

Grundlagen der Künstlichen Intelligenz

23. Constraint-Satisfaction-Probleme: Pfadkonsistenz

Malte Helmert

Universität Basel

14. April 2014

Grundlagen der Künstlichen Intelligenz

14. April 2014 — 23. Constraint-Satisfaction-Probleme: Pfadkonsistenz

23.1 Jenseits von Kantenkonsistenz

23.2 Pfadkonsistenz

23.3 Zusammenfassung

Constraint-Satisfaction-Probleme: Überblick

Kapitelüberblick Constraint-Satisfaction-Probleme:

- ▶ 19.–20. Einführung
- ▶ 21.–23. Kernalgorithmen
 - ▶ 21. Backtracking
 - ▶ 22. Kantenkonsistenz
 - ▶ 23. Pfadkonsistenz
- ▶ 24.–25. Problemstruktur

23.1 Jenseits von Kantenkonsistenz

Jenseits von Kantenkonsistenz: Pfadkonsistenz

Grundidee der Kantenkonsistenz:

- ▶ zu jeder Belegung einer Variable u muss es eine passende Belegung jeder anderen Variable v geben
- ▶ ansonsten werden Werte von u , die nicht auf v erweiterbar sind, verboten
- ↪ verschärfter **unärer Constraint** auf u

Idee lässt sich auf drei Variablen erweitern (**Pfadkonsistenz**):

- ▶ zu jeder gemeinsamen Belegung von Variablen u, v muss es eine passende Belegung jeder anderen Variablen w geben
- ▶ ansonsten werden Wertepaare für u und v , die nicht auf w erweiterbar sind, verboten
- ↪ verschärfter **binärer Constraint** auf u und v

Jenseits von Kantenkonsistenz: i -Konsistenz

Generelles Konzept der i -Konsistenz für $i \geq 2$:

- ▶ zu jeder gemeinsamen Belegung von $v_1, \dots, v_{i-1}$ muss es eine passende Belegung jeder anderen Variable v_i geben
- ▶ ansonsten werden Wertetupel für $v_1, \dots, v_{i-1}$, die nicht auf v_i erweiterbar sind, verboten
- ↪ verschärfter **$(i-1)$ -stelliger Constraint** auf $v_1, \dots, v_{i-1}$
- ▶ **2-Konsistenz = Kantenkonsistenz**
- ▶ **3-Konsistenz = Pfadkonsistenz (*)**

Wir betrachten allgemeine i -Konsistenz nicht näher, zumal höhere Werte als $i = 3$ selten verwendet werden und wir uns hier auf höchstens binäre Constraints beschränken.

(*) übliche Definitionen von 3-Konsistenz vs. Pfadkonsistenz unterscheiden sich, wenn ternäre (dreistellige) Constraints erlaubt sind

23.2 Pfadkonsistenz

Pfadkonsistenz: Definition

Definition (pfadkonsistent)

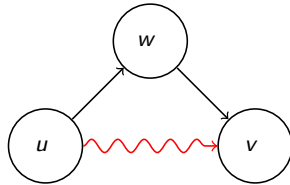
Sei $\mathcal{C} = \langle V, \text{dom}, (R_{uv}) \rangle$ ein Constraint-Netz.

- Zwei verschiedene Variablen $u, v \in V$ sind **pfadkonsistent** in Bezug auf eine dritte Variable $w \in V$, wenn für beliebige Werte $d_u \in \text{dom}(u)$, $d_v \in \text{dom}(v)$ mit $\langle u, v \rangle \in R_{uv}$ immer ein Wert $d_w \in \text{dom}(w)$ mit $\langle d_u, d_w \rangle \in R_{uw}$ und $\langle d_v, d_w \rangle \in R_{vw}$ existiert.
- Das Constraint-Netz $\mathcal{C}$ ist **pfadkonsistent**, wenn für drei verschiedene Variablen u, v, w immer gilt, dass u und v pfadkonsistent in Bezug auf w sind.

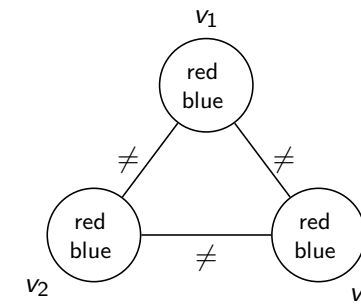
Pfadkonsistenz: Anmerkungen

Anmerkungen:

- ▶ Selbst wenn Constraint R_{uv} trivial ist, kann Pfadkonsistenz nichttriviale Constraints zwischen u und v inferieren.
- ▶ Wenn das Netz **kantenkonsistent** ist, kann Pfadkonsistenz nur dann zu neuer Information führen, wenn sowohl R_{uw} als auch R_{vw} nichttrivial sind.
- ▶ Name „Pfadkonsistenz“:
Pfad $u \rightarrow w \rightarrow v$ führt zu neuer Information über $u \rightarrow v$



Pfadkonsistenz: Beispiel



kantenkonsistent, aber nicht pfadkonsistent

Verarbeitung eines Variablentripels: revise-3

Analog zu **revise** für Kantenkonsistenz:

function revise-3($\mathcal{C}, u, v, w$):

$\langle V, \text{dom}, (R_{uv}) \rangle := \mathcal{C}$

for each $\langle d_u, d_v \rangle \in R_{uv}$:

if there is no $d_w \in \text{dom}(w)$ with

$\langle d_u, d_w \rangle \in R_{uw}$ **and** $\langle d_v, d_w \rangle \in R_{vw}$:

remove $\langle d_u, d_v \rangle$ from R_{uv}

Eingabe: Constraint-Netz $\mathcal{C}$ und drei Variablen u, v, w von $\mathcal{C}$

Effekt: Macht u, v pfadkonsistent in Bezug auf w .

Alle verletzenden Paare werden aus R_{uv} entfernt.

Zeitaufwand: $O(k^3)$, wenn k maximale Wertebereichsgrösse

Herstellen von Pfadkonsistenz: PC-2

Analog zu **AC-3** für Kantenkonsistenz:

function PC-2($\mathcal{C}$):

$\langle V, \text{dom}, (R_{uv}) \rangle := \mathcal{C}$

$queue := \emptyset$

for each set of two variables $\{u, v\}$:

for each $w \in V \setminus \{u, v\}$:

 insert $\langle u, v, w \rangle$ into $queue$

while $queue \neq \emptyset$:

 remove any element $\langle u, v, w \rangle$ from $queue$

 revise-3($\mathcal{C}, u, v, w$)

if R_{uv} changed in the call to revise-3:

for each $w' \in V \setminus \{u, v\}$:

 insert $\langle w', u, v \rangle$ into $queue$

 insert $\langle w', v, u \rangle$ into $queue$

PC-2: Diskussion

Die Aussagen zu AC-3 gelten analog.

- ▶ PC-2 stellt Pfadkonsistenz her
- ▶ **Beweisidee:** Invariante der **while**-Schleife:
Wenn $\langle u, v, w \rangle \notin \text{queue}$, dann u, v pfadkonsistent in Bezug auf w
- ▶ mögliche Optimierung: $\langle u, v, w \rangle$ immer nur dann in *queue* einfügen, wenn R_{uw} und R_{vw} nichttrivial
- ▶ Laufzeit $O(n^3 k^5)$ für n Variablen und maximale Wertebereichsgrösse k (**Warum?**)

23.3 Zusammenfassung

Zusammenfassung

- ▶ Verallgemeinerung von
Kantenkonsistenz (betrachtet **Paare** von Variablen)
auf **Pfadkonsistenz** (betrachtet **Tripel** von Variablen)
und ***i*-Konsistenz** (betrachtet ***i*-Tupel** von Variablen)
- ▶ Kantenkonsistenz verschärft **unäre** Constraints
- ▶ Pfadkonsistenz verschärft **binäre** Constraints
- ▶ *i*-Konsistenz verschärft **(*i* - 1)-stellige** Constraints
- ▶ höhere Konsistenzstufen mächtiger,
aber teurer als Kantenkonsistenz