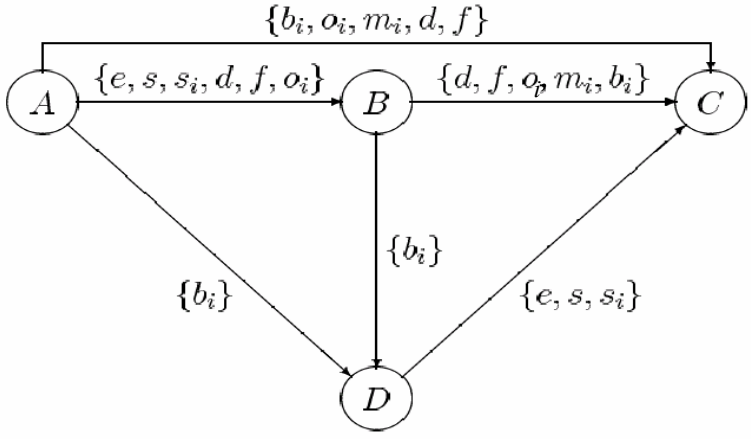
C und D beginnen gleichzeitig

- Welche Beziehungen gelten zwischen A und D?
- 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?

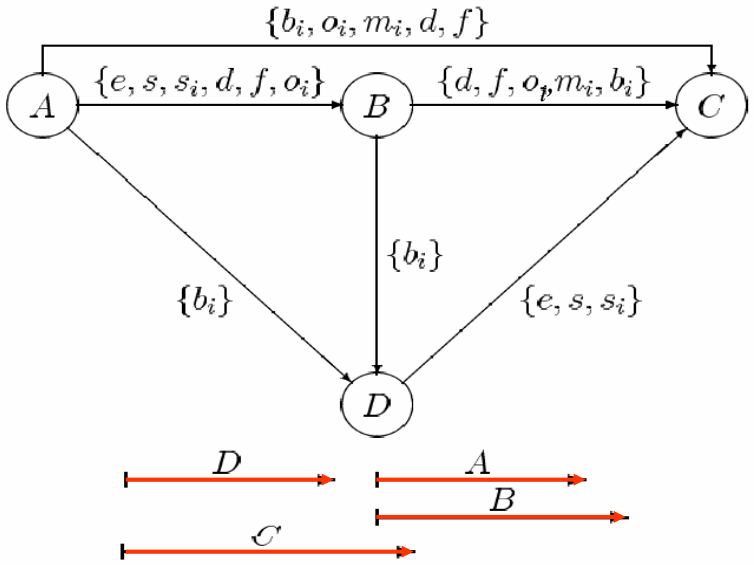
Darstellung als Graph („Allen-Netz“)



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.)

  (beschriftete Kanten)
- 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 Beziehungen zwischen den Intervallen t_1 und t_2 an.

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: $r(\beta(t_1), \beta(t_2)) \in c(t_1, t_2)$

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

(r ist eindeutig aufgrund der Axiome)

Globale Konsistenz:

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

(Beschreibung realistischer Zeitverläufe)

Andernfalls: *global inkonsistent*.

Betrachtung als binäres CSP

Wir werden aber anderes Lösungsverfahren anwenden

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 (siehe Tabelle) :

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

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.

Prüfung auf lokale Konsistenz

Der Algorithmus benutzt

- Stack K für die zu prüfenden Intervallpaare
- Abbildung $R : T \times T \rightarrow 2^{\{s,si,\ldots,e\}}$
  der aktuellen Beschriftungen.

Initialisierung:

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

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

Prüfung auf lokale Konsistenz

Äußerer Zyklus:

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

sonst: für $(t_1,t_2) := pop(K)$ 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).

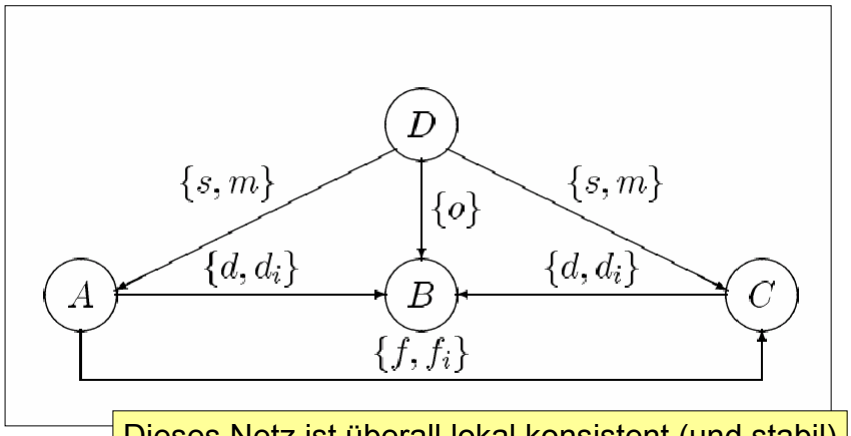
$O(n^2)$ für äußere Schleife:

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.

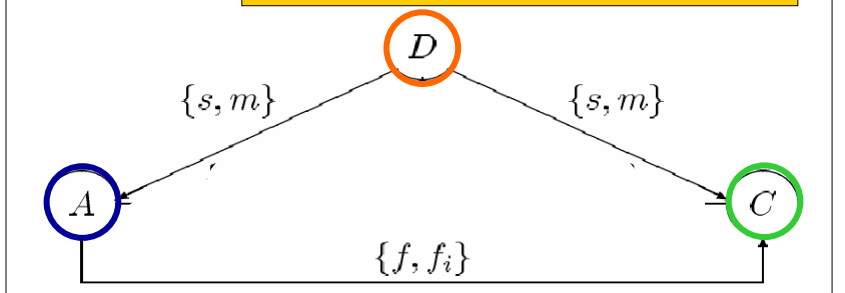
Lokale vs. Globale Konsistenz



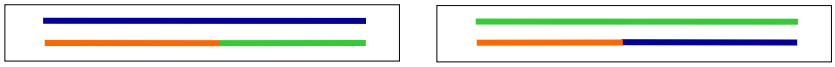
Dieses Netz ist überall lokal konsistent (und stabil)

Lokale vs. Globale Konsistenz

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



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



Anwendung des Verfahrens

- 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?

Suchraum als Parameterraum.

Zusammenhänge im Suchraum explizit machen:
Einschränkende Bedingungen für Lösungen.
Ziel: Suchraum einschränken (keine Suche in Bereichen, die keine Lösung enthalten können).

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

Lösungsraum einschränken

Zusätzliche Constraints schränken Lösungsmenge ein.

Besonders bequem:

Einschränkungen der Definitions-Bereiche.

Constraint-Propagation:

- Definitions-Bereiche sukzessive einschränken.
- Ergebnis ist ein kleinerer Suchraum oder der Nachweis der Inkonsistenz.
- Abschließende Suche kann auch noch Inkonsistenz ergeben.

CSP: Zerlegungen des Problemraums

Und-Zerlegung (= Problemzerlegung, separierte CSP)

Zerlegung der Variablenmenge

Probleme einzeln lösen

Gesamtergebnis aus Einzel-Ergebnissen zusammengesetzt

Beispiele:

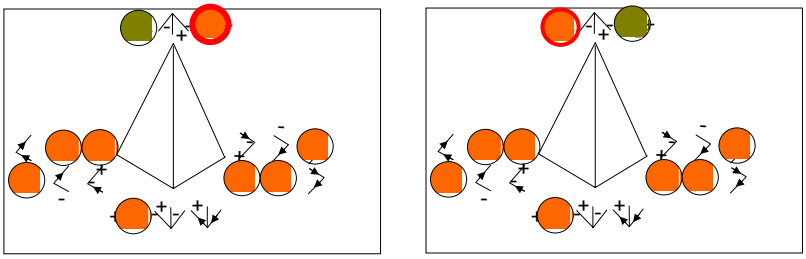
- Bildverarbeitung: nicht zusammenhängende Objekte
- Allen-Netze: Nicht verknüpfte zeitliche Angaben
- Färbungsproblem für Inseln

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-Zerlegung.

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

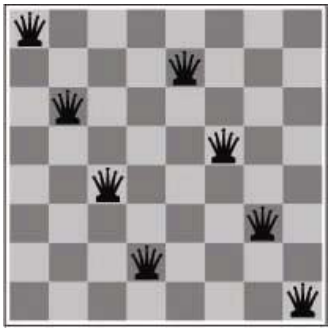
Start mit einer (Erfolg versprechenden) Belegung
Sukzessive Werte einer Variablen ändern,
so dass Zahl der Constraint-Verletzungen sinkt
(Heuristik für Suche)

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:

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

Fallstudie: 8-Damen-Problem

Formulierung S2:

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

Ausgangszustand: leeres Brett

Zielzustand: 8 Damen auf dem Brett, keine angegriffen

Zustandsübergang:

Wähle die am weitesten links liegende freie Spalte

Setze eine Dame auf ein nicht angegriffenes Feld in dieser Spalte

Komplexität: 2057

Fallstudie: 8-Damen-Problem

Behandlung als heuristisches Suchproblem

Formulierung S3 (Bergsteigen, „greedy search“):

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

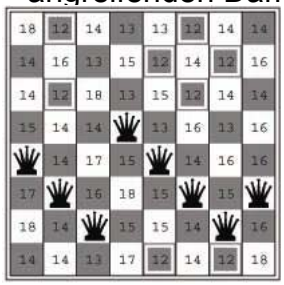
Anfangszustand: Beliebig aus dieser Menge

Zielzustand: 8 Damen auf dem Brett, keine angegriffen

Zustandsübergang: Eine Dame in ihrer Spalte verschieben, bei der die Anzahl der sich direkt oder indirekt angreifenden Damenpaare am meisten verringert wird (indirekter Angriff: Es steht eine Figur dazwischen).

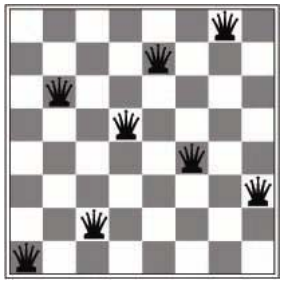
Fallstudie: 8-Damen-Problem

Heuristik h: Anzahl der sich direkt oder indirekt angreifenden Damenpaare



Angabe der h-Werte bei Verschieben der Dame innerhalb der Spalte auf das betreffende Feld

Lokales Minimum (h=1)



Fallstudie: 8-Damen-Problem

Anwendung genetischer Algorithmen

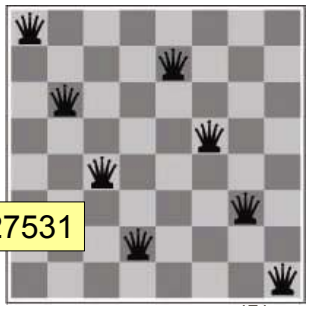
Formulierung G

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

Fitness:

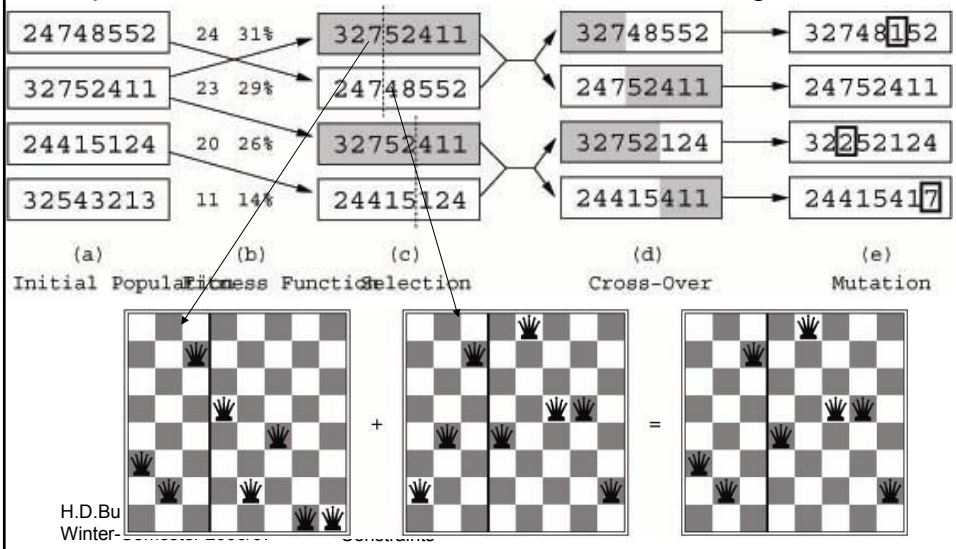
Paare von Damen, die sich nicht gegenseitig angreifen
(f=28 für Lösung)

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



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 $[v_k,v_l]$, $1 \le k < l \le 8$:
C_{kl} = „Dame in Spalte k greift Dame in Spalte l nicht an“

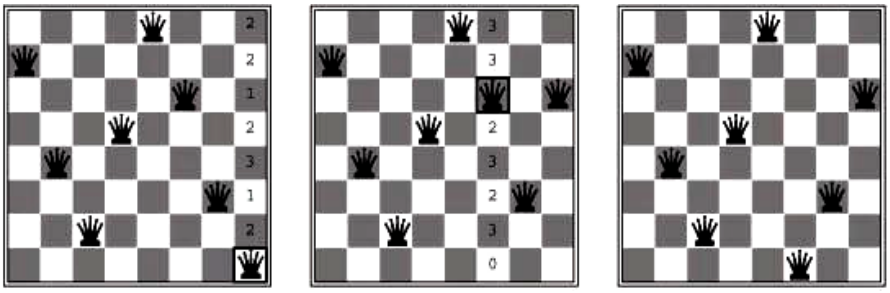
Fallstudie: 8-Damen-Problem

Behandlung als Constraint-Problem mit lokaler Suche
Formulierung CL:

Variable $v_1,...,v_8$ 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 $[v_k,v_l]$, $1 \le k < l \le 8$:
C_{kl} = „Dame in Spalte k greift Dame in Spalte l nicht an“
Zustandsraum: Variablenbelegungen
Ausgangszustand: Eine Variablenbelegung
(zufällig, aber „günstig“ gewählt)
Zustands-Übergang:
- Wähle zufällig eine Variable mit Konflikt.
- Ändere ihren Wert so, dass Konflikte minimiert werden.

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!