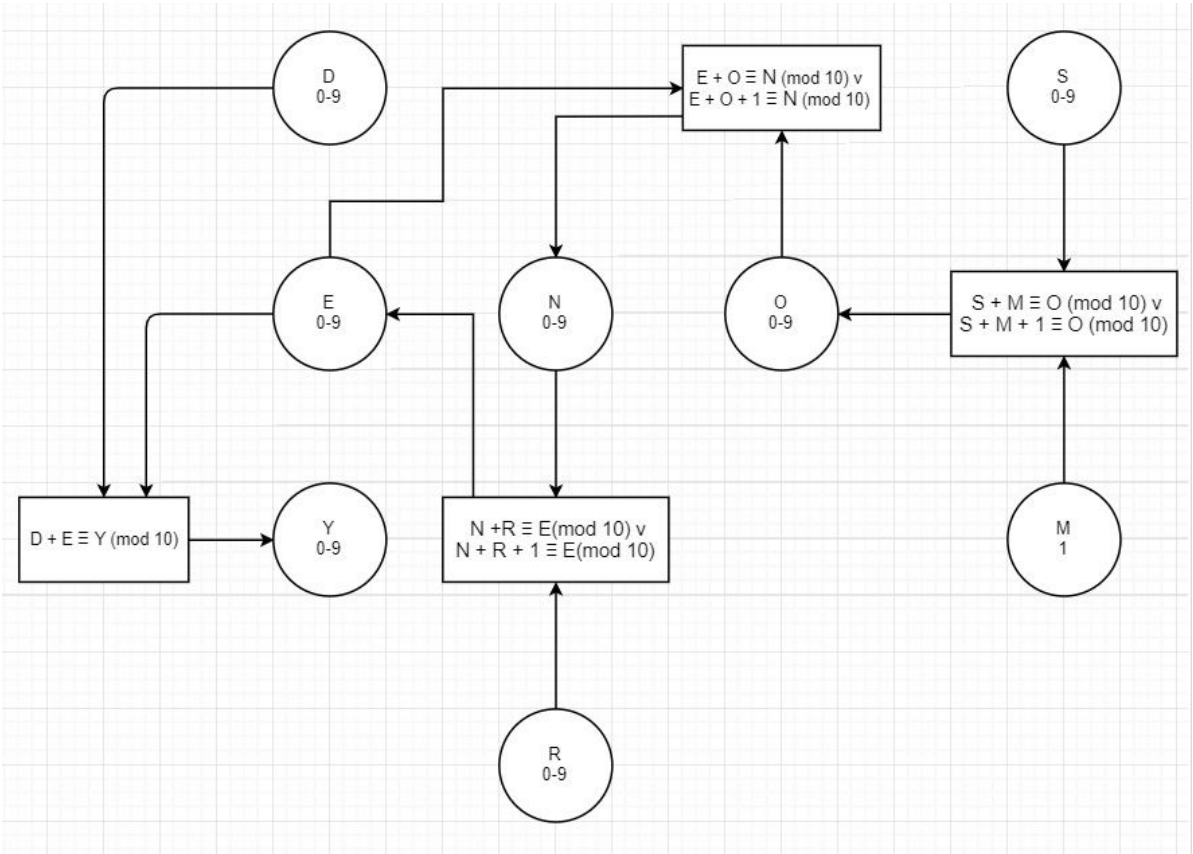
GWV – Grundlagen der Wissensverarbeitung
Tutorial 7 : Constraint Satisfaction

Enrico Milutzki (6671784), Love Kumar (7195374),
Nikolai Poets (6911432), Jan Willruth (6768273)

Exercise 7.1 : (Constraints)

1.

Jede Spalte wird durch eine Bedingung dargestellt. Die unter dem Strich stehende Zahl muss zur Summe der zu addierenden Buchstaben (ein Buchstabe ist eine Variable mit der Domäne 0-9) oder bei eventuell entstehendem Übertrag kongruent zur Summe +1 kongruent sein (modulo 10).



2.

Ohne formale Methoden oder Tools:

Wir betrachten zuerst die erste Zeile und die erste Spalte. Es kommen nur Wörter in Frage, zu deren Buchstaben jeweils Wörter in der Liste existieren, die mit diesem Buchstaben beginnen. Zudem muss es mindestens zwei verschiedene Wörter mit demselben Anfangsbuchstaben geben, die diese Anforderung erfüllen. Als mögliche Wörter verbleiben are, art, bat, bee, boa, ear, eel, eft, far, fat, tar.

Wenn wir diese Möglichkeiten durchprobieren, stellen wir fest, dass bee und boa bzw. boa und bee die einzig möglichen sind.

Wir betrachten nun die zweite Zeile und die zweite Spalte. Wir benötigen zwei Wörter, die ein a in der Mitte haben und mit e bzw. o beginnen. Es gibt nur genau zwei Wörter, auf die dies zutrifft: ear und oaf.

Als letztes betrachten wir die dritte Zeile und die dritte Spalte. Wir benötigen zwei Wörter, die mit ar bzw. ef beginnen und auf denselben Buchstaben enden. Es gibt nur genau zwei Wörter, auf die dies zutrifft: art und eft.

Hiermit haben wir das Kreuzworträtsel gelöst und die zwei Lösungen ([bee, oaf, art] und [boa, ear, eft]), die sich gegenseitig durch eine Transposition der Matrix ergeben, gefunden.

Mit domain consistency und generalized arc consistency algorithm:

Zu Anfang besteht die Domäne jeder Variable, wobei jedes Feld der 3x3 Matrix eine Variable darstellt, aus der Menge aller in der Wortliste vorkommenden Buchstaben:
{a, b, c, d, e, f, g, h, i, k, l, m, n, o, p, r, s, t, u, w, y}

Zunächst können wir unsere domain consistency constraints erstellen. Für A1D3, A1D2, A1D1, A2D1 und A3D1 kommen nur Buchstaben in Frage, zu denen es ein Wort mit demselben Anfangsbuchstaben gibt.
Analog lassen sich die constraints für A2D1, A2D2, A2D3, A1D2 und A3D2 und A3D1, A3D2, A3D3, A2D3 und A1D3 bilden für die nur Buchstaben in Frage kommen, zu denen es ein Wort mit demselben mittleren bzw. Endbuchstaben gibt.
Nach Anwendung dieser constraints, sehen unsere Domänen folgendermaßen aus:

x	D1	D2	D3
A1	{a, b, e, f, l, o, r, t}	{a, e, f, o, r}	{a, e, f, l, o, r, t}
A2	{a, e, f, o, r}	{a, d, e, f, g, i, n, o, p, r, s, u, w, y}	{a, d, e, f, g, n, o, r, y}
A3	{a, e, f, l, o, r, t}	{a, d, e, f, g, n, o, r}	{a, c, d, e, f, g, h, k, l, m, n, o, r, t, y}

Nun können wir unsere arc consistency constraints erstellen. Wir erstellen constraints dafür, dass jede Zeile und jede Spalte untereinander paarweise verschieden sein müssen, da jedes Wort nur einmal vorkommen soll. Anschließend erstellen wir constraints dafür, dass diese entstehenden Wörter in unserer Wortliste enthalten sein müssen.

Auf der Basis dieser constraints können wir den GAC Algorithmus anwenden, sodass anfänglich alle arcs markiert sind und diese nacheinander abgearbeitet werden, wobei die Markierung entfernt wird, sobald es bearbeitet wurde und anschließend alle von dem constraint betroffenen arcs wieder zur Bearbeitung markiert werden.

Durch Anwendung dieser Prozesse erhalten wir letztendlich die Lösung:
A1 & D1: {bee, boa}, A2 & D2: {ear, oaf}, A3 & D3: {art, eft}.

3. (puzzle.py)

Die Implementation probiert einfach nur alle Möglichkeiten durch ohne GAC zu verwenden… 1 Mitleidspunkt?