Universität Augsburg, LSt. Softwaretechnik, A. Schiendorfer, O. Kosak

Selbst-organisierende, adaptive Systeme (WS 16/17)

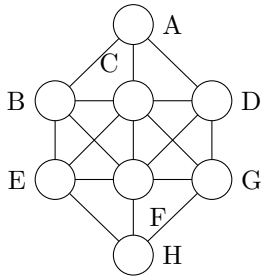
Übungsblatt 12 (Bearbeitung bis: 25.01.2017, 23:59 Uhr)

Constraint Satisfaction Probleme

Aufgabe 1 *(Constraint Propagation)*

In dieser Aufgabe machen Sie sich mit Techniken des Constraint-Programming vertraut.

a) Tragen Sie die Zahlen von 1 bis 8 der Reihe nach in die Kreise ein. Dabei dürfen aufeinanderfolgende Zahlen wie 2 und 3 nicht in Kreisen stehen, die direkt miteinander verbunden sind. Wählen Sie eine geeignete CSP-Modellierung, die es Ihnen erlaubt, möglichst viele Werte direkt zu schließen und möglichst wenig zu probieren. Dokumentieren Sie Ihre Schritte zur Lösung.



b) Nachdem Sie händisch das Puzzle gelöst haben, versuchen Sie sich nun daran, ein ausführbares Constraint-Modell dafür zu erstellen. Laden Sie sich die MiniZinc-IDE[1] herunter und schreiben Sie Ihr Modell aus der ersten Aufgabe als MiniZinc-Code. Konsultieren Sie bei Bedarf das Tutorial.

c) Die *Arität* eines Constraints gibt an, wie viele Variablen im Scope liegen. Ein unärer Constraint wäre also $x < 5$, ein binärer $x < y - 5$ etc. Zeigen Sie, wie man ternäre Constraints (z.B. $a + b = c$) mittles Hilfsvariablen in binäre Constraints umwandeln kann (*Hinweis:* nehmen Sie Hilfsvariablen, die Paare aus den Domänen anderer Variablen als Domäne haben und Constraints wie `elem(x,y,1)` für "x ist das erste Element von y). Verallgemeinern Sie nun, wie sich $n + 1$-äre Constraints als n-äre darstellen lassen, um somit zu rechtfertigen, dass sich manche Constraint-Solver auf binäre Constraints konzentrieren.

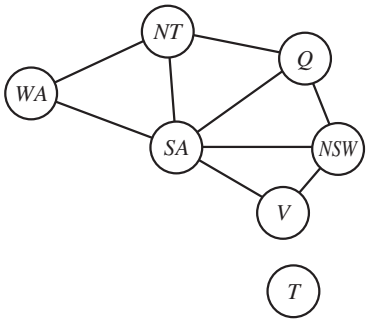
d) Die Tatsache, dass sich CSPs in binäre überführen lassen, führt zu der übersichtlichen Darstellung des Constraint-Graphen in Abbildung 1b. Führen Sie per Hand Constraint-Propagation aus, um herauszufinden, dass die partielle Zuweisung $\{\mathsf{WA} \leftarrow \text{red}, \mathsf{V} \leftarrow \text{blue}\}$ in dem Landkartenfärbeproblem Australiens zu keiner gültigen Lösung führen kann.

Aufgabe 2 *(Sudoku mit Constraints)*

In dieser Übungsaufgabe vergleichen Sie die Effizienz einer naiven Backtracking-Implementierung zur Lösung von Sudokus mit einer constraint-basierten Variante.
Basisdatentypen und ein rudimentärer Backtracking-Algorithmus sind für Sie in der Datei `ConstraintsSudoku.zip` als Eclipse-Java-Projekt vorbereitet. Machen Sie sich mit dem Code vertraut, welcher in einem Testfall `test/isse.SudokuTest` Probleme in drei Schwierigkeitsgraden im sdk-Format[2] einliest und Löser instanziiert.

[1] `http://www.minizinc.org/`
[2] siehe `http://www.sudocue.net/download.php`

(a) Politische Landkarte



(b) Modellierung als *Constraint*-Graph mit Variablen als Knoten und Constraints als Kanten

Abbildung 1: Landkartenfärbeproblem Australiens [1]

a) Modellieren Sie das Finden einer Lösung zu einem Sudoku-Rätsel als Constraint-Satisfaction-Problem mittels geeigneter Variablen, Domänen und Constraints.

b) Implementieren Sie dafür einen Constraint-Löser (Klasse `src/isse.ConstraintSolver`). Aus diesem ergeben sich die benötigten Datentypen für Domänen und Variablen. Entwickeln Sie geeignete Constraint-Typen und Propagationsalgorithmen. Orientieren Sie sich bei der Architektur an den Folien 18 und 19 und dokumentieren Sie Ihre Implementierungsentscheidungen mit geeigneten Mitteln (UML, Code-Kommentierung mit Vorbedingungen und Nachbedingungen).

c) Versuchen Sie, möglichst viele Werte direkt über Propagation zu ermitteln. Sollte dennoch ein „echter“ Suchschritt nötig sein, wählen Sie eine geeignete Heuristik für Variablen- und Wertreihenfolgen.

d) Analysieren Sie die Beschleunigung, die Sie erzielen konnten, indem Sie die prozentuale Beschleunigung und Reduktion der rekursiven Aufrufe über alle Probleme im Ordner `Pack1` messen. Dokumentieren Sie die unabhängigen und abhängigen Variablen und präsentieren Sie die Daten in grafischer Form.

Literatur

[1] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2010.