

CSP01: Logikrätsel (Einsteinrätsel)

Formulierung als CSP

Wir modellieren das Rätsel als Constraint Satisfaction Problem $CSP = \langle V, D, C \rangle$.

Variablen

Für jede Eigenschaft wird eine Variable definiert, deren Wert die Hausnummer $1, \dots, 5$ ist. Die Variablen sind in Kategorien unterteilt:

- Farben: {Rot, Gruen, Weiss, Gelb, Blau}
- Nationalitäten: {Engländer, Spanier, Ukrainer, Norweger, Japaner}
- Getränke: {Kaffee, Tee, Milch, OSoft, Wasser}
- Haustiere: {Hund, Schnecken, Fuchs, Pferd, Zebra}
- Zigarettensmarken: {OldGold, Kools, Chesterfield, LuckyStrike, Parliament}

Domänen

Alle Variablen besitzen dieselbe Domäne:

$$D = \{1, 2, 3, 4, 5\}.$$

Allgemeine Constraints

Innerhalb jeder Kategorie muss jede Eigenschaft genau einem Haus zugeordnet werden. Daher gilt pro Kategorie ein All-Different Constraint, beispielsweise für die Farben:

$$\text{Rot} \neq \text{Gruen} \neq \text{Weiss} \neq \text{Gelb} \neq \text{Blau}.$$

Analog gilt dies für Nationalitäten, Getränke, Haustiere und Zigarettensmarken.

Constraints aus den Hinweisen

Die Textaussagen werden als Constraints formuliert:

Englaender = Rot
Spanier = Hund
Gruen = Kaffee
Ukrainer = Tee
Gruen = Weiss + 1
OldGold = Schnecken
Kools = Gelb
Milch = 3
Norweger = 1
 $|\text{Chesterfield} - \text{Fuchs}| = 1$
 $|\text{Kools} - \text{Pferd}| = 1$
LuckyStrike = OSaft
Japaner = Parliament
 $|\text{Norweger} - \text{Blau}| = 1$

Lösung

Durch Lösen des CSP ergibt sich die folgende Zuordnung für den Häuser:

| Haus | Farbe | Nationalität | Haustier | Getränk | Zigaretten |
|------|-------|--------------|-----------|-------------|--------------|
| 1 | gelb | Norweger | Fuchs | Wasser | Kools |
| 2 | blau | Ukrainer | Pferd | Tee | Chesterfield |
| 3 | rot | Englaender | Schnecken | Milch | OldGold |
| 4 | weiss | Spanier | Hund | Orangensaft | LuckyStrike |
| 5 | gruen | Japaner | Zebra | Kaffee | Parliament |

Also:

- Das Zebra gehört dem Japaner.
- Wasser wird vom Norweger getrunken.

CSP.02 Aufgabe 2: Vergleich von Suchverfahren

1. Basis Backtracking Suche

Der Algorithmus wählt Variablen in fixer Reihenfolge und probiert Werte durch. Wichtige Schritte:

- Norweger muss in Haus 1 stehen.
- Milch muss in Haus 3 stehen.
- Blau muss neben dem Norweger stehen, also Haus 2.
- Grün steht direkt rechts von Weiss, daher Weiss Haus 4 und Grün Haus 5.
- Kaffee im grünen Haus zwingt Kaffee nach Haus 5.
- Ukrainer trinkt Tee, einzig möglich ist Haus 2.
- Spanier hat Hund, einzig möglich ist Haus 4.
- Japaner raucht Parliament, einzig möglich ist Haus 5.

Nach und nach entsteht die eindeutige Lösung. Backtracking wird nur nötig, wenn ein Wert direkt ein Constraint verletzt.

2. Backtracking mit MRV und Gradheuristik

MRV wählt zuerst Variablen mit kleiner Domäne. Da einige Constraints stark einschränken, sind diese Variablen:

- Norweger (nur Haus 1)
- Milch (nur Haus 3)
- Blau (nur Haus 2)
- Weiss und Grün (nur die Kombination 4 und 5 ist möglich)

Die Suche beginnt daher sofort mit den wichtigsten Variablen. Dadurch entstehen fast keine Sackgassen und weniger Rücksprünge. Die Lösung wird wesentlich schneller gefunden als bei reinem Backtracking.

3. AC3 vor der Suche

AC3 reduziert die Domänen, bevor die Suche startet. Beispiele:

- Norweger wird sofort auf Haus 1 reduziert.
- Blau wird auf Haus 2 reduziert.
- Weiss und Grün bekommen nur noch Werte 4 und 5.
- Kaffee und Grün fallen zusammen auf Haus 5.

Danach ist ein grosser Teil der möglichen Zuordnungen schon ausgeschlossen. Wenn man jetzt Backtracking mit MRV nutzt, ist die Suche nochmal kürzer. Die Lösung entsteht fast ohne Rücksprünge.

4. Min Conflicts

Min Conflicts arbeitet nicht wie Backtracking, sondern verbessert eine vollständige zufällige Belegung Schritt für Schritt:

1. Start mit einer beliebigen Belegung aller Häuser.
2. Eine Variable mit Konflikten wird gewählt.
3. Sie bekommt einen Wert, der die Zahl der Konflikte minimiert.
4. Wiederholen, bis alle Constraints erfüllt sind.

Für dieses Rätsel findet Min Conflicts die korrekte Lösung meist sehr schnell, weil jede Änderung mehrere Konflikte löst. Die Methode garantiert keine Lösung, funktioniert hier aber sehr zuverlässig.