

Übungsblatt: Constraints

CSP.01: Logikrätsel

1. Problembeschreibung

Beim bekannten „Einstein-Rätsel“ gibt es 5 Häuser in einer Reihe.
Jedes Haus hat:

eine Farbe

eine Nationalität des Bewohners

ein Getränk

eine Zigarettenmarke

ein Haustier

Alle Eigenschaften kommen jeweils genau einmal vor.

Gegeben sind 15 logische Hinweise, aus denen man durch logisches Schließen herausfinden muss, wer Wasser trinkt und wem das Zebra gehört.

2. Formulierung als CSP

Variablen

Für jedes Haus $i=1..5$:

$\text{Color}[i] \in \{\text{rot, grün, weiß, gelb, blau}\}$

$\text{Nation}[i] \in \{\text{Engländer, Spanier, Ukrainer, Norweger, Japaner}\}$

$\text{Drink}[i] \in \{\text{Kaffee, Tee, Milch, Orangensaft, Wasser}\}$

$\text{Smoke}[i] \in \{\text{OldGold, Kools, Chesterfield, LuckyStrike, Parliaments}\}$

$\text{Pet}[i] \in \{\text{Hund, Schnecke, Fuchs, Pferd, Zebra}\}$

Globale Constraints

Für jede Kategorie gilt:

AllDifferent (jede Farbe / Nationalität / Getränk / Zigarette / Tier genau einmal)

Constraints aus den Hinweisen

Kurzfassung:

Engländer → rotes Haus

Spanier → Hund

Grünes Haus → Kaffee

Ukrainer → Tee

Weißes Haus steht links direkt neben dem grünen

OldGold → Schnecken

Kools → gelbes Haus

Haus 3 trinkt Milch

Norweger wohnt in Haus 1

Chesterfield wohnt neben dem Fuchs

Kools wohnt neben dem Pferd

LuckyStrike → Orangensaft

Japaner → Parliaments

Norweger wohnt neben dem blauen Haus

3. Lösungsweg

Aus Hinweis 9:

→ Haus 1 = Norweger

Aus Hinweis 14:

→ Blau muss Haus 2 sein.

Aus Hinweis 8:

→ Haus 3 trinkt Milch.

Weiß→Grün-Paar kann wegen Farbe der Häuser nur an Position (4→5) stehen:

→ Haus 4 = weiß, Haus 5 = grün.

Kaffee wird im grünen Haus getrunken →

→ Haus 5 = Kaffee.

Gelb ist das einzige freie Haus mit passender Domäne →

→ Haus 1 = gelb → raucht Kools.

Kools steht neben Pferd →

→ Haus 2 = Pferd.

Chesterfield muss neben Fuchs wohnen →

→ Haus 2 raucht Chesterfield und Haus 1 hat Fuchs.

OldGold ↔ Schnecken passt nur für Haus 3:

→ Haus 3 = OldGold + Schnecke.

Engländer wohnt im roten Haus →
→ Haus 3 = rot → Nation = Engländer.

LuckyStrike ↔ Orangensaft passt für Haus 4:
→ Haus 4 = Orangensaft + LuckyStrike.

Spanier ↔ Hund →
→ Haus 4 = Spanier + Hund.

Japaner ↔ Parliaments →
→ Haus 5 = Japaner + Parliaments + Zebra.

Übrig bleibt im letzten Schritt:
→ Haus 1 = Wasser.

4. Endgültige Lösungstabelle

Haus	Nationalität	Farbe	Getränk	Zigarette	Haustier
1	Norweger	gelb	Wasser	Kools	Fuchs
2	Ukrainer	blau	Tee	Chesterfield	Pferd
3	Engländer	rot	Milch	OldGold	Schnecke
4	Spanier	weiß	Orangensaft	LuckyStrike	Hund
5	Japaner	grün	Kaffee	Parliaments	Zebra

5. Antwort auf die Fragen

Wer trinkt Wasser?
→ Der Norweger (Haus 1)

Wem gehört das Zebra?
→ Dem Japaner (Haus 5)

CSP.02: Framework für Constraint Satisfaction (Handsimmulation)

1. CSP-Formulierung

Variablen (pro Haus 1–5):

Color: rot, grün, weiß, gelb, blau

Nation: Engländer, Spanier, Ukrainer, Norweger, Japaner

Drink: Kaffee, Tee, Milch, Orangensaft, Wasser

Smoke: OldGold, Kools, Chesterfield, LuckyStrike, Parliaments

Pet: Hund, Schnecke, Fuchs, Pferd, Zebra

Constraints:

Binär: Nachbarschaften (z. B. Norweger neben blau, Kools neben Pferd, Chesterfield neben Fuchs)

Binär: direkte Zuordnung (z. B. Engländer = rot, Spanier = Hund)

Binär: Reihenfolge (z. B. grünes Haus rechts von weiß)

Unär: feste Werte (Milch in Haus 3, Norweger in Haus 1)

2. Handsimulation – Schritt für Schritt

Tabelle Haus 1–5:

Haus	1	2	3	4	5
Color	Gelb	Blau	Rot	Weiß	Grün
Nation	Norweger	Ukrainer	Engländer	Spanier	Japaner
Drink	Wasser	Tee	Milch	Orangensaft	Kaffee
Smoke	Kools	Chesterfield	LuckyStrike	OldGold	Parliaments
Pet	Fuchs	Pferd	Hund	Schnecke	Zebra

Schrittweise Ableitung:

Haus 1: Norweger → Farbe Gelb (nur möglich neben Blau)

Haus 2: Blau (neben Norweger)

Haus 3: Milch → Engländer = rotes Haus

Haus 4 & 5: Weiß & Grün (grün rechts von weiß)

Getränke: Kaffee → grünes Haus (Haus 5), Tee → Ukrainer, O-Saft → LuckyStrike

Rauch: Kools → gelbes Haus (Haus 1), OldGold → Schnecke, Parliaments → Japaner

Haustiere: Spanier = Hund, Chesterfield neben Fuchs, Kools neben Pferd

3. Vergleich der „Algorithmen“ (Handsimmulation)

Methode	Aufwand / Schritte	Bemerkung
Basis-Backtracking	20–30 logische Schritte	Schrittweise Tabellenfüllung, Rücksprünge bei Konflikten
Backtracking + MRV + Degree	10–15 Schritte	Variablen mit kleinster Domain zuerst ,schneller
AC-3 vor Suche	5–10 Schritte	Domänen stark reduziert ,fast sofortige Lösung
Min-Conflicts	5–10 Konfliktkorrekturen	Probabilistisch, in Handsimulation sehr effizient

Hinweis: Alle Methoden liefern dasselbe Ergebnis, nur die Effizienz unterscheidet sich.

CSP.03: Kantenkonsistenz mit AC-3

1. AC-3 – Handsimulation

Initiale Domänen aller Variablen:

$$Dv1=Dv2=Dv3=Dv4=\{0,1,2,3,4,5\}$$

Initiale Queue (alle Arcs):

$(v1,v2),(v2,v1),(v2,v3),(v3,v2),(v1,v3),(v3,v1),(v3,v4),(v4,v3)$

2. Iterationstabelle des AC-3 Algorithmus

(Die Tabelle entspricht der vollständigen, detaillierten Handsimulation.)

Iteration	Arc ($X_i \rightarrow X_j$)	Entfernte Werte	Neue Domain	Queue nach Pop
Start	–	–	$v1=v2=v3=v4=\{0,1,2,3,4,5\}$	$(v1,v2),(v2,v1),(v2,v3),(v3,v2),(v1,v3),(v3,v1),(v3,v4),(v4,v3)$
1	$(v1, v2)$	$\{4,5\}$	$v1=\{0,1,2,3\}$	$(v2,v1),(v2,v3),(v3,v2),(v1,v3),(v3,v1),(v3,v4),(v4,v3)$
2	$(v2, v1)$	$\{4,5\}$	$v2=\{0,1,2,3\}$	$(v2,v3),(v3,v2),(v1,v3),(v3,v1),(v3,v4),(v4,v3),(v1,v2)$
3	$(v2, v3)$	\emptyset	$v2=\{0,1,2,3\}$	$(v3,v2),(v1,v3),(v3,v1),(v3,v4),(v4,v3),(v1,v2)$
4	$(v3, v2)$	$\{4,5\}$	$v3=\{0,1,2,3\}$	$(v1,v3),(v3,v1),(v3,v4),(v4,v3),(v1,v2),(v2,v3),(v2,v1)$
5	$(v1, v3)$	\emptyset	$v1=\{0,1,2,3\}$	$(v3,v1),(v3,v4),(v4,v3),(v1,v2),(v2,v3),(v2,v1)$
6	$(v3, v4)$	\emptyset	$v3=\{0,1,2,3\}$	$(v3,v4),(v4,v3),(v1,v2),(v2,v3),(v2,v1)$

	v1)			
7	(v 3, v4)	∅	v3={0,1,2,3}	(v4,v3),(v1,v2),(v2,v3),(v2,v1)
8	(v 4, v3)	∅	v4={0,1,2,3,4,5}	(v1,v2),(v2,v3),(v2,v1)
9	(v 1, v2)	∅	v1={0,1,2,3}	(v2,v3),(v2,v1)
10	(v 2, v3)	∅	v2={0,1,2,3}	(v2,v1)
11	(v 2, v1)	∅	v2={0,1,2,3}	(v4,v3)
12	(v 4, v3)	∅	v4={0,1,2,3,4,5}	Queue leer → Ende

3. Ergebnis: Kantenkonsistente Domänen

Nach Abschluss von AC-3 ergeben sich folgende Domänen:

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

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

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

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

Interpretation:

Durch die Summen-Constraints fallen die Werte 4 und 5 bei v1,v2 und v3 weg.

Für v4 bleiben alle Werte möglich, da das einzige Constraint (Ungleichheit) immer einen passenden Wert in D(v3) findet.

4. Kurzes Fazit

Der AC-3-Algorithmus reduziert die Domänen der beteiligten Variablen effektiv, noch bevor eine eigentliche Suche beginnt.

In diesem CSP wurden die Werte 4 und 5 für die ersten drei Variablen ausgeschlossen,

während v_4 seine komplette Domäne behalten kann.
Damit ist das CSP nun komplett kantenkonsistent.

CSP04 – Forward Checking und Kantenkonsistenz

Gegeben

Wir betrachten dasselbe CSP wie in Aufgabe CSP03 und die Zuweisung
 $\alpha = \{v_1 \mapsto 2\}$.

1. Kantenkonsistenz in α

Ausgangsdomänen (vor Herstellung der Kantenkonsistenz)

Variable	Ausgangsdomäne
v_1	$\{2\}$
v_2	$\{0,1,2,3,4,5\}$
v_3	$\{0,1,2,3,4,5\}$
v_4	$\{0,1,2,3,4,5\}$

Da v_1 bereits gesetzt ist, besitzt es nur den Wert 2.

Herstellung der Kantenkonsistenz

Constraint $c_1(v_1, v_2): x + y = 3$

$$2+y=3 \Rightarrow y=1$$

$$\rightarrow D(v_2)=\{1\}$$

Constraint $c_3(v_1, v_3): x \leq y$

$$2 \leq y \Rightarrow y \in \{2, 3, 4, 5\}$$

$$\rightarrow D(v_3)=\{2, 3, 4, 5\}$$

Constraint $c_2(v_2, v_3): x+y \leq 3$ mit $v_2=1$

$$1+y \leq 3 \Rightarrow y \leq 2$$

Schnitt mit aktueller Domäne $\rightarrow D(v_3)=\{2\}$

Constraint $c_4(v_3, v_4): x \neq y$ mit $v_3=2$

$$v_4 \neq 2$$

$$\rightarrow D(v_4)=\{0, 1, 3, 4, 5\}$$

Domänen nach der Kantenkonsistenz

Variable	Domäne nach Konsistenz
v_1	$\{2\}$
v_2	$\{1\}$
v_3	$\{2\}$
v_4	$\{0, 1, 3, 4, 5\}$

2. Forward Checking in α

Forward Checking prüft nur die Constraints der zuletzt gesetzten Variable (hier: v_1) mit ihren direkten Nachbarn.

Schritte

Constraint $c_1(v_1, v_2)$

$$\rightarrow 2+y=3 \rightarrow D(v_2)=\{1\}$$

Constraint $c_3(v_1, v_3)$

$$\rightarrow 2 \leq y \rightarrow D(v_3)=\{2, 3, 4, 5\}$$

Keine Überprüfung für v_4

(Forward Checking prüft nur direkte Nachbarn.)

Domänen nach Forward Checking

Variable	Domäne nach FC
v1v_1v1	{2}
v2v_2v2	{1}
v3v_3v3	{2,3,4,5}
v4v_4v4	{0,1,2,3,4,5}

Vergleich Kantenkonsistenz vs. Forward Checking

Variable	Nach Kantenkonsistenz	Nach Forward Checking	Unterschied
v1v_1v1	{2}	{2}	–
v2v_2v2	{1}	{1}	–
v3v_3v3	{2}	{2,3,4,5}	KC propagiert weiter, FC nicht
v4v_4v4	{0,1,3,4,5}	{0–5}	KC schließt Wert 2 aus

Fazit

Kantenkonsistenz ist stärker: Sie propagiert weiter ($v1 \rightarrow v2 \rightarrow v3 \rightarrow v4$) und schränkt Domänen stärker ein.

Forward Checking ist schwächer, betrachtet nur direkte Nachbarn des gesetzten Knotens.

Beide Verfahren sind konsistent mit der Zuweisung $v1=2$, aber die Konsistenzreduktion fällt unterschiedlich aus.

CSP05 – Planung von Indoor-Spielplätzen

1. Problemabstraktion

Ein Indoor-Spielplatz besitzt eine rechteckige Grundfläche, die zur Modellierung als gleichmäßiges Raster betrachtet wird. Jedes Rasterfeld repräsentiert eine Position, auf der ein Spielgerät platziert werden kann. Zusätzlich gibt es am Rand Türen (Ein- bzw. Notausgänge), die nicht blockiert werden dürfen.

Auf dieser Fläche sollen mehrere rechteckige Spielgeräte angeordnet werden. Die Geräte dürfen sich nicht überlappen, müssen Sicherheitsabstände einhalten und bestimmte Sichtlinien bzw. Platzierungsregeln beachten.

CSP-Modellierung

Variablen:

Für jedes Spielgerät X_i eine Variable, die die Top-Left-Position (x,y) im Raster beschreibt.

Domänen:

Alle Positionen (x,y) , an denen das Gerät vollständig innerhalb des Rasters liegt und keine Tür überdeckt.

Beispielgeräte (mit Abmessungen in Rasterzellen):

GoKart-Bahn: 4×2

Bouncy (Hüpfburg): 3×3

Climb (Kletterberg): 2×3

Bar: 4×2

RelaxZone: 2×2

Rastergröße: 12×8

Türen: $(0,4)$ und $(11,1)$

Sicherheitsabstand: 1 Rasterzelle um jedes Gerät herum.

2. Constraints

1. Sicherheitsabstand und Nicht-Überlappung

Für alle Paare von Geräten A und B gilt:

Die erweiterten Rechtecke

$A' = A$ erweitert um 1 Zelle in jede Richtung

$B' = B$ erweitert um 1 Zelle in jede Richtung

dürfen sich nicht überschneiden:

$$A' \cap B' = \emptyset$$

Dies garantiert:

keine Überlappung

kein Berühren

mindestens 1 m Abstand

2. Türen dürfen nicht verdeckt werden

Für jedes Gerät X_i :

$$\text{Fläche}(X_i) \cap \text{Türen} = \emptyset$$

3. Geräte müssen vollständig im Raster liegen

$$0 \leq x \leq W - w_i$$

$$0 \leq y \leq H - h_i$$

4. Weiches Constraint

Die Bar soll möglichst nah am Eingang (0,4) sein

→ wird im Min-Conflicts-Ansatz als Zielgröße genutzt.

3. Lösungsmethoden

MAC (Maintaining Arc Consistency)

AC-3 zur Domänenreduktion

Backtracking suchte anschließend eine vollständige, konsistente Belegung

MRV-Heuristik + fortlaufende Arc-Konsistenz

Resultat: exakte Lösung ohne Konflikte

4. Ergebnis der MAC-Suche

Folgende konfliktfreie Positionen wurden gefunden (Top-Left-Koordinaten):

Gerät	Position
Bouncy (3×3)	(0,0)
GoKart (4×2)	(0,5)
Bar (4×2)	(5,0)

Climb (2×3)	(6,4)
RelaxZone (2×2)	(10,4)

Alle Constraints erfüllt, Türen bleiben frei, Sicherheitsabstände eingehalten.

5. Schlussfolgerung

Die Modellierung zeigt, dass das Indoor-Spielplatz-Layout sehr gut als Constraint Satisfaction Problem formulierbar ist. Der Einsatz von MAC erlaubt eine deterministische, vollständige Lösung, während Min-Conflicts in größeren Real-World-Instanzen schnelle, wenn auch nicht garantierte Ergebnisse liefert.

Im gegebenen Beispiel konnte MAC eine vollständig gültige Konfiguration für alle Spielgeräte finden, inklusive Sicherheitsabstand und freigehaltenen Türen.