Prof. Dr. Manfred Schmidt-Schauß, David Sabel
Prof. Dr. Ingo Timm, Andreas Lattner
Fachbereich Informatik und Mathematik/ Institut für Informatik
Johann Wolfgang Goethe-Universität Frankfurt am Main

Einführung in die Methoden der Künstlichen Intelligenz

Sommersemester 2007

Aufgabenblatt Nr. 2

Abgabe: Dienstag, 22. Mai 2007 **vor!** der Vorlesung

Aufgabe 1 (25 Punkte)

Erstellen Sie mit Hilfe von Constraint-Satisfaction-Techniken ein Kreuzworträtsel. Ein Kreuzworträtsel ist eine Anordnung von Worten in einem Gitter, wobei in jeder Zelle ein Buchstabe steht, wenn diese nicht durch ein schwarzes Kästchen geblockt ist. Worte dürfen dabei von links nach rechts und von oben nach unten angeordnet werden. Weiterhin darf kein Wort im Rätsel doppelt erscheinen. Benutzen Sie ein Gitter mit 3 x 3 Zellen und 2 geblockten Zellen wie in der Abbildung dargestellt. Die leeren Zellen werden von 1 bis 7 nummeriert.

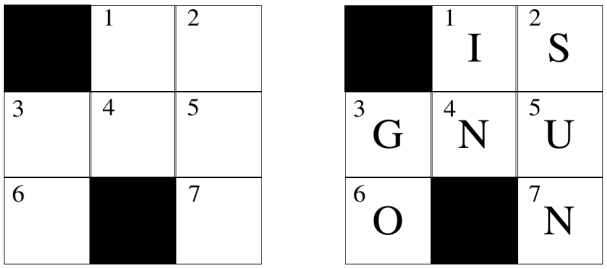

Abbildung 1: Kreuzworträtsel

Die komplette Wortliste (alphabetisch sortiert), die Sie benutzen sollen, ist: ANT, AS, AT, GNU, GO, GPL, IN, IS, PI, STL, SUN.

a) Spezifizieren Sie die Domänen für jede Variable in diesem Constraint Satisfaction Problem (CSP). Es gibt fünf Variablen, die wir mit: 1L – das Wort welches an Position 1 von links nach rechts beginnt (im Beispiel IS), 1O – das Wort welches an Position 1 von oben nach unten beginnt (IN), 2O, 3L und 3O. Ordnen Sie die Worte in den Domänen alphabetisch.

   1L : IN, ...
   1O : ...
   2O : ...
   3L : ...
   3O : ...

b) Schreiben Sie die Constraints auf, welche die Variabelenpaare einschränken. Benutzen Sie folgende Notation, um einen bestimmten Buchstaben eines Wortes zu markieren: Variablenname(Buchstabenposition), z.B. 1O(2) bezeichnet den zweiten Buchstaben des Wortes

1O. Um auszudrücken, dass der erste Buchstabe des Wortes 1O derselbe sein muss, wie der zweite Buchstabe des Wortes 3L schreiben Sie: 1O(1) = 3L(2). Bitte denken Sie auch daran, die Constraints bzgl. der nicht erlaubten doppelten Benutzung von Wörtern aufzuschreiben.

c) Zeichnen Sie den Constraint-Graphen für das Problem.

d) Simulieren Sie die Schritte, die notwendig sind, ein Kreuzworträtsel aus der obigen Wortliste durch Constraint Propagation zu lösen. Schränken Sie zunächst die möglichen Werte durch Kantenkonsistenz ein und dokumentieren Sie die einzelnen Schritte in einer Tabelle (siehe Beispiel unten). Bitte schreiben Sie in die Spalte Kommentar genau den Schritt, den der Algorithmus gerade ausführt. Führen Sie nach diesem Schritt ein einfaches Backtracking durch, um eine Lösung zu finden und zeichnen Sie den zugehörigen Suchbaum (der eingeschränkten Wertmengen für die einzelnen Variablen). Die Wertezuweisung soll dabei in alphabetischer Reihenfolge (wie oben angeführt) erfolgen.

Schritt	1L	1O	2O	3L	3O	Zu prüfende Constraints	Kommentar
1							
2							
...							

Aufgabe 2 (25 Punkte)

Wir betrachten nun das Spiel "Vier gewinnt". Bei diesem Spiel besteht das Feld in der Originalversion aus sieben Spalten und sechs Reihen (siehe Bild unten). Zwei Spieler werfen abwechselnd Spielchips ihrer jeweiligen Farbe in eine Spalte. Die Spielchips landen dann in der Regel dank der Schwerkraft in dem untersten freien Feld. Es ist nicht erlaubt, mehr als sechs Steine in eine Spalte zu werfen. Gewonnen hat der Spieler, der als erstes vier seiner Spielsteine waagerecht, senkrecht oder diagonal direkt nebeneinander anordnen konnte. Wenn das komplette Feld mit Spielsteinen gefüllt ist, jedoch keiner der Spieler vier Steine in einer Reihe hat, endet das Spiel unentschieden.

a) Denken Sie sich eine sinnvolle Evaluierungsfunktion aus, mit der Spielsituationen bewertet werden können und beschreiben Sie diese. Stellen Sie sicher, dass die Funktion Werte in dem Intervall [-1, 1] berechnet. Je kleiner (größer) die Werte, desto wahrscheinlicher ist eine Sieg für Spieler 2 (Spieler 1). Die utility-Funktion gibt entsprechend 1, -1 oder 0 zurück, wenn Spieler 1 oder Spieler 2 gewinnt, bzw. ein Unentschieden vorliegt.

b) Implementieren Sie einen Computerspieler, der mit Hilfe des Minimax-Algorithmus seinen nächsten Zug für ein "Vier gewinnt"-Spiel auswählt. Sehen Sie in den zugehörigen Methoden vor, dass ein Tiefenlimit angegeben werden kann. Ist dieses Limit erreicht, muss die Suche abgebrochen werden und Ihre Evaluierungsfunktion den Ausgang des Spiels abschätzen.

Hinweis: Für diese Aufgabe wird ein Programmgerüst bereitgestellt, was Sie benutzen können (aber nicht müssen). Hier sind bereits die wesentlichen Funktionalitäten zum "Vier gewinnt"-Spielen enthalten. Sie müssen jedoch noch für den Computerspieler eine Evaluierungsfunktion und die Minimax-Suche implementieren. Leiten Sie von den vorgegebenen Klassen eigene Klassen

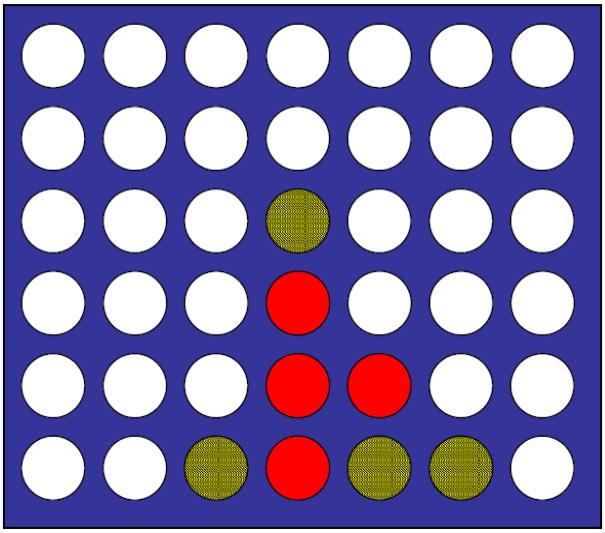

Abbildung 2: Vier gewinnt

ab und geben sie jeweils einen eindeutigen Namen (z.B. mit Ihrem Namen: MustermannComputerPlayer.java, MustermannMiniMaxSearch.java, MustermannFourInARowState.java). Dies ist wichtig, um verschiedene Implementierungen auch gegeneinander Spielen lassen zu können. Dummy-Implementierungen können den jeweiligen Dummy-Klassen entnommen werden (analog zu jeder Dummy-Klasse müssen Sie eine eigene Klasse implementieren). Das Paket mit dem Programmgerüst kann auf der KI-Homepage heruntergeladen werden: `http://www.informatik.uni-frankfurt.de/~bachki/SS2007/aufgaben/fourinarow.tgz`. Weitere Informationen zu den bereitgestellten Sourcen werden in den Tutorien gegeben.