Hochschule Bielefeld – Campus Minden

IFM 3.2: Grundlagen der Künstlichen Intelligenz

Übungsblatt: **Constraints**

Name: Aurelius Pilat

Wintersemester 2025/26

Dozent: Prof. Dr. Carsten Gips

CSP.01: Logikrätsel als CSP

Wir betrachten die Variante des Einstein-Rätsels mit 5 Häusern in einer Reihe. Jedes Haus hat genau eine Farbe, einen Bewohner (Nationalität), ein Getränk, eine Zigarettenmarke und ein Haustier.

Variablen und Wertebereiche

Wir modellieren jede mögliche Ausprägung als Variable, deren Wert die Hausnummer $1, \dots, 5$ ist.

- Hauspositionen: $H = \{1, 2, 3, 4, 5\}$.
- Farben: `red, green, white, yellow, blue`.
  Variablen: $C_{\text{red}}, C_{\text{green}}, C_{\text{white}}, C_{\text{yellow}}, C_{\text{blue}} \in H$.
- Nationalitäten: `Brit, Swede, Dane, Norwegian, German`.
  Variablen: $N_{\text{brit}}, N_{\text{swede}}, N_{\text{dane}}, N_{\text{norwegian}}, N_{\text{german}} \in H$.
- Getränke: `tea, coffee, milk, beer, water`.
  Variablen: $D_{\text{tea}}, D_{\text{coffee}}, D_{\text{milk}}, D_{\text{beer}}, D_{\text{water}} \in H$.
- Zigaretten: `PallMall, Dunhill, Blend, BlueMaster, Prince`.
  Variablen: $S_{\text{pall}}, S_{\text{dunhill}}, S_{\text{blend}}, S_{\text{blue}}, S_{\text{prince}} \in H$.
- Haustiere: `dogs, birds, cats, horses, fish`.
  Variablen: $P_{\text{dogs}}, P_{\text{birds}}, P_{\text{cats}}, P_{\text{horses}}, P_{\text{fish}} \in H$.

Allgemeine Constraints (All-different)

In jeder Kategorie muss jede Ausprägung in einem anderen Haus liegen. Das modellieren wir als binäre Ungleichheits-Constraints:

- Für alle verschiedenen Farben $c \neq c'$ gilt: $C_c \neq C_{c'}$.
- Für alle verschiedenen Nationalitäten $n \neq n'$ gilt: $N_n \neq N_{n'}$.
- Entsprechend für Getränke, Zigarettenmarken, Haustiere:

  $$D_d \neq D_{d'}, \quad S_s \neq S_{s'}, \quad P_p \neq P_{p'} \quad \text{für } d \neq d',\ s \neq s',\ p \neq p'.$$

  Dies sind jeweils binäre Constraints der Form $((X, Y), \{\, (x, y) \mid x \neq y \,\})$.

Constraints aus den Hinweisen (Auswahl)

Wir übernehmen die (klassischen) Hinweise aus der Wikipedia-Variante und formulieren sie als (unäre oder binäre) Relationen:

1. *Der Brite lebt im roten Haus.*

$$N_{\text{brit}} = C_{\text{red}}$$

(binärer Gleichheits-Constraint zwischen Nationalität und Farbe).

2. *Der Schwede hält Hunde.*

$$N_{\text{swede}} = P_{\text{dogs}}$$

3. *Der Däne trinkt Tee.*

$$N_{\text{dane}} = D_{\text{tea}}$$

4. *Das grüne Haus steht links vom weißen Haus (direkt daneben).*

$$C_{\text{green}} + 1 = C_{\text{white}}$$

5. *Der Besitzer des grünen Hauses trinkt Kaffee.*

$$C_{\text{green}} = D_{\text{coffee}}$$

6. *Der, der Pall Mall raucht, hält Vögel.*

$$S_{\text{pall}} = P_{\text{birds}}$$

7. *Der Besitzer des gelben Hauses raucht Dunhill.*

$$C_{\text{yellow}} = S_{\text{dunhill}}$$

8. *Im mittleren Haus wird Milch getrunken.*

$$D_{\text{milk}} = 3$$

(unärer Constraint auf D_{milk}).

9. *Der Norweger wohnt im ersten Haus.*

$$N_{\text{norwegian}} = 1$$

10. *Der, der Blend raucht, wohnt neben dem, der Katzen hält.*

$$|S_{\text{blend}} - P_{\text{cats}}| = 1$$

11. *Der, der Pferde hält, wohnt neben dem, der Dunhill raucht.*

$$|P_{\text{horses}} - S_{\text{dunhill}}| = 1$$

12. *Der, der Blue Master raucht, trinkt Bier.*

$$S_{\text{blue}} = D_{\text{beer}}$$

13. *Der Deutsche raucht Prince.*

$$N_{\text{german}} = S_{\text{prince}}$$

14. *Der Norweger wohnt neben dem blauen Haus.*

$$|N_{\text{norwegian}} - C_{\text{blue}}| = 1$$

15. *Der, der Blend raucht, hat einen Nachbarn, der Wasser trinkt.*

$$|S_{\text{blend}} - D_{\text{water}}| = 1$$

Alle Gleichheits- und Ungleichheitsbedingungen sind binäre Constraints, die jeweils als Menge erlaubter Paare (x, y) aufgefasst werden können.

CSP.02: Framework für Constraint Satisfaction

Wir verwenden für das Einstein-Rätsel das BT_Search-Framework mit unterschiedlichen Erweiterungen.

(1) Basis-Algorithmus `BT_Search`

Der Basisalgorithmus arbeitet wie folgt (Pseudocode):

```
BT_Search(assignment, csp):
    if complete(assignment): return assignment

    var = select_unassigned_variable(csp, assignment)
    for val in domain[var]:
        if is_consistent(var, val, assignment, csp):
            assignment[var] = val
            result = BT_Search(assignment, csp)
            if result != failure:
                return result
```

```
        remove var from assignment

    return failure
```

Mit diesem Algorithmus findet man für das Einstein-Rätsel eine eindeutige Lösung. In üblicher Darstellung (Hausnummern 1 bis 5 von links nach rechts):

- Haus 1: Norweger, gelb, Wasser, Dunhill, Katzen
- Haus 2: Däne, blau, Tee, Blend, Pferde
- Haus 3: Brite, rot, Milch, Pall Mall, Vögel
- Haus 4: Deutscher, grün, Kaffee, Prince, Fische
- Haus 5: Schwede, weiß, Bier, Blue Master, Hunde

(2) Erweiterung um MRV und Gradheuristik

Wir ersetzen in `BT_Search` die Wahl der Variable durch:

1. MRV (Minimum Remaining Values): wähle die Variable mit der kleinsten aktuellen Domänengröße.
2. Bei Gleichstand: Gradheuristik (degree heuristic): wähle die Variable mit den meisten Constraints zu noch nicht belegten Variablen.

Diese Heuristiken reduzieren den Suchbaum deutlich: Es werden weniger Backtracking-Schritte benötigt, obwohl das Ergebnis (die Lösung) gleich bleibt.

(3) Vorverarbeitung mit AC-3 (Kantenkonsistenz)

Vor dem Start von `BT_Search` wenden wir AC-3 auf den CSP an und verkürzen so die Domänen:

- Einige Werte in den Domänen werden sofort entfernt, da sie keine Unterstützung mehr besitzen.
- In diesem Problem allein reicht AC-3 noch nicht zur vollständigen Lösung, aber die Suche wird weiter beschleunigt.
- Anschließend ruft man `BT_Search` mit MRV+Gradheuristik auf den reduzierten Domänen auf.

(4) Min-Conflicts

Alternativ kann die Min-Conflicts-Heuristik verwendet werden:

```
MinConflicts(csp, maxSteps):
    assignment = random_complete_assignment(csp)
    for step in 1..maxSteps:
        if assignment ist vollständig konsistent:
            return assignment
        var = zufällige konfliktbehaftete Variable
        value = Wert in Domain[var] mit minimalen Konflikten
        assignment[var] = value
    return failure
```

In der Praxis findet Min-Conflicts für das Einstein-Rätsel sehr schnell eine konsistente Belegung. Der Algorithmus ist jedoch unvollständig (keine Terminierungsgarantie), funktioniert aber für dieses Problem sehr zuverlässig.

Vergleich (Ergebnis und Laufzeit, qualitativ)

- Alle Varianten liefern dieselbe Lösung (siehe oben).
- Reine Backtracking-Suche benötigt die meisten Rekursionsschritte (viel Backtracking).
- BT mit MRV+Gradheuristik reduziert die Zahl der Versuche deutlich.
- AC-3 vorab plus BT mit Heuristiken ist nochmals effizienter.
- Min-Conflicts findet mit hoher Wahrscheinlichkeit in wenigen Iterationen eine Lösung, benötigt aber eine Zufallskomponente und besitzt keine Vollständigkeitsgarantie.

CSP.03: Kantenkonsistenz mit AC-3

Gegeben sei $D = \{0, 1, 2, 3, 4, 5\}$ und

$$\langle \{v_1, v_2, v_3, v_4\}, \{D_{v_1} = D_{v_2} = D_{v_3} = D_{v_4} = D\}, \{c_1, c_2, c_3, c_4\} \rangle$$

mit

$$\begin{aligned} c_1 &= ((v_1, v_2), \{(x, y) \in D^2 \mid x + y = 3\}), \\ c_2 &= ((v_2, v_3), \{(x, y) \in D^2 \mid x + y \leq 3\}), \\ c_3 &= ((v_1, v_3), \{(x, y) \in D^2 \mid x \leq y\}), \\ c_4 &= ((v_3, v_4), \{(x, y) \in D^2 \mid x \neq y\}). \end{aligned}$$

(1) Constraint-Graph

Knoten: v_1, v_2, v_3, v_4.

Kanten (ungerichtet):

$$v_1 - v_2 \quad (c_1), \qquad v_2 - v_3 \quad (c_2), \qquad v_1 - v_3 \quad (c_3), \qquad v_3 - v_4 \quad (c_4).$$

(2) AC-3: Queue und Domänenentwicklung

Wir betrachten gerichtete Bögen. Anfangszustand:

$$D_{v_1} = D_{v_2} = D_{v_3} = D_{v_4} = \{0, 1, 2, 3, 4, 5\}.$$

Initiale Queue (alle gerichteten Kanten):

$$Q_0 = \big[(v_1, v_2), (v_2, v_1), (v_2, v_3), (v_3, v_2), (v_1, v_3), (v_3, v_1), (v_3, v_4), (v_4, v_3)\big].$$

Wir skizzieren die relevanten Iterationen, d. h. nur dort, wo sich Domänen ändern.

Iteration 1: Bogen (v_1, v_2) Constraint $x + y = 3$.

Für $x \in D_{v_1}$ muss es $y \in D_{v_2}$ mit $x + y = 3$ geben.

Werteprüfung:

$$x = 0 \Rightarrow y = 3, \quad x = 1 \Rightarrow y = 2, \quad x = 2 \Rightarrow y = 1, \quad x = 3 \Rightarrow y = 0$$

für $x = 4, 5$ gibt es kein $y \in \{0, \dots, 5\}$ mit $x + y = 3$.

$$\Rightarrow D_{v_1} \leftarrow \{0, 1, 2, 3\}.$$

Queue wird um die Bögen von Nachbarn von v_1 ergänzt, also (v_3, v_1) (sofern noch nicht vorhanden).

Iteration 2: Bogen (v_2, v_1) Wieder $x + y = 3$, jetzt Richtung $v_2 \to v_1$:

Analog ergibt sich

$$D_{v_2} \leftarrow \{0, 1, 2, 3\}.$$

Queue wird um (v_3, v_2) ergänzt (falls noch nicht vorhanden).

Iteration 3: Bogen (v_2, v_3) Constraint $x + y \leq 3$ mit $x \in D_{v_2} = \{0, 1, 2, 3\}$, $y \in D_{v_3} = \{0, 1, 2, 3, 4, 5\}$.

Für alle $x \in \{0, 1, 2, 3\}$ gibt es jeweils ein y mit $x+y \leq 3$, daher bleibt D_{v_2} unverändert. Keine Änderungen, keine neuen Bögen.

Iteration 4: Bogen (v_3, v_2) Wieder $x + y \leq 3$, jetzt Richtung $v_3 \to v_2$:

Für $y \in D_{v_3}$ (wir nennen hier y den Wert von v_3):

$$\begin{aligned}
&y = 0 : \exists x \in \{0,1,2,3\} \text{ mit } x + 0 \leq 3 \Rightarrow \checkmark \\
&y = 1 : \exists x \in \{0,1,2,3\} \text{ mit } x + 1 \leq 3 \Rightarrow x \in \{0,1,2\} \Rightarrow \checkmark \\
&y = 2 : \exists x \in \{0,1,2,3\} \text{ mit } x + 2 \leq 3 \Rightarrow x \in \{0,1\} \Rightarrow \checkmark \\
&y = 3 : \exists x \in \{0,1,2,3\} \text{ mit } x + 3 \leq 3 \Rightarrow x = 0 \Rightarrow \checkmark \\
&y = 4 : \exists x \in \{0,1,2,3\} \text{ mit } x + 4 \leq 3 \Rightarrow \text{unmöglich} \\
&y = 5 : \exists x \in \{0,1,2,3\} \text{ mit } x + 5 \leq 3 \Rightarrow \text{unmöglich}
\end{aligned}$$

$$\Rightarrow D_{v_3} \leftarrow \{0,1,2,3\}.$$

Queue wird um Bögen mit Ziel v_3 ergänzt, insbesondere (v_1, v_3) und (v_4, v_3) (falls nicht schon in der Queue).

Iteration 5: Bogen (v_1, v_3) Constraint $x \leq y$ mit $x \in D_{v_1} = \{0,1,2,3\}$, $y \in D_{v_3} = \{0,1,2,3\}$.

Für jeden x gibt es mind. ein $y \geq x$:

$$x = 0 \Rightarrow y \in \{0,1,2,3\}, \ \dots, \ x = 3 \Rightarrow y = 3.$$

Keine Änderung der Domänen.

Iteration 6: Bogen (v_3, v_1) Umgekehrte Richtung, wieder $x \leq y$:

Für jeden $y \in D_{v_3}$ gibt es ein $x \in D_{v_1}$ mit $x \leq y$, also bleibt D_{v_3} unverändert.

Iteration 7: Bogen (v_3, v_4) Constraint $x \neq y$ mit $x \in D_{v_3} = \{0,1,2,3\}$, $y \in D_{v_4} = \{0,1,2,3,4,5\}$.

Da D_{v_4} mehr als einen Wert enthält, gibt es für jeden x immer mindestens ein $y \neq x$. Keine Änderung.

Iteration 8: Bogen (v_4, v_3) Umgekehrte Richtung, $x \neq y$ mit $x \in D_{v_4}$, $y \in D_{v_3}$.

Für jeden $x \in \{0,\dots,5\}$ existiert ein $y \in \{0,1,2,3\}$ mit $y \neq x$, daher keine Änderung.

Nach Abarbeiten aller Bögen ist die Queue leer und wir erhalten die kantenkonsistenten Domänen:

$$D_{v_1} = D_{v_2} = D_{v_3} = \{0,1,2,3\}, \qquad D_{v_4} = \{0,1,2,3,4,5\}.$$

CSP.04: Forward Checking und Kantenkonsistenz

Wir betrachten erneut das CSP aus CSP.03 und die Belegung

$$\alpha = \{v_1 \mapsto 2\}.$$

Damit wird $D_{v_1} = \{2\}$ gesetzt, alle anderen Domänen sind zunächst voll: $D_{v_2} = D_{v_3} = D_{v_4} = \{0, 1, 2, 3, 4, 5\}$.

(1) Kantenkonsistenz unter α

Wir erzeugen Kantenkonsistenz für das eingeschränkte Problem.

Constraint $c_1 : (v_1, v_2)$ **mit** $x + y = 3$ Da $v_1 = 2$ fest ist, muss für v_2 gelten: $2 + y = 3$, also $y = 1$.

$$D_{v_2} \leftarrow \{1\}.$$

Constraint $c_3 : (v_1, v_3)$ **mit** $x \leq y$ Da $v_1 = 2$, muss $y \geq 2$ gelten:

$$D_{v_3} \leftarrow \{2, 3, 4, 5\}.$$

Constraint $c_2 : (v_2, v_3)$ **mit** $x + y \leq 3$ Nun $v_2 = 1$, $v_3 \in \{2, 3, 4, 5\}$.

Bedinung: $1 + y \leq 3 \Rightarrow y \leq 2$.

Schnitt mit $\{2, 3, 4, 5\}$ ergibt:

$$D_{v_3} \leftarrow \{2\}.$$

Umgekehrt (Bogen (v_3, v_2)) ist $y = 2$ mit $x = 1$ verträglich, also bleibt $D_{v_2} = \{1\}$.

Constraint $c_4 : (v_3, v_4)$ **mit** $x \neq y$ Da $v_3 = 2$ fest ist, darf v_4 nicht den Wert 2 annehmen:

$$D_{v_4} \leftarrow \{0, 1, 3, 4, 5\}.$$

Ergebnis der Kantenkonsistenz unter α:

$$D_{v_1} = \{2\}, \quad D_{v_2} = \{1\}, \quad D_{v_3} = \{2\}, \quad D_{v_4} = \{0, 1, 3, 4, 5\}.$$

(2) Forward-Checking unter α

Beim Forward-Checking betrachtet man nur die noch unbelegten Nachbarn der gerade belegten Variablen v_1.

- Nachbar v_2 über $c_1 : x + y = 3$ mit $x = v_1 = 2$:

$$2 + y = 3 \Rightarrow y = 1 \Rightarrow D_{v_2} \leftarrow \{1\}.$$

- Nachbar v_3 über $c_3 : x \leq y$ mit $x = 2$:

$$y \geq 2 \Rightarrow D_{v_3} \leftarrow \{2, 3, 4, 5\}.$$

Andere Constraints (c_2 zwischen v_2 und v_3, c_4 zwischen v_3 und v_4) werden beim einfachen Forward-Checking an dieser Stelle noch nicht ausgewertet. Somit:

$$D_{v_1} = \{2\}, \quad D_{v_2} = \{1\}, \quad D_{v_3} = \{2, 3, 4, 5\}, \quad D_{v_4} = \{0, 1, 2, 3, 4, 5\}.$$

Vergleich

- Kantenkonsistenz (AC-3) propagiert die Einschränkungen weiter durch den gesamten Graphen. Dadurch werden zusätzlich D_{v_3} auf $\{2\}$ und D_{v_4} auf $\{0, 1, 3, 4, 5\}$ reduziert.
- Forward-Checking betrachtet nur direkte Nachbarn der belegten Variablen und verpasst dadurch einige Inkonsistenzen (größere Domänen für v_3 und v_4).

CSP.05: Planung von Indoor-Spielplätzen

Wir modellieren das Platzierungsproblem als CSP.

Variablen und Domänen

Wir vereinfachen den Spielplatz als diskretes Raster von $W \times H$ Zellen, z. B. 40×100 Rastereinheiten.

- Spielgeräte:
  - Go-Kart-Bahn (rechteckig, z. B. 10×30),
  - Hüpfburg (z. B. 10×10),
  - Kletterberg (z. B. 8×8),
  - Bar (z. B. 8×12).
- Für jedes Objekt i definieren wir eine Positionsvariable (X_i, Y_i) für die linke obere Ecke.

Domänen:

$$X_i \in \{0, \dots, W - w_i\}, \quad Y_i \in \{0, \dots, H - h_i\},$$

wobei w_i, h_i die Breite/Höhe des Objekts i sind.

Constraints

Nicht-Überlappung und Mindestabstand Zwei Spielgeräte i und j dürfen sich nicht überlappen und müssen mindestens einen Abstand von d (z. B. 1 m) haben. Dies modellieren wir über die rechteckigen Ausdehnungen:

Sei

$$\text{Rechteck}_i = [X_i, X_i + w_i) \times [Y_i, Y_i + h_i).$$

Dann verlangen wir für alle $i \neq j$:

$$\text{dist}(\text{Rechteck}_i, \text{Rechteck}_j) \geq d.$$

Dies kann über disjunktive Constraints ausgedrückt werden, z. B.:

$$(X_i + w_i + d \leq X_j) \ \vee \ (X_j + w_j + d \leq X_i) \ \vee \ (Y_i + h_i + d \leq Y_j) \ \vee \ (Y_j + h_j + d \leq Y_i).$$

Bar in Eingangsnähe Angenommen, der Eingang liegt im Bereich $E = [0, e_x] \times [0, e_y]$. Dann fordern wir:

$$X_{\text{Bar}} \leq e'_x, \quad Y_{\text{Bar}} \leq e'_y,$$

oder allgemeiner eine Begrenzung in der Nähe einer Eingangswand.

Notausgänge freihalten Für jede Notausgangszone Z_k (z. B. Rechtecke am Rand des Spielfelds) gilt für jedes Spielgerät i:

$$\text{Rechteck}_i \cap Z_k = \emptyset.$$

Sichtlinie Kletterberg – Bar Wir fordern eine freie Sichtlinie, z. B. entlang einer horizontalen oder vertikalen Achse. Eine Möglichkeit:

Es existiert eine Gitterlinie (feste y-Koordinate) von der Bar zum Kletterberg, auf der keine anderen Spielgeräte stehen. Dies lässt sich als zusätzliche Menge von Nicht-Überlappungs-Constraints formulieren.

Lösung mit MAC und Min-Conflicts

MAC (Maintaining Arc Consistency) Wir verwenden BT_Search mit AC-3 als Inferenzschritt (MAC):

- Nach jeder Zuweisung (X_i, Y_i) wird AC-3 auf die verbleibenden Variablen angewendet, um deren Domänen zu reduzieren.
- Dadurch werden viele unzulässige Positionen früh ausgeschlossen, die Anzahl der Backtracking-Schritte sinkt deutlich.

Min-Conflicts Alternativ kann eine Min-Conflicts-Heuristik eingesetzt werden:

- Initialisierung mit zufälliger vollständiger Belegung aller (X_i, Y_i) innerhalb der erlaubten Bereiche.
- In jedem Schritt wird ein Objekt gewählt, das momentan Constraints verletzt, und seine Position so geändert, dass die Anzahl der Konflikte minimal wird.
- Durch Wiederholung und ggf. Restarts erhält man typischerweise schnell eine konfliktfreie Anordnung der Spielgeräte.

Beispielhafte Lösung (schematisch)

Auf einem Raster 40×100 könnte z. B. eine Lösung sein:

$$\begin{aligned}
(X_{\text{Bar}}, Y_{\text{Bar}}) &= (0, 0),\\
(X_{\text{Kart}}, Y_{\text{Kart}}) &= (10, 0),\\
(X_{\text{Hüpfburg}}, Y_{\text{Hüpfburg}}) &= (0, 50),\\
(X_{\text{Kletterberg}}, Y_{\text{Kletterberg}}) &= (20, 50),
\end{aligned}$$

wobei alle Abstands- und Sichtlinien-Constraints erfüllt sind (Bar nahe am Eingang, Notausgänge freigehalten, Mindestabstände eingehalten).