Praktika 5

Aufgabe 1

> 1. Es gibt fünf Häuser.
> 2. Der Engländer wohnt im roten Haus.
> 3. Der Spanier hat einen Hund.
> 4. Kaffee wird im grünen Haus getrunken.
> 5. Der Ukrainer trinkt Tee.
> 6. Das grüne Haus ist direkt rechts vom weißen Haus.
> 7. Der Raucher von Old-Gold-Zigaretten hält Schnecken als Haustiere.
> 8. Die Zigaretten der Marke Kools werden im gelben Haus geraucht.
> 9. Milch wird im mittleren Haus getrunken.
> 10. Der Norweger wohnt im ersten Haus.
> 11. Der Mann, der Chesterfield raucht, wohnt neben dem Mann mit dem Fuchs.
> 12. Die Marke Kools wird geraucht im Haus neben dem Haus mit dem Pferd.
> 13. Der Lucky-Strike-Raucher trinkt am liebsten Orangensaft.
> 14. Der Japaner raucht Zigaretten der Marke Parliaments.
> 15. Der Norweger wohnt neben dem blauen Haus.
>
> Wer trinkt Wasser? Wem gehört das Zebra?

Häuser (Blau,Rot, Weiß ,Grün, Gelb)

Nationalität (Norwegen, England, Ukrainien, Spanien ,Japan)

Haustier(Zebra, Pferd, Fuchs , Hund , Schnecken)

Zigerreten(Old Gold , Kools , chesterfield , Lucky Strike , Parliaments)

Getränke (Kaffe , Wasser ,Tee , Milch , Orangensaft)

Häuser (Rot=1 , Blau=2, Weiß=3, Grün=4, Gelb=5)

Constraints:

Für jede Kategorie gilt: alle 5 Werte kommen genau einmal vor

Engländer = Rot

Spanier hat ein Hund

Grün = Kaffe

Ukrainer trinkt tee

Grün ist rechts von Weiß

Old Gold hat schnecken als haustier

Gelb = Kools , neben haus hat ein Pferd

Milch in mittleren haus

Norweger in ersten haus , wohnt neben blau

der Chesterfield raucher wohnt neben den Haus der ein Fuchs hat

Japaner raucht Parliaments

Lucky strike trinkt Orangensaft

Soft Constraints:

Gibts nicht

Lösung:

1 = Gelb, Norwegen, Kools , Wasser, Fuchs

2= Blau, Ukrainer ,Chesterfield , Tee ,Pferd

3=Rot , Engländer , Milch , Old Gold, Schnecken

4=Weiß,Spanier, Lucky strike ,Orangensaft, Hund

5=Grün,Japaner, Parliaments , Kaffe, Zebbra

Aufgabe 2

Ich habe zuerst ganz normal Backtracking verwendet.
Dabei benutze ich die offensichtlichen Hinweise direkt:

Der Norweger ist im ersten Haus → das setze ich sofort.

Milch im dritten Haus → ebenfalls sofort gesetzt.

Der Norweger wohnt neben dem blauen Haus → also ist Haus 2 blau.

Jetzt prüfe ich die möglichen Farben für Haus 1.
Rot geht nicht (weil Engländer im roten Haus wäre).
Weiß geht nicht (dann müsste Haus 2 grün sein, ist aber blau).
Grün geht auch nicht (grün muss rechts von weiß sein).
Also bleibt Haus 1 = gelb.

Damit ist Kools im gelben Haus → also Kools in Haus 1.
Kools ist neben dem Pferd → also Pferd in Haus 2.

So gehe ich Schritt für Schritt weiter, bis alle Häuser festgelegt sind.
Am Ende kommt die bekannte Lösung raus:

Wasser trinkt der Norweger

Das Zebra gehört dem Japaner

Aufgabe 2.2

Hier habe ich denselben Algorithmus benutzt, nur mit den zwei Heuristiken:

MRV → zuerst die Variable mit den wenigsten möglichen Werten

Gradheuristik → bei Gleichstand die Variable mit den meisten Constraints

Das Ergebnis ist genau das gleiche wie davor, aber der Algorithmus macht weniger Fehlversuche.
Er findet die passende Reihenfolge schneller und muss weniger zurückspringen.

Aufgabe 2.3

Hier habe ich vor dem Backtracking einmal AC-3 ausgeführt.

Dadurch werden einige Werte schon aus den Domains gestrichen, weil sie nicht konsistent wären.
Zum Beispiel bleiben bei manchen Häusern nur noch ein oder zwei mögliche Farben übrig.

Wenn man danach mit BT_Search + MRV weitermacht, findet der Algorithmus die Lösung fast sofort, weil kaum noch unnötige Möglichkeiten übrig sind.

Die Lösung bleibt natürlich dieselbe wie vorher.

Aufgabe 2.4

Bei Min-Conflicts starte ich nicht mit einer leeren Belegung, sondern mit einer zufälligen kompletten Belegung.
Dann schaue ich immer nur, welche Variable gerade Konflikte hat, und setze sie so, dass es weniger Konflikte gibt.

Das läuft schrittweise weiter, bis keine Konflikte mehr da sind.

Auch hier landet man schließlich bei derselben Lösung wie vorher.
Der Unterschied ist nur, dass der Algorithmus eher „hinprobiert", statt alle Möglichkeiten systematisch auszuprobieren.

Für dieses Rätsel funktioniert Min-Conflicts ziemlich gut.

Aufgabe 3

1) Grobe Zeichnung (Constraint-Graph)

v1 -------- v2 -------- v3 -------- v4

. .

.__________.

(Die Punkte sollen nur andeuten, dass zwischen v3 und v4 zwei Constraints bestehen.)

2) AC-3 Algorithmus

Domains

Alle Variablen starten identisch:

`D(v1) = D(v2) = D(v3) = D(v4) = {0,1,2,3,4,5}`

Initiale Queue

`Q0 = [(v1,v2), (v2,v1), (v2,v3), (v3,v2), (v3,v4), (v4,v3)]`

Iteration 1: (v1, v2)

Constraint: **x + y = 3**

Mögliche Paare:

- 0 → y = 3
- 1 → y = 2
- 2 → y = 1
- 3 → y = 0
- 4 → geht nicht
- 5 → geht nicht

Damit darf v1 nur Werte benutzen, die eine Lösung zulassen.

ARC-Reduce(v1, v2) entfernt {4,5}

Neue Domain:

`D(v1) = {0,1,2,3}`

Nachbarn von v1 sind: v2 und v3 → Arcs zurück in die Queue.

Weitere Iterationen

Die weiteren Schritte laufen nach demselben Prinzip weiter:

- Wegen c1 und c2 reduzieren sich v1, v2 und v3 auf `{0,1,2,3}`.
- c3 (v3 ≤ v4) und c4 (v4 ≠ v3) führen zu **keiner** weiteren Reduktion.
  Für jeden Wert von v4 gibt es immer einen gültigen Partner in v3.

Endzustand der Domains (nach AC-3)

D(v1) = {0,1,2,3}

D(v2) = {0,1,2,3}

D(v3) = {0,1,2,3}

D(v4) = {0,1,2,3,4,5}

Aufgabe 4

Start-Domains (unter α):

v1={2}

v2,v3,v4={0,1,2,3,4,5}

Durch das Constraint v1+v2=3 bleibt für v2 nur noch der passende Wert übrig:

v2={1}

Damit ist für das nächste Constraint v2+v3=3 nur ein Wert möglich:

v3={2}

Jetzt bleiben nur noch die Constraints zwischen v3 und v4.
Mit v3=2 muss v4 größer sein und darf nicht gleich v3 sein:

v4={3,4,5}

Endzustand nach Kantenkonsistenz:

- v1={2}
- v2={1}
- v3={2}
- v4={3,4,5}

2)

Forward-Checking schaut nur auf Variablen, die direkt von v1v_1v1 abhängen.
Das betrifft nur das Constraint v1+v2=3.

Also bleibt:

v2={1}

Die anderen Domains bleiben unangetastet:

- v3={0,1,2,3,4,5}
- v4={0,1,2,3,4,5}

Aufgabe 5

Für jedes Spielgerät Gi wird die Position der linken oberen Ecke als Variable modelliert:

- Xi: x-Koordinate des Geräts
- Yi: y-Koordinate des Geräts
- Wi: feste Breite des Geräts
- Hi: feste Höhe des Geräts

Für die Bar verwenden wir entsprechend:

- Xba Ybar

Damit ergibt sich die komplette Variablenmenge:

V={X1,Y1,X2,Y2,X3,Y3, Xbar,Ybar}

Beispielhafte Spielgeräte

1. Go-Kart-Strecke G1G_1G1: Maße **10 × 25**
2. Hüpfburg G2G_2G2: Maße **8 × 12**
3. Kletterburg G3G_3G3: Maße **12 × 12**
4. Bar BBB: Maße **15 × 8**

Domain

Die Halle besitzt ein Raster von **40 Spalten** × **100 Reihen**.

Damit ergeben sich die Wertebereiche:

Für jedes Gerät Gi:

D(Xi)={0,…,40−Wi}

D(Yi)={0,…,100−Hi}

Für die Bar analog:

$D(Xbar)=\{0,\ldots,40-Wbar\}$

$D(Ybar)=\{0,\ldots,100-Hbar\}$

Constraints

1. Keine Überlappung zwischen Geräten

Für zwei beliebige Geräte GiG_iGi und GjG_jGj gilt:

$(X_i + W_i =< X_j) \lor (X_j + W_j =< X_i) \lor (Y_i + H_i =< Y_j) \lor (Y_j + H_j =< Y_i)$

Diese Bedingung stellt sicher, dass sich die Rechtecke nicht schneiden.

2. Sicherheitsabstand von 1 Rasterpunkt

Der Abstand wird modelliert, indem jedes Gerät künstlich „vergrößert" wird:

$W_i' = W_i + 1$

$H_i' = H_i + 1$

3. Bar bevorzugt in Eingangsbereich

Ein Soft-Constraint :

$Ybar \leq 10$

Die Bar liegt somit im oberen Bereich des Spielfelds.

4. Freie Sicht von der Bar zur Kletterburg

Zwischen Bar Bund Kletterburg G3 darf sich kein anderes Gerät befinden.
Eine einfache Sichtlinien-Anforderung lautet beispielsweise:

$Xbar < X3$

Damit liegt die Bar links der Kletterburg und die Sicht ist nicht direkt verbaut.