Blatt 05: Constraints

Michel Bünger

CSP.01: Logikrätsel

Das "Einstein-Rätsel" besitzt fünf Häuser, welche die Variablen des CSP sind.
Diese Häuser werden Beschrieben, als: H $\in$ { H_1, H_2, H_3, H_4, H_5 }

Die vorhandenen Variablen, welche die Häuser und dessen Bewohner beschreiben, im gegebenem "Einstein-Rätsel", sind:

- Farbe
- Haustier
- Getränk
- Nationalität
- Zigarettenmarke

Die Variablen werden als $< Variablenname > \_ < Haus >$ dargestellt
(bsp. Farbe_H_2)

Die Wertebereiche der obigen Variablen sind:

Farbe: { rot, grün, weiß, gelb, blau }

Haustier: { Hund, Schnecke, Fuchs, Pferd, Zebra }

Getränk: { Kaffee, Tee, Milch, Orangensaft, Wasser }

Nationalität: { Engländer, Spanier, Ukrainer, Norweger, Japaner }

Zigarettenmarke: { Old-Gold, Kools, Chesterfield, Lucky-Strike, Parliaments }

Die Constraints, können wie folgt definiert werden:

$$\forall i \in \{1, ..., 5\}$$

Binäre Constraints

$$(\text{Nationalität\_}H_i = \text{Engländer} \Rightarrow \text{Farbe\_}H_i = \text{rot})$$

$$(\text{Nationalität\_}H_i = \text{Spanier} \Rightarrow \text{Haustier\_}H_i = \text{Hund})$$

$$(\text{Nationalität\_}H_i = \text{Ukrainer} \Rightarrow \text{Getränk\_}H_i = \text{Tee})$$

$$(\text{Nationalität\_}H_i = \text{Japaner} \Rightarrow \text{Zigarettenmarke\_}H_i = \text{Parliaments})$$

$$(\text{Getränk\_}H_i = \text{Kaffee} \Rightarrow \text{Farbe\_}H_i = \text{grün})$$

$$(\text{Farbe\_}H_i = \text{weiß} \Rightarrow \text{Farbe\_}H_{i+1} = \text{grün})$$

$$(\text{Zigarettenmarke\_}H_i = \text{Old-Gold} \Rightarrow \text{Haustier\_}H_i = \text{Schnecke})$$

$$(\text{Zigarettenmarke\_}H_i = \text{Kools} \Rightarrow \text{Farbe\_}H_i = \text{gelb})$$

$$(\text{Zigarettenmarke\_}H_i = \text{Kools} \Rightarrow \text{Haustier\_}H_{i-1} \vee H_{i+1} = \text{Pferd})$$

$$(\text{Zigarettenmarke\_}H_i = \text{Chesterfield} \Rightarrow \text{Haustier\_}H_{i-1} \vee H_{i+1} = \text{Fuchs})$$

$$(\text{Zigarettenmarke\_}H_i = \text{Lucky-Strike} \Rightarrow \text{Getränk\_}H_i = \text{Orangensaft})$$

Unäre Constraints

(Nationalität_H_i = Norweger $\Rightarrow$ H = H_1)

(Nationalität_H_i = Norweger $\Rightarrow$ Farbe_H_i $\neq$ rot)

(Nationalität_H_i = Norweger $\Rightarrow$ Haustier_H_i $\neq$ Hund)

(Nationalität_H_i = Norweger $\Rightarrow$ Getränk_H_i $\neq$ Tee)

(Nationalität_H_i = Norweger $\Rightarrow$ Zigarettenmarke_H_i $\neq$ Parliaments)

(Nationalität_H_i = Norweger $\Rightarrow$ Farbe_H_{i+1} = blau)

(Nationalität_H_i = Norweger $\Rightarrow$ Farbe_H_1 $\neq$ grün)

(Getränk_H_i = Milch $\Rightarrow$ H = H_3)

(Getränk_H_i = Milch $\Rightarrow$ Farbe_H_i $\neq$ grün)

(Getränk_H_i = Milch $\Rightarrow$ Nationalität_H_i $\neq$ Ukrainer)

(Getränk_H_i = Milch $\Rightarrow$ Zigarettenmarke_H_i $\neq$ Lucky-Strike)

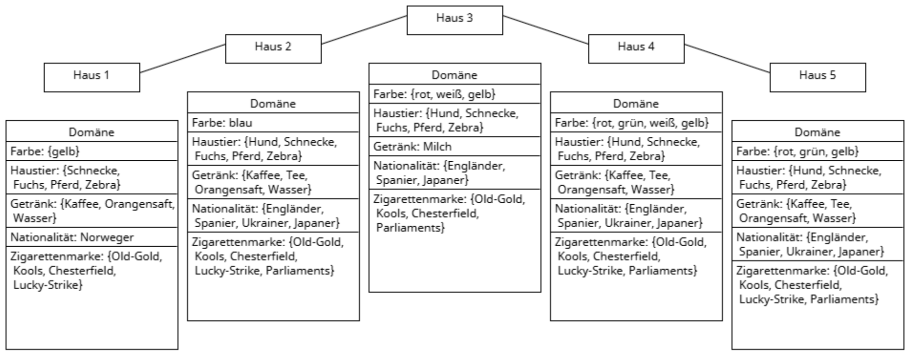
(Farbe_H_i = weiß $\Rightarrow$ H $\neq$ H_5)

(Farbe_H_i = weiß $\Rightarrow$ H $\neq$ H_1)

CSP.02: Framework für Constraint Satisfaction

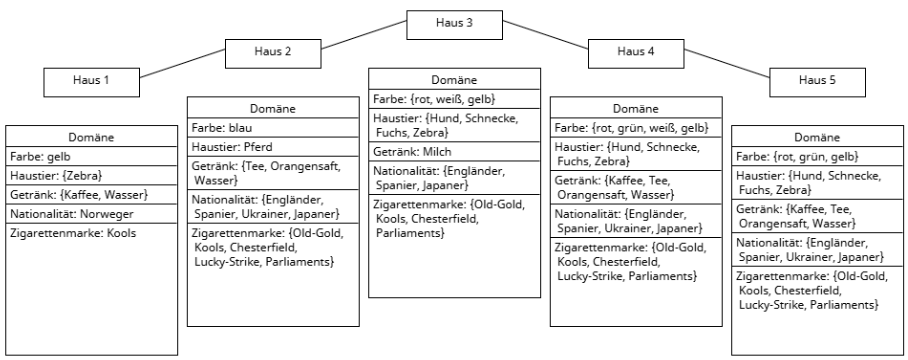
BT_Search

Die unären Constraints werden vor Durchführung, bereits angewendet, sodass die Ausgangslage, wie folgt aussieht:

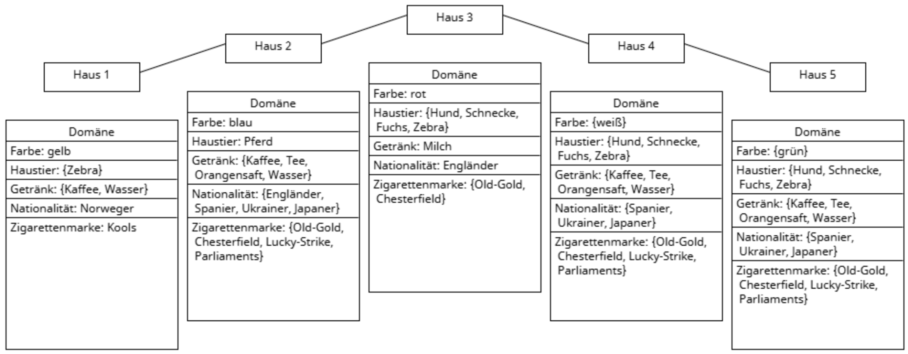


Für die erste Zuweisung kann jetzt, da in diesem Durchlauf keine Heuristik vorhanden ist, Farbe_H_1 = gelb, zugewiesen werden. Dadurch wird aufgrund der Binären Constraints, Zigarettenmarke_H_i = Kools und darauf folgend Haustier_$H_{i-1} \vee H_{i+1}$ = Pferd, ausgelöst. Zudem z.B., da nun die Zigarettenmarke bestimmt ist, wird durch die Inferenz gesehen, dass aufgrund von (Zigarettenmarke_H_i = Lucky-Strike $\Rightarrow$ Getränk_H_i = Orangensaft), Orangensaft aus der Variablen von Haus 1 gelöscht wird.

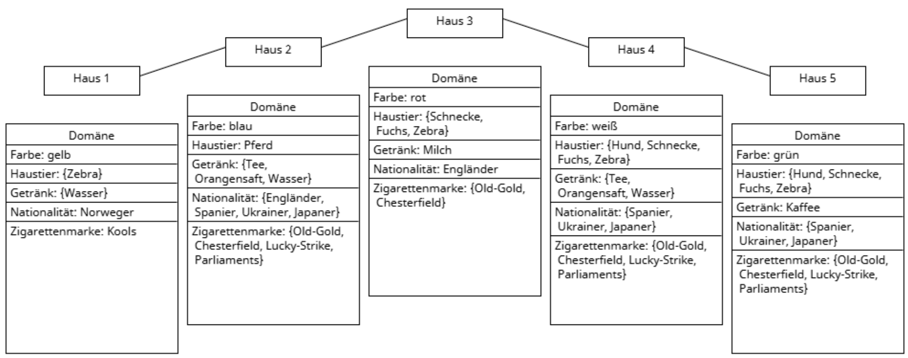
Das Resultat sieht, wie folgt aus:



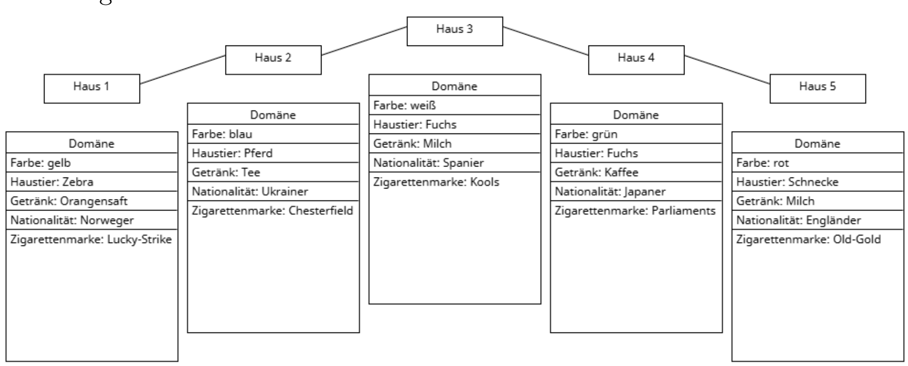
Wenn man dann Farbe_H_3 = rot zuweist, sieht es, wie folgt aus:



Nach Farbe_H_4 = weiß, sieht es, wie folgt aus:



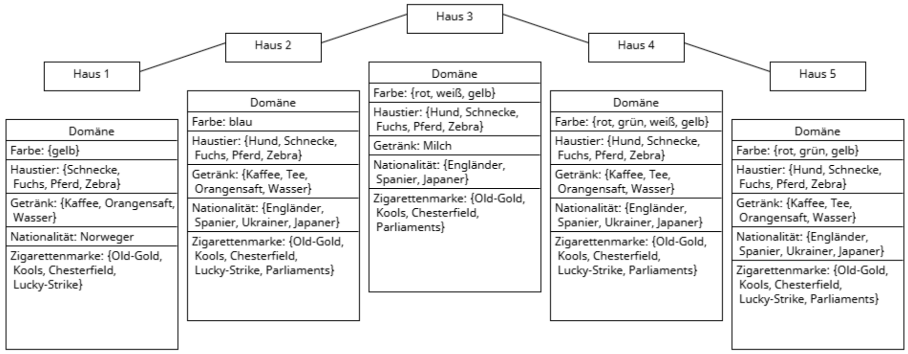
Wenn man hiernach so weitermacht, werden bei dem 7. Aufruf Probleme aufkommen, wo dann backtracking angewendet werden muss, sodass am ende, folgendes Ergebnis herauskommt:



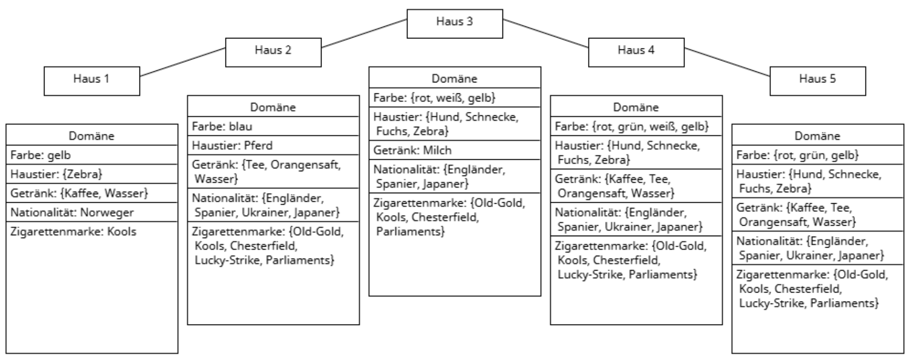
MRV und Gradheuristik

In diesem neuem Durchlauf werden die Variablen anhand der MRV und Gradheuristik entschieden, welche immer erst die Variable mit der kleinsten Domäne wählen und im falle, dass es mehrere zur Auswahl gibt, entschiedet die Gradheuristik, anhand der Anzahl der, durch Constraints Verbindete andere Objekte und wählt die größte Menge.

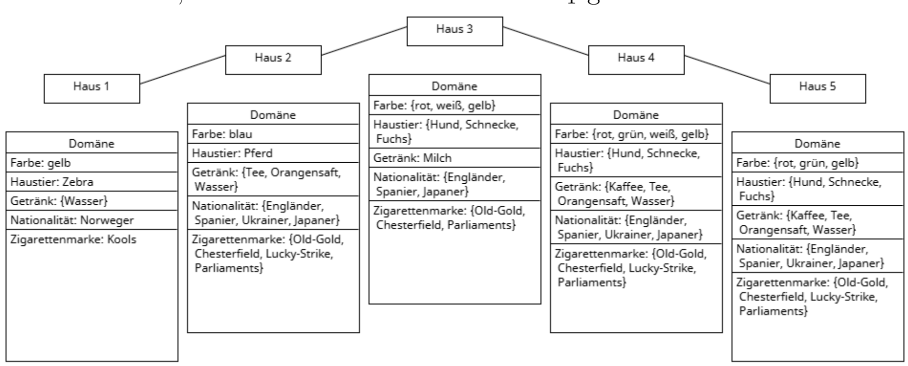
Der ausgang ist der gleiche, wie zuvor:



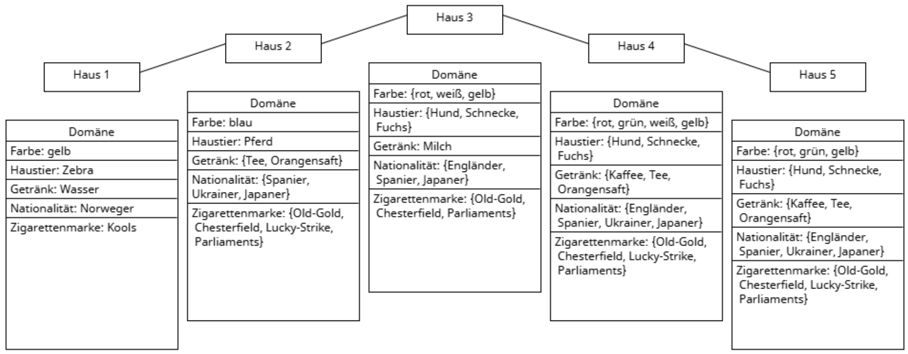
Zuerst wird hier Farbe_H_1 = gelb gewählt.



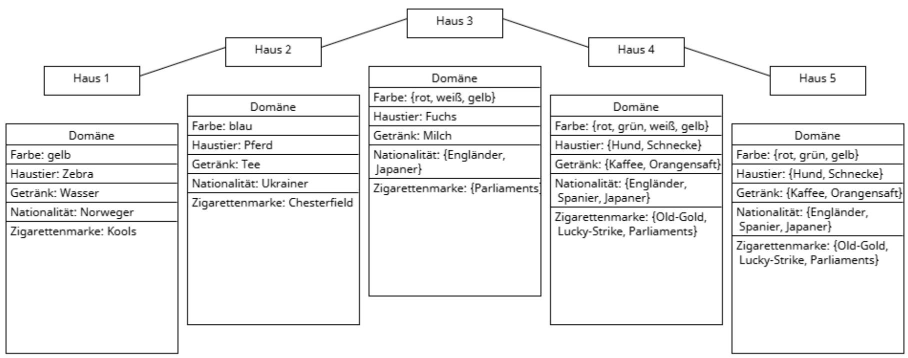
MRV liefert hier Haustier_H_1 und Getränk_H_1 zurück, welche bei de jedoch nicht in einem Constrint erwähnt werden, weshalb die GRdaheuristik für beide eine 0 zurückliefert, weshalb hier einfach Haustier_H_1 gewählt wird.



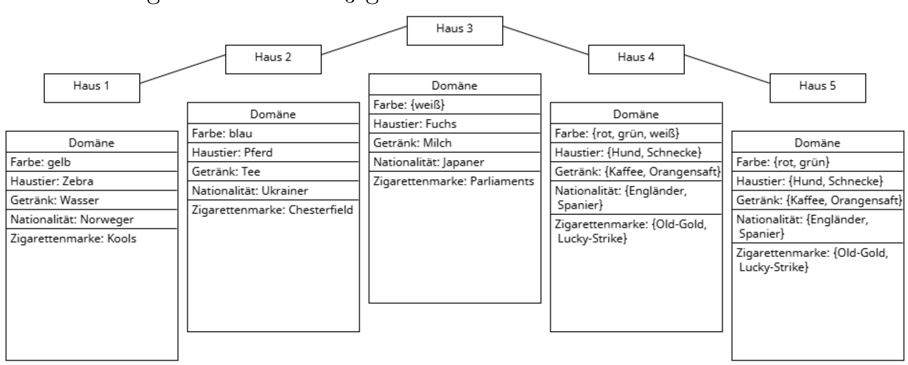
Jetzt wird Getränk_H_1 gewählt.



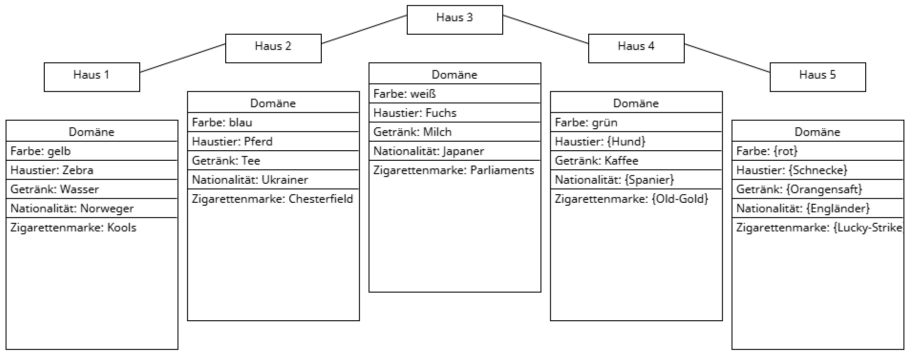
Jetzt wählt MRV Getränk_H_2



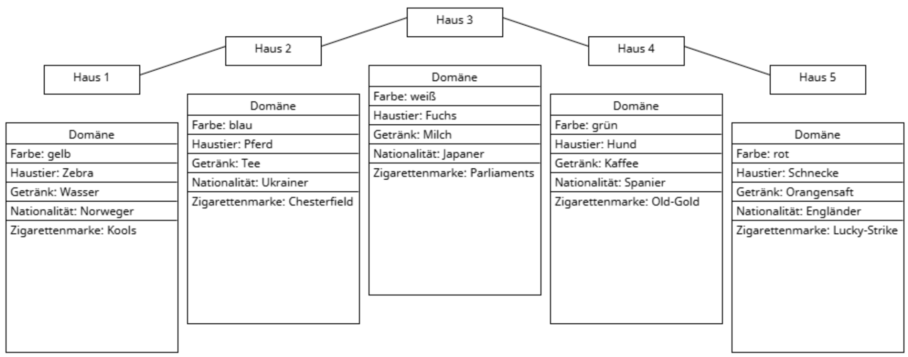
Jetzt wird Zigarettenmarkt_H_3 gewählt



Jetzt wird Farbe_H_3 gewählt



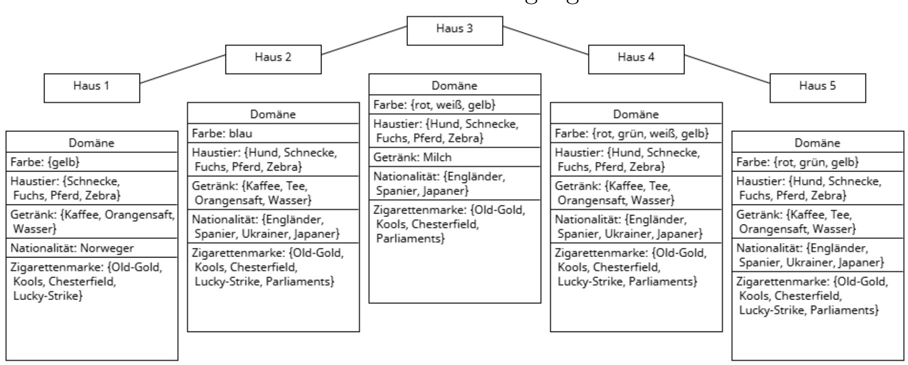
Dies lässt sich jetzt ausfüllen, um folgendes Ergebnis zu bekommen:



Die Durchführung des Algorithmus BT_Search war deutlich schneller und einfacher mit der Heuristik von MRV und der Gradheuristik und hat zu einem Ergebnis schneller und ohne backtracking geführt.

0.0.1 AC-3

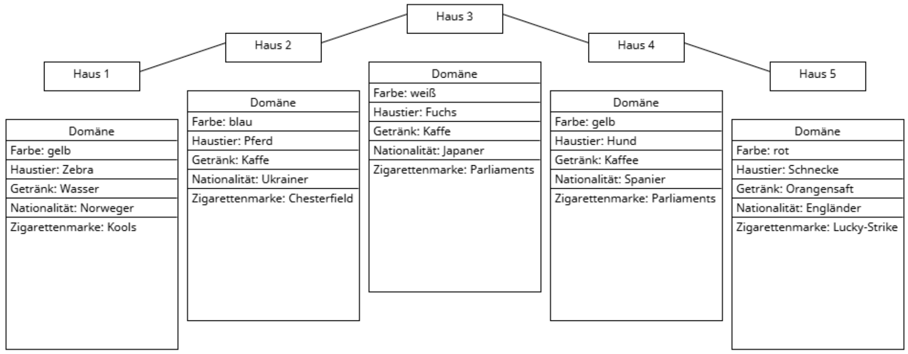
Wenn man AC-3 zuerst durchführt sieht der Ausgangszustand so aus:



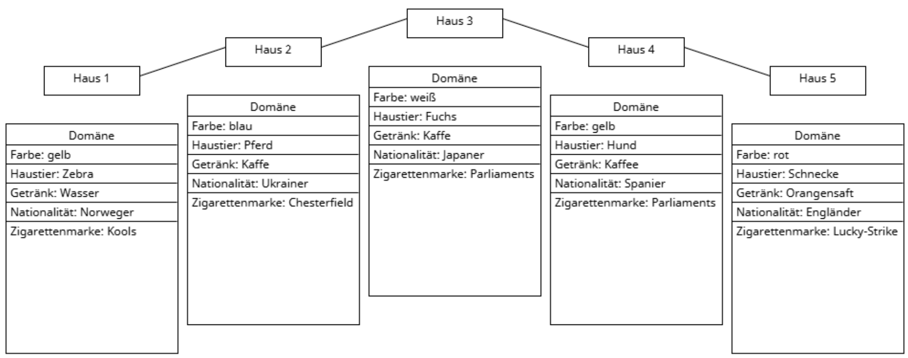
Das Suchverfahren hiermit verlief gleich schnell, wie das, des vorherigen Durchlaufs, weil beide kein backtracking betreiben mussten.

Min-Conflict

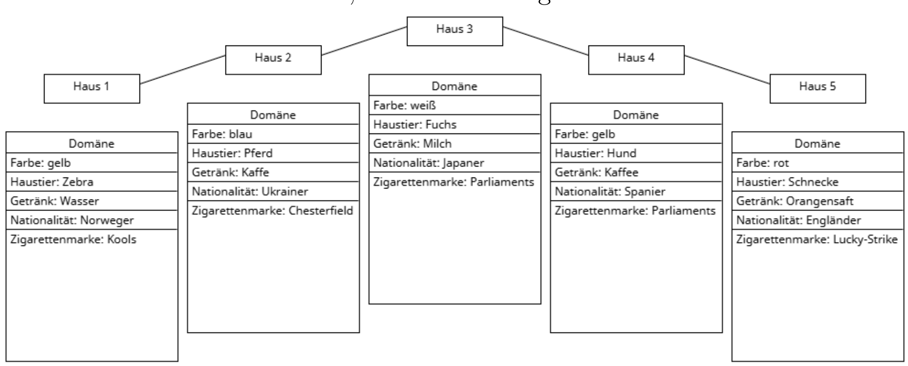
Für min-Conflicts verwenden wir hier folgende ausgangslage:



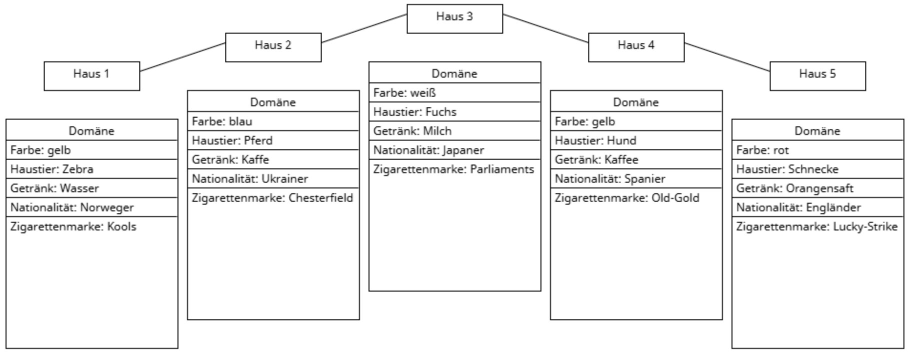
Als erste variable, welche im Konflikt steht, wählen wir Farbe_H_1. Hier ist es jedoch, dass jede andere Farbe, zwei Konflikte (mit Zigarettenmarke und Farbe/grün mit Getränk) aufrufen würde, wobei die Farbe gelb nur einen Konflikt (mit Farbe) hat. Hier wird jetzt also gelb erneut für Farbe_H_1 ausgewählt und der Zustand des CSP bleibt.



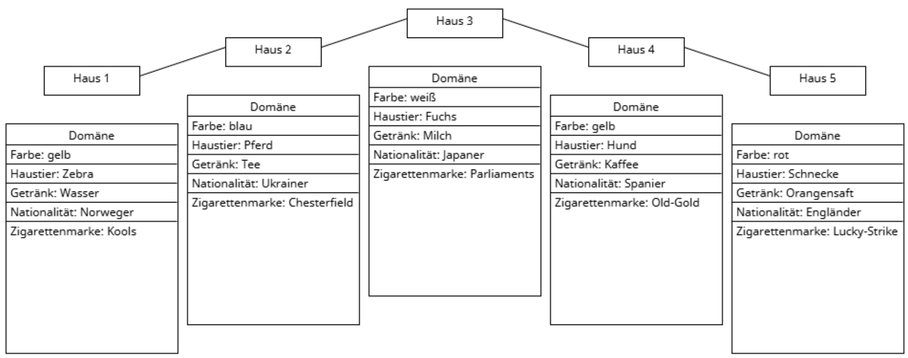
Zunächst wählen wir Getränk_H_3, welches drei Konflikte aufweist. Hier wirft die Auswahl "Milch" für das Getränk keine Konflikte auf, wo alle anderen Getränke mindestens einen haben, weshalb Milch gewählt wird.



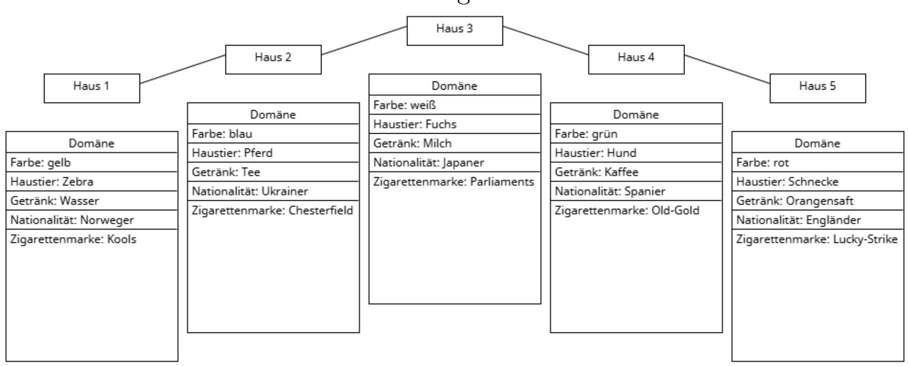
Danach wird Zigarettenmarke_H_4 gewählt, welches zwei Konflikte aufwirft. Der Wert, der keine Konflikte aufwirft ist "Old-Gold", welche gewählt wird.



Dann wird Getränk_H_2 gewählt, welche zwei Konflikte aufwirft. Der Wert "Tee" hat keine Konflikte und wird ausgewählt.



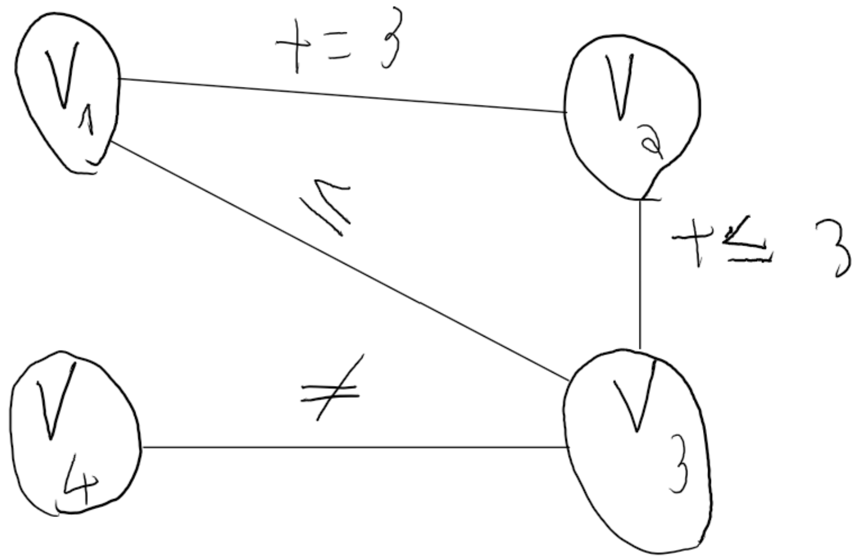
Zuletzt wird Farbe_H_4 gewählt, welches zwei Konflikte aufwirft. der Wert "grün" wirkt keine Konflikte auf und wird gewählt.



Die Laufzeit dieses Vorgehens ist schwer mit denen, der vorherigen zu vergleichen, weil dieses Verfahren mit einem Zufälligem Anfangszustand bei jedem Durchlauf beginnt, wodurch die Laufzeit stark variieren kann. Dafür die die Rechenzeit in einer Quellcodeimplementierung deutlich schneller in einem Durchlauf, als die anderen, aufgrund seiner simplen Implementation.

CSP.03: Kantenkonsistenz mit AC-3

Dies ist der beschriebene Constraint-Graph:



durchführung AC-3

Zu beginn, sieht die Queue, so aus:

Queue: $\{ \ v_{12}, v_{13}, v_{23}, v_{34} \ \}$

Nach der ersten Schleifendurchführung, sieht est so aus:

Queue: $\{ \ v_{13}, v_{23}, v_{34} \ \}$ *clear: true*

Hier wurden die Domänen von v_1 und v_2 auf { 1, 2} reduziert.

Nach der nächsten Durchführung, sieht es so aus:

Queue: $\{ \ v_{23}, v_{34} \ \}$ *clear: true*

Hier wurde die Domäne von v_3 um "1" reduziert.

nach der Nächsten Durchführung, sieht es so aus:

Queue: $\{ \ v_{34} \ \}$ *clear: false*

Hier hat sich keine Domäne verändert.

Nach der nächsten Durchführung, sieht es so aus:

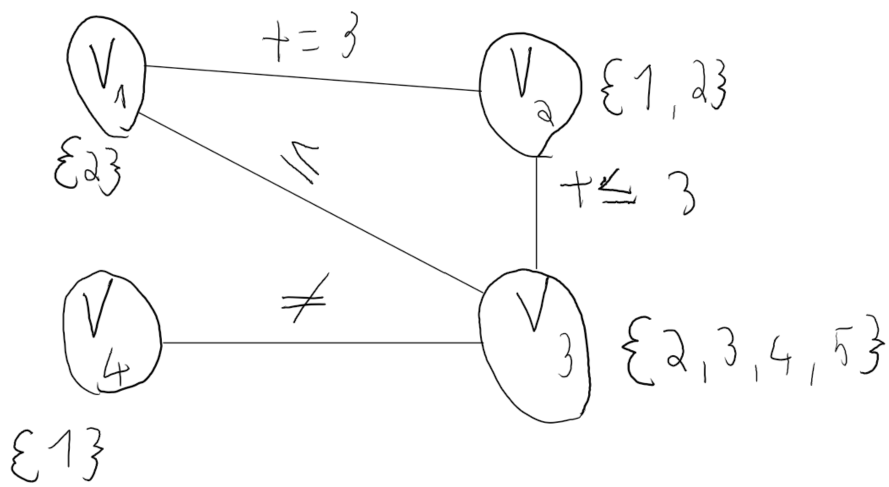
***Queue:* { }** *clear: true*

Hier wurde die Domäne von v_4 auf "1" reduziert.

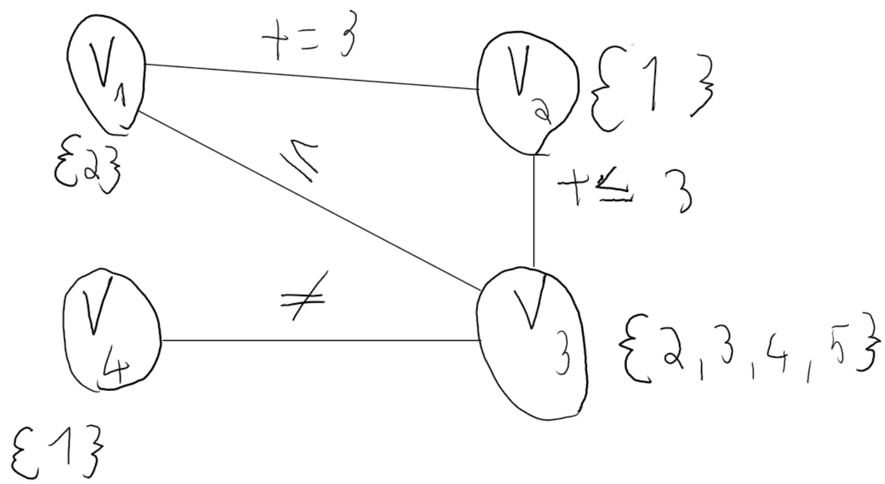
CSP.04: Forward Checking und Kantenkonsistenz

Kantenkonsistenz

Die sind hier die Ausgangswerte der Domänen:



Nachdem Kantenkonsistenz hergestellt wurde, sieht der Graph so aus:



Hier wurde die Domäne von v_2 auf "1" reduziert, damit die Kantenkonsistenz entsteht, Kantenkonsistenz mit v_3 galt bereits.

Forward-Checking (FC)

In diesem fall sieht das Ergebnis beider Verfahren gleich aus, weil die einzige Veränderung in einem Nachbar von v_1 war, weshalb FC diesen erreichen konnte. Falls z.B. eine Veränderung in v_4 gewesen wäre, hätte FC diese nicht erreichen können, die Kantenkonsistenz jedoch schon.