Constraint-Satisfaction-Probleme

B. Nebel, S. Wölfl
R. Mattmüller, M. Westphal
Wintersemester 2009/2010

Universität Freiburg
Institut für Informatik

Projekt P1

Abgabe: Mittwoch, 18. November 2009

Im Rahmen dieses Projektes sollen die Algorithmen AC-3 und AC-4 anhand von zufällig generierten binären Constraintnetzen verglichen werden.

> *Hinweis:* Bei den Projekten geht es um die Implementierung, nicht so sehr um theoretische Aspekte. Wenn Sie an bestimmten Stellen nicht genau wissen, wie die Algorithmen und Definitionen aus der Vorlesung in die Praxis umgesetzt werden, können Sie daher gerne Fragen stellen — entweder per E-Mail oder in der Übungsgruppe.
>
> Projekte können in C, C++, Java oder Python bearbeitet werden. Andere Programmiersprachen sind nach Absprache möglich; in diesem Fall bitte vor Bearbeitung des Projekts bei uns melden.
>
> Die eingereichten Programme müssen die geforderten Ein- und Ausgabeformate verwenden, einige Tests bestehen und **ausreichend kommentiert** sein. Programme, die diesen Anforderungen nicht genügen, werden nicht akzeptiert, aber es besteht die Möglichkeit, innerhalb der Abgabefrist nachzubessern. Daher bitten wir darum, frühzeitig abzugeben, um ausreichend Zeit für Nachbesserungen zu haben.

Aufgabe P1.1 (Zufällige Constraintnetze, 1 Punkt)

Schreiben Sie einen Generator, der binäre Constraintnetze in Abhängigkeit bestimmter Parameter zufällig erzeugt. Verwenden Sie die folgenden Parameter:

- **v** für die Anzahl der Variablen,
- **w** für die Anzahl der Werte in jeder Domäne,
- **c** für die Anzahl der Constraints und
- **d** für die Dichte der Constraints, d.h. die Anzahl der tatsächlichen Einträge geteilt durch die Anzahl der möglichen Einträge.

Die Parameter sollen dabei als Kommandozeilenargumente übergeben werden. Orientieren Sie sich bei den Ein- und Ausgaben Ihres Programms am folgenden Beispiel (beachten Sie bitte insbesondere die Sortierung der Constraints und Mengenelemente):

```
# random-network -v 4 -w 3 -c 4 -d 0.4
Variables:
V={v_0, v_1, v_2, v_3}
Domains:
D_0={V_0_0, V_0_1, V_0_2}
```

```
D_1={V_1_0, V_1_1, V_1_2}
D_2={V_2_0, V_2_1, V_2_2}
D_3={V_3_0, V_3_1, V_3_2}
Constraints:
R_0_1={(V_0_0,V_1_0), (V_0_1,V_1_0), (V_0_2,V_1_0), (V_0_2,V_1_1)}
R_0_2={(V_0_0,V_2_1), (V_0_0,V_2_2), (V_0_1,V_2_0), (V_0_2,V_2_1)}
R_1_3={(V_1_0,V_3_1), (V_1_1,V_3_1), (V_1_2,V_3_1), (V_1_2,V_3_2)}
R_2_3={(V_2_0,V_3_0), (V_2_0,V_3_1), (V_2_1,V_3_1), (V_2_2,V_3_1)}
```

Aufgabe P1.2 (Kantenkonsistenzalgorithmen, 2 Punkte)

Erweitern Sie das Programm aus der vorigen Aufgabe, indem Sie die Algorithmen AC-3 und AC-4 implementieren. Geben Sie dabei neben der ursprünglichen Ausgabe jeweils die resultierenden Domänen der (kantenkonsistenten) Ergebnisnetzwerke sowie die Kosten der Algorithmen aus.

Schätzen Sie für AC-3 die Kosten, indem Sie die Werte $|D_i| \cdot |D_j|$ für alle Aufrufe von Revise(v_i, v_j) aufsummieren. (Verwenden Sie dabei jeweils die *aktuellen reduzierten* Domänen D_i und D_j, nicht die ursprünglichen.) Addieren Sie zusätzlich die Anzahl der Constraints, um die Initialisierungskosten abzuschätzen.

Schätzen Sie für AC-4 die Kosten, indem Sie für die Initialisierung $\sum_{R \in C} |R|$ veranschlagen (da alle Constraints gleich viele Tupel enthalten, können Sie diese Summe auch direkt als Produkt berechnen). Addieren Sie zu diesen Kosten noch die Anzahl der Aufrufe von "decrement *counter*(x_j, a_j, x_i)", um die Gesamtkosten zu erhalten.

Achtung: Beachten Sie bei der Implementation von AC-4, dass jedes Variable-Wert-Paar nur einmal in die Liste eingefügt werden darf.

Beispiel für die Ein- und Ausgabe:

```
# arc-consistency -v 3 -w 3 -c 3 -d 0.4
Variables:
V={v_0, v_1, v_2}
Domains:
D_0={V_0_0, V_0_1, V_0_2}
D_1={V_1_0, V_1_1, V_1_2}
D_2={V_2_0, V_2_1, V_2_2}
Constraints:
R_0_1={(V_0_0,V_1_0), (V_0_0,V_1_1), (V_0_0,V_1_2), (V_0_1,V_1_0)}
R_0_2={(V_0_0,V_2_0), (V_0_0,V_2_1), (V_0_0,V_2_2), (V_0_2,V_2_0)}
R_1_2={(V_1_0,V_2_2), (V_1_1,V_2_0), (V_1_1,V_2_1), (V_1_1,V_2_2)}
Costs (AC-3): 49
Domains of the arc-consistent network (AC-3):
D_0={V_0_0}
D_1={V_1_0, V_1_1}
D_2={V_2_0, V_2_1, V_2_2}
Costs (AC-4): 15
Domains of the arc-consistent network (AC-4):
D_0={V_0_0}
D_1={V_1_0, V_1_1}
D_2={V_2_0, V_2_1, V_2_2}
```