Praktikum zur Vorlesung

Angewandte Logik

Prof. Torsten Schaub, Anita Dieckhoff, Daniel Schurtzmann, Martin Gebser, Robert Engelmann, Sven Papenfuß
Universität Potsdam (Wissensverarbeitung und Informationssysteme) — Sommersemester 2013

Praktikumsaufgabe 3 (Creek)

Bei dem Puzzle "Creek" besteht die Aufgabe darin, einige Felder eines rechteckigen Gitters so zu schwärzen, dass die folgenden Bedingungen erfüllt sind:

1. Alle nicht geschwärzten Felder müssen untereinander verbunden sein, wobei horizontal oder vertikal benachbarte nicht geschwärzte Felder direkt verbunden sind.
2. Die für einige Knotenpunkte des Gitters vorgegebenen Nummern von angrenzenden zu schwärzenden Feldern müssen eingehalten sein.

Creek-Puzzles und Lösungen können z.B. wie folgt aussehen:

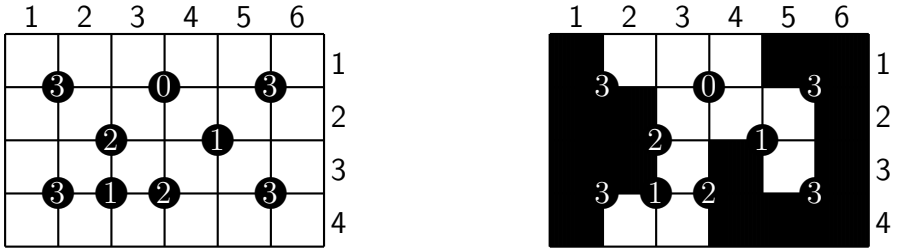

Abbildung 1: Ein Creek-Puzzle (links) und eine Lösung (rechts).

Ein Beispielpuzzle ist in Abbildung 1 dargestellt. Die linke Seite zeigt ein 6x4-Gitter mit vorgegebenen Nummern in neun Knotenpunkten. Auf der rechten Seite sind zudem die geschwärzten Felder einer (hier der einzigen) Lösung für das Gitter eingezeichnet. Beobachten Sie, dass die Anzahl der an die Knotenpunkte angrenzenden geschwärzten Felder der jeweils vorgegebenen Nummer entspricht. Außerdem sind die nicht geschwärzten Felder untereinander verbunden.

Creek kann auch online ausprobiert werden. Das Puzzle (als Java-Applet) finden Sie hier:

`http://www.janko.at/Raetsel/Creek/index.htm`

Repräsentation in Prolog. Die Seitenlängen eines rechteckigen Gitters und die Knotenpunkte mit Nummern stellen wir durch Prolog-Terme der folgenden Form dar:

```
M                                   % Zeilenanzahl
N                                   % Spaltenanzahl
[c(f(x1,y1),n1),...,c(f(xk,yk),nk)] % Knotenpunkte mit Nummern
```

In der Liste von Knotenpunkten geben die Terme `c(f(xi,yi),ni)` zum einen eine Position (`xi`,`yi`) und zum anderen eine Nummer `ni` ($0 \leq$ `ni` ≤ 4) an. Das in Abbildung 1 links dargestellte 6x4-Gitter wird z.B. durch folgende Terme beschrieben:

```
M = 6
N = 4
Numbers = [c(f(1,1),3),c(f(1,3),3),c(f(2,2),2),c(f(2,3),1),
           c(f(3,1),0),c(f(3,3),2),c(f(4,2),1),c(f(5,1),3),c(f(5,3),3)]
```

Eine Lösung, d.h. eine Menge (von Koordinaten) geschwärzter Felder, sodass die oben angegebenen Bedingungen erfüllt sind, wird als Liste folgender Form repräsentiert:

```
[f(x1,y1),...,f(xj,yj)]  % Koordinaten geschwärzter Felder
```

Die in Abbildung 1 rechts dargestellte Lösung wird z.B. durch folgende Liste beschrieben:

```
[f(1,1),f(1,2),f(1,3),f(1,4),f(2,2),f(2,3),f(4,3),f(4,4),
 f(5,1),f(5,4),f(6,1),f(6,2),f(6,3),f(6,4)]
```

Da die Sortierung der Liste und eventuelle Duplikate unerheblich sind, repräsentiert auch jede anders geordnete Liste mit den obigen Elementen dieselben geschwärzten Felder.

Framework. Auf der Moodle-Seite zur Vorlesung finden Sie die Datei `creek.pl`, in der Sie das Prädikat `creek(+M,+N,+Numbers,-Blacks)` implementieren sollen, sodass die Koordinaten geschwärzter Felder in der zu berechnenden Liste `Blacks` eine Lösung für das durch `M`, `N` und `Cells` beschriebene Gitter repräsentieren. Die Datei enthält neun Testfälle in Form von Fakten des Prädikats `grid(?ID,?M,?N,?Numbers)`. Mit folgendem Aufruf können Sie das Prädikat `creek/4` z.B. auf den sechsten Testfall anwenden:

```
?- grid(6,M,N,Numbers), creek(M,N,Numbers,Blacks).
```

Die Lösungen für die vorgegebenen Testfälle können Sie Fakten des Prädikats `solution/2` entnehmen. Für jeden Testfall muss eine der (innerhalb von Listen aller Lösungen) angegebenen Mengen von Koordinaten geschwärzter Felder mit Ihrer Implementation des Prädikats `creek/4` berechenbar sein, wobei die Reihenfolge und Häufigkeit von Termen der Form `f(xi,yi)` innerhalb eines Ergebnisses beliebig sind. Ob Ihre Implementation auf den vorgegebenen Testfällen zu Lösungen führt, erkennen Sie daran, dass der Aufruf

```
?- test_all.
```

wiederholt zur Ausgabe von `success` (und nicht `failure`!) führt.

Wir erwarten, dass Ihre Implementation auf "hinreichend vielen" Testfällen in akzeptabler Zeit (maximal eine Minute je Testfall) terminiert. Wenn dieses Kriterium nicht eingehalten ist oder Testfälle falsch beantwortet werden (Ausgabe von `failure`), dann ist Ihre Implementation nicht akzeptabel. (Beim Testen auf Korrektheit verwenden wir neben den vorgegebenen Gittern auch weitere, die Ihnen vorher unbekannt sind. Auch unsere zusätzlichen Testfälle müssen von Ihrer Implementation richtig behandelt werden.) Beachten Sie, dass die Schwierigkeit von Puzzles zur Gittergröße (und den Knotenpunkten mit Nummern) korreliert. Daher verlangen wir nicht zwingend, dass Ihre Implementation auf allen Testfällen, insbesondere den größeren, innerhalb von einer Minute terminiert.

Formalitäten. Sie können die Praktikumsaufgabe in Gruppen von **bis zu drei** Studenten gemeinsam bearbeiten. Verschiedene Gruppen müssen verschiedene Lösungen einreichen. Bei Plagiaten wird die Praktikumsaufgabe für alle beteiligten Gruppen als "nicht bestanden" gewertet. Reichen Sie Ihre Lösung bitte bis zum **07.07.13** über die YETI-Plattform ein, und **registrieren Sie dort alle Ihre Gruppenmitglieder**. Achten Sie darauf, dass Sie Ihre Lösung in einer Datei mit dem Namen `creek.pl` einreichen. Nach

der Einreichung wird Ihre Implementation mit kurzer zeitlicher Verzögerung automatisch getestet. Welche Testfälle erfolgreich behandelt wurden, erkennen Sie an den Ausgaben von `success`. Sollten ein oder mehrere Testfälle mit `failure` beantwortet werden, korrigieren Sie Ihre Implementation bitte umgehend selbstständig (bzw. in Rücksprache mit uns, falls Ihnen keiner der falsch behandelten Testfälle vorliegt).

Bonus: Effiziente Lösungen werden mit **bis zu 2 Bonuspunkten auf die Klausur** honoriert, wobei die Effizienz relativ bzgl. der rechtzeitig in YETI eingereichten Implementationen ermittelt wird.

Empfehlungen und Hinweise:

- Überlegen Sie sich zunächst eine Strategie, mit der Sie die Aufgabe lösen wollen, und identifizieren Sie die erforderlichen Teilschritte, um die Strategie auf Prolog-Prädikate abzubilden und zu implementieren.
- Entwickeln und testen Sie Ihr Prolog-Programm inkrementell, von einfachen hin zu komplexeren Prädikaten. Jedes Basisprädikat sollte hinreichend getestet sein, bevor es für die Implementation eines darauf aufbauenden Prädikats verwendet wird.
- Verwenden Sie der jeweiligen Bedeutung entsprechende Namen für Prädikate und Variablen, und kommentieren Sie die Bedeutung außerdem in angemessenem Umfang. Neben dem höheren Wiedererkennungseffekt ist dies eine wichtige Voraussetzung, um Hilfe bei der Lösung eventueller Schwierigkeiten leisten zu können.
- Bei Fragen oder Problemen können Sie sich gern an uns wenden, z.B. in den Sprechstunden donnerstags, 14:00–16:00 Uhr, im Poolraum 3.04.1.03. Wir sind für alle Fragen offen und versuchen sie bestmöglich zu beantworten.
- Fangen Sie bald mit der Bearbeitung der Aufgabe an, damit Ihnen die Zeit nicht davonläuft. (Sollten Sie trotzdem Schwierigkeiten mit der Einhaltung des Termins haben, dann wenden Sie sich bitte an uns, anstatt eine beliebige Lösung zu kopieren!)
- Behalten Sie Ihren Quelltext für sich. Sie helfen Ihren Kommilitonen nicht, wenn Sie Ihren Quelltext weitergeben, da das Nachvollziehen eines fremden Prolog-Programms wesentlich schwieriger ist als ein eigenes zu entwerfen. (Über Herangehensweisen können Sie sich natürlich gern austauschen.)

Bei Fragen oder Problemen zu der Praktikumsaufgabe wenden Sie sich in der Folge bitte persönlich an uns (z.B. in den Sprechstunden donnerstags, 14:00–16:00 Uhr, im Poolraum 3.04.1.03) oder per Mail an:

`lp1@cs.uni-potsdam.de`