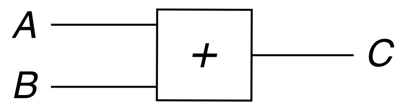
Kapitel MK:IV

IV. Modellieren mit Constraints

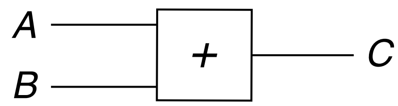
Algebraische Constraints

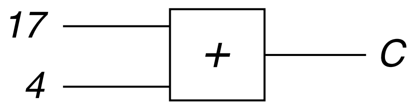
Beispiel Addierer $A + B = C$:

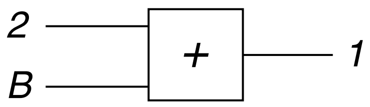


Algebraische Constraints

Beispiel Addierer $A + B = C$:

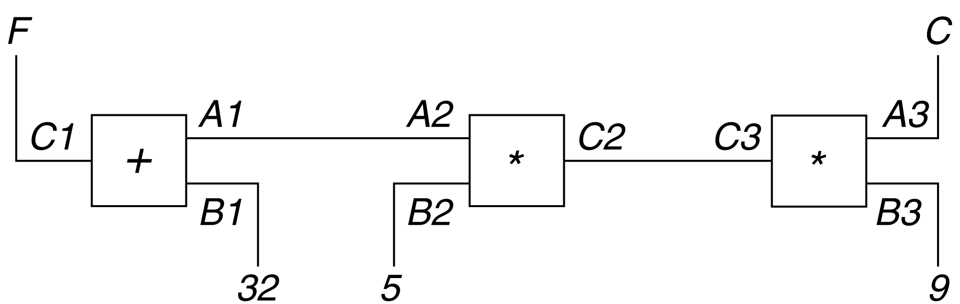






Beispiel Umrechnung zwischen Fahrenheit (F) und Celsius (C):

$$C = (((F - 32) * 5)/9)$$



Algebraische Constraints

Definition 19 (lokal bestimmt (locally determined))

Sei $x_1, x_2, \ldots, x_n \in X$ eine Menge von Variablen mit den Wertebereichen $D_1, D_2, \ldots, D_n$. Sei $C(x_1, x_2, \ldots, x_n)$ ein über X definierter n-stelliger Constraint. Dann heißt eine Variable $x_i \in X$ lokal bestimmt mit Wert d, falls für jede erfüllende Belegung $(d_1, \ldots, d_n)$ von C gilt: $d_i = d$.

Beispiel:

$$X = \{x, y\}, \quad \mathcal{C} = \{C_1, C_2\}$$

$$C_1(x, y) : (2 - x) \cdot \max\{1, y\} = 0$$

$$C_2(x, y) : x + y = 4$$

x ist durch C_1 bestimmt mit Wert 2.

Algebraische Constraints

Definition 20 (Einschrittableitung)

Sei C ein Constraint, definiert auf den Variablen $x_1, \ldots, x_n$, und sei x_i lokal bestimmt mit Wert d. Dann heißt die Zuweisung von d zu x_i ($X := d$) Einschrittableitung.

Algorithm: simpleLocalPropagation

Input: $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$. Domains of the n constraint variables.
$\mathcal{C}$. Constraint net.

Output: $\{(x \mapsto d) \mid x \in x, d \in D_i\}$.

simpleLocalPropagation($\mathcal{D}$,$\mathcal{C}$)

1. WHILE $\exists$**x** $\in$ X $\exists$C $\in \mathcal{C}$: **x** locally determined with **d** DO
2. **x** := **d**

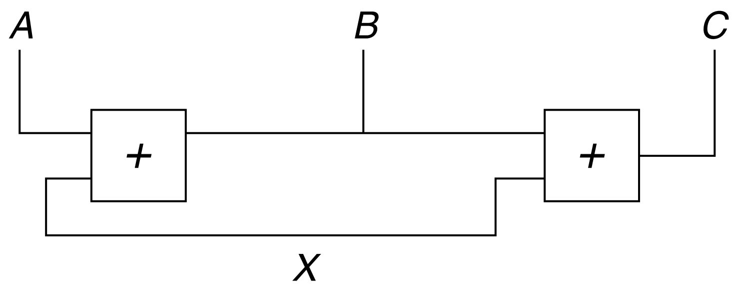
Bemerkungen:

- Algebraische Constraints sind Constraints auf nicht-endlichen Wertebereichen.
- Die Einschrittableitung bzw. lokale Propagierung ist mit dem Forward-Chaining in der Regelverarbeitung vergleichbar.

Algebraische Constraints

Grenzen lokaler Wertepropagierung I

Eine existierende Lösung kann nicht immer mit einem lokalen Verfahren bestimmt werden. Beispiel:



Gegeben A, B.

➜ C kann abgeleitet werden.

Gegeben A, C.

➜ B ist bestimmt, aber nicht durch lokale Propagierung ableitbar.

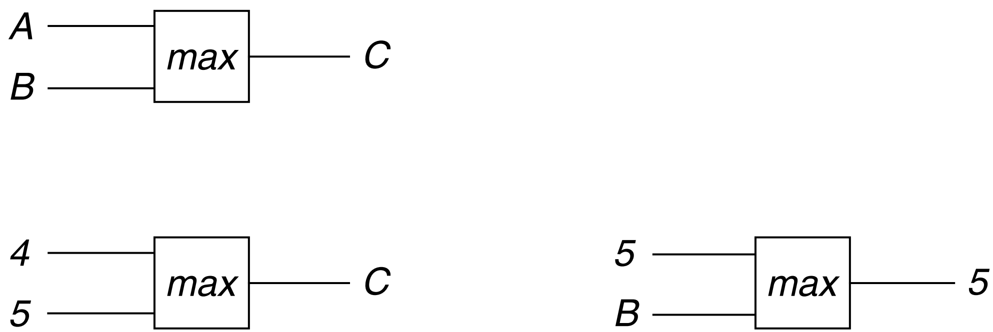
$A + x = B$

$B + x = C$

Algebraische Constraints

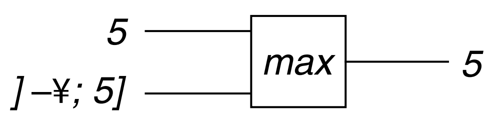
Grenzen lokaler Wertepropagierung II

Eine existierende Lösung kann nicht immer mit einem lokalen Verfahren bestimmt werden. Beispiel:



Löungsansätze:

- Bei algebraischen Problemen evtl. Intervallpropagierung:



- Bei endlichen Wertbereichen Wertauswahl plus Backtracking.

Algebraische Constraints

Constraint-Löser „gaussElimination“

Algorithm: gaussElimination

Input: $\mathcal{C} = \mathtt{C}_1, \ldots, \mathtt{C}_\mathtt{m}$ Constraint net with linear equations.

Output: TRUE, if $\mathcal{C}$ is satisfiable, FALSE otherwise.

gaussElimination($\mathcal{C}$)

1. result := TRUE
2. REPEAT
3. Choose $\mathtt{x} \in \mathtt{X}$ $\mathtt{C} \in \mathcal{C}$: $\mathtt{x}$ occurs in $\mathtt{C}$
4. Isolate $\mathtt{x}$ in $\mathtt{C}$
5. Substitute $\mathtt{C}$ for $\mathtt{x}$ in $\mathcal{C}$
6. IF $\mathcal{C}$ widerspruchsvoll THEN result := FALSE
7. UNTIL $|\mathcal{C}| = 1$ $\vee$ result = FALSE
8. RETURN(result)

Algebraische Constraints

Intervallpropagierung

Den Variablen eines Constraint-Netzes werden Intervalle zugeordnet. Beispiel:

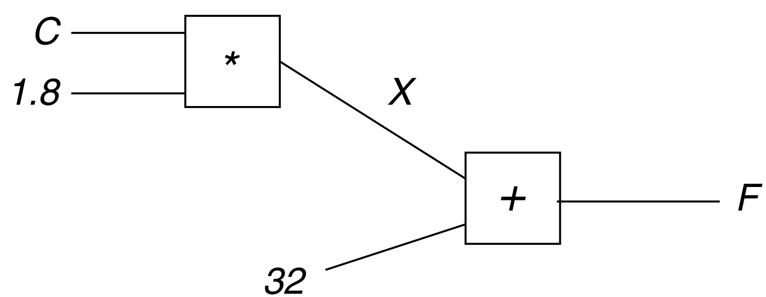
x hat das Intervall $[-1; 7]$

Semantik: $x \geq -1 \ \wedge \ x \leq 7$

Einsatzbereiche:

1. Behandlung unscharfer Information; Wissen über Parameter ist nur in Form von Werteintervallen bekannt. Beispiel:

$$F = C \cdot 1.8 + 32, \quad 1 \leq C \leq 5 \quad \Rightarrow \quad 27 \leq F \leq 35$$



2. Geschlossenen Darstellung aller Lösungen, falls unendlich viele existieren.

Algebraische Constraints

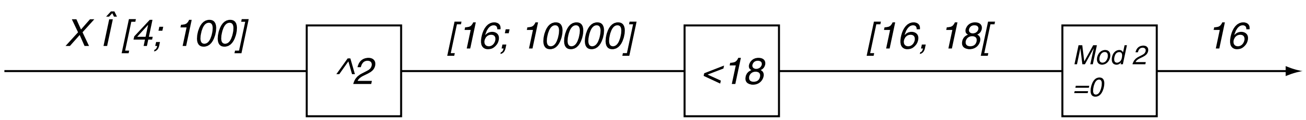
Intervallpropagierung

Einsatzbereiche (Fortsetzung):

3. Behandlung unterbestimmter Systeme. Hier existiert mehr als eine Lösung ➜ Lösungsintervalle.

   Beispiel: Optimaler Drehzahlbereich eines Motors sei $[3000, 4000]$.

4. Gleichzeitige Einschränkung unendlich vieler Werte statt Generate-and-Test auf Basis einzelner Werte.



Kapitel MK:IV

IV. Modellieren mit Constraints

Intervall Constraints

Intervall Constraint Satisfaction Problem (I-CSP)

Definition 21 (I-CSP, Toleranzsituation [Hyvönen 1992]**)**

Sei X eine Menge von Variablen mit den Wertebereichen $\mathcal{D}$, und sei $\mathcal{C}$ ein über X definiertes Constraint-Netz (vgl. Constraint-Definition).

Darf den Variablen x_i ein geschlossenes reelles Intervall I_i, $I_i \in D_i$ zugewiesen werden (in Zeichen: $x_i := I_i$), dann bezeichnen wir ein so definiertes Constraint-Problem als Intervall-CSP bzw. I-CSP.

Eine Zuordnung eines Intervalls zu jeder Variable $x \in X$ wird als Toleranzsituation bezeichnet. Eine Toleranzsituation definiert folgende Menge von Ausprägungen:

$$\{x_1 = d_1, \ldots, x_n = d_n \mid d_i \in I_i, 1 \leq i \leq n\}$$

Intervall Constraints

Intervall Constraint Satisfaction Problem (I-CSP)

Definition 22 (zulässig (admissible), konsistent [Hyvönen 1992]**)**

Ein I-CSP ist zulässig (admissible) genau dann, wenn seine Toleranzsituation eine Ausprägung mit folgender Eigenschaft besitzt:

$$\exists\{x_1 = d_1 \in I_1, \ldots, x_n = d_n \in I_n\} :$$

„Alle Constraints sind erfüllt“

Eine Variable $x_i := I_i$ ist konsistent genau dann, wenn jede Interpretation $x_j = d, d \in I_i$ wie folgt zu einer Lösung des I-CSP erweitert werden kann:

$$\forall d \in I_i \; \exists\{x_1 = d_1 \in I_1, \ldots, x_i = d, \ldots, x_n = d_n \in I_n\} :$$

„Alle Constraints sind erfüllt“

Ein I-CSP ist konsistent genau dann, wenn jede Variable konsistent ist.

Intervall Constraints

Intervall Constraint Satisfaction Problem (I-CSP)

Definition 23 ($\Rightarrow_S$ [Hyvönen 1992])

Eine Toleranzsituation $S = \{x_1 := I_1, \ldots, x_n := I_n\}$ ist genereller als eine Toleranzsituation $S' = \{x_1 := I'_1, \ldots, x_n := I'_n\}$, in Zeichen: $S' \Rightarrow_S S$, genau dann, wenn gilt: $I'_i \subseteq I_i,\ 1 \leq i \leq n$.

Bemerkungen:

- Global konsistente Variablenbereiche werden auch als Lösung eines I-CSP bezeichnet.
- Mit jeder Intervalleinschränkung verliert eine Toleranzsituation an Allgemeinheit.
- Ziel ist es, eine möglichst eingeschränkte (*Least General Constraint Solution, LGCS*) Toleranzsituation zu finden.
- Die Relation $\Rightarrow_S$ ist reflexiv, antisymetrisch und transitiv. Sie definiert also eine partielle Ordnung auf der Menge aller möglichen Toleranzsituationen eines I-CSP.
- Eine Lösung S ist eine Least General Constraint Solution genau dann, wenn keine Lösung S' existiert, für die $S' \Rightarrow_S S$ gilt.

Intervall Constraints

Lokale Toleranzpropagierung für I-CSP

Lösungsverfahren zur Bestimmung einer Least General Constraint Solution funktionieren im Kern wie lokale Propagierung; statt einzelner Werte werden jedoch Intervalle propagiert.

→ Intervallarithmetik

Intervallarithmetik der einfachen Constraints „$+$“, „$-$“, „$*$“, „$/$“:

1. $[a; b] + [c; d] = [a + c; b + d]$
2. $[a; b] - [c; d] = [a - d; b - c]$
3. $[a; b] * [c; d] = [\min\{ac, ad, bc, bd\}; \max\{ac, ad, bc, bd\}]$
4. $[a; b]/[c; d] = [a, b] * [1/d; 1/c]$, wenn $c, d > 0 \ \vee \ c, d < 0$

Bemerkungen:

- Intervallpropagierung erfordert die Berechnung von Minimum und Maximum der expliziten Constraint-Funktionen.
- Monotonie der expliziten Constraint-Funktionen ist eine nützliche Eigenschaft.
- Bei komplexen Constraints (Stichwort: Nichtmonotonie) kann das Ergebnis einer Intervallpropagierung aus einer Menge von neuen Intervallen bestehen.
  Beispiel: $X^2 = Y, \quad Y := [1; 4]$
  Zwei Lösungen: $X := [1; 2] \ \vee \ X := [-1; -2]$
- Sinnvoll sind auch Funktionen, die sich in wenige, monotone Bereiche aufteilen lassen.

Intervall Constraints

Lokale Toleranzpropagierung für I-CSP

Algorithm: localTolerancePropagation

Input: $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$. Domains of the n constraint variables.
$\mathcal{C}$. Constraint net.

Output: $\mathtt{S} = \{\mathtt{x_1} := \mathtt{I_1}, \ldots, \mathtt{x_n} := \mathtt{I_n}\}$. Tolerance situation.

```
localTolerancePropagation(𝒟,𝒞)

Agenda := functions of the constraints in 𝒞
S := tolerance value assignment for the variables
REPEAT
   FOREACH f ∈ Agenda DO
      I' := f(...)
      IF I' = {}
      THEN return({})
      ELSE IF I ⊆ I'
          THEN remove f from Agenda
          ELSE I := I ∩ I'
UNTIL AGENDA = {}
```