Karlsruher Institut für Technologie
Institut für Theoretische Informatik
Prof. Dr. Bernhard Beckert
Alexander Weigl

Formale Systeme, WS 2017/2018

Praxisaufgabe 1: Das Nonogramme mittels SAT-Solver lösen

Abgabe der Lösungen bis zum 07.01.2018 über das ILIAS-Portal zur Vorlesung

Hinweis: Bitte beachten Sie, dass die Praxisaufgabe in Einzelarbeit und selbständig zu bearbeiten ist. Wir behalten uns vor, die selbständige Bearbeitung dadurch stichprobenartig zu prüfen, dass wir uns die eingereichte Lösung erklären lassen.
Für die vollständige Lösung dieser Praxisaufgabe erhalten Sie 10 Punkte. Die erzielten Übungspunkte werden im Verhältnis 1:10 als Bonuspunkte auf die bestandene Abschlussklausur angerechnet (max. ein Notenschritt Verbesserung).

Nonogramme

Nonogramme sind japanische NP-harte Logik-Puzzle,die Sie in dieser Praxisaufgabe mit Hilfe eines Lösers für aussagenlogische Erfüllbarkeit (*engl.* satisfiability solver, SAT solver) lösen sollen. Ein Nonogramm ist ein rechteckigen Spielbrett mit jeweils eine Liste von Zahlenangaben für jede Reihe und Spalte des Brettes.

Ziel ist es nun eine Schwarz-Weiß-Färbung zu finden. Dabei stellen die zusammenhängenden schwarzen Felder in einer Reihe und Spalte einen *Lauf* dar. Jede Zahl am Rand steht dabei für einen Lauf und dessen Länge. Folgende Bedingungen muss nun eine Färbung garantieren:

- Anzahl, Reihenfolge und Länge der Läufe in einer Reihe entsprechenden der Anzahl, Reihenfolge und Wert der Zahlenangaben zur Reihe, analog dazu die Spalten.
- Zwei Läufe sind durch mindestens ein weißes Feld voneinander getrennt.

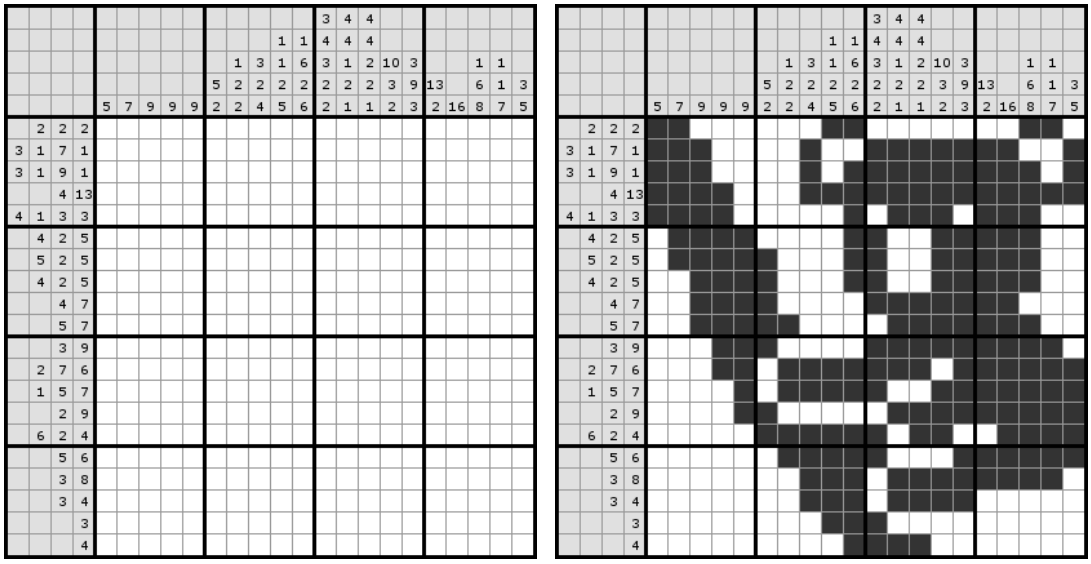
Abbildung 1 zeigt eine Instanz des Puzzles. Als Beispiel besagt die Constraint zur ersten Reihe `2 2 2`, dass sich in dieser Reihe drei Läufe befinden bestehend aus jeweils 2 schwarzen Blöcken. Weitere Informationen und Lösungsansätze, die nicht auf SAT basieren, finden Sie in der englischen Wikipedia.

Die Aufgabe

Die Praxisaufgabe besteht darin, ein Java-Programm zu schreiben, das

1. die Randbedingunen für ein Nonogramm in eine aussagenlogische Klauselmenge transformiert, so dass jede erfüllende Belegung der Klauselmenge einer Lösung des Puzzles entspricht,
2. einen vorgegebenen SAT-Solver (SAT4J) aufruft, um eine solche erfüllende Belegung zu finden,
3. die gefundene Belegung in eine Lösung des Puzzles zurückübersetzt.

Die Bedingungen müssen dazu aussagenlogisch formalisiert werden. Sie müssen entscheiden, was Sie durch AL-Variablen modellieren und die Forderungen aus der Problemstellung in Klauseln formulieren. Achten Sie bitte darauf, dass Ihre Lösung nachvollziehbar und verständlich ist.



(a) Das ursprüngliche Spielbrett

(b) Eine gültige Belegung von weißen und schwarzen Feldern

Abbildung 1: Ein Beispiel für ein Nonogram.

Lösungshinweis: Die Verwendung von zusätzlichen aussagenlogischen Variablen kann Ihnen bei der Kodierung helfen, z. B. bei der Kodierung der Startposition der Läufe.

Bitte beachten Sie: Die Klauselmenge wird von Ihrem Java-Programm für jede Instanz (jedes Spielbrett) jeweils neu erstellt. Sie kodiert sowohl die allgemeinen Regeln des Spiels als auch die Instanz-abhängigen Einschränkungen.

Rahmenwerk

Um Ihnen die Bearbeitung der Aufgabe zu erleichtern, stellen wir ein Rahmenwerk zur Verfügung, so dass Sie sich für die Bearbeitung auf die Aufgabe konzentrieren können.

Auf der Seite der Vorlesung finden Sie ein Archiv mit folgenden Dateien:

- `MyNonogramSolver.java`: Ein Skelett für Ihren Nonogram-Löser. Falls Sie weitere Klassen benötigen, speichern Sie diese ebenfalls in dieser Datei.

  Beachten Sie:

  - Ihre Implementierung von `MyNonogramSolver` muss von `edu.kit.iti.formal.NonogramSolver` erben. Dadurch erhalten Sie Zugriff auf folgende Membervariablen:

    `game : Nonogram` beinhaltet die Informationen zum aktuellen Spielbrett.

    `solution : boolean[][]` Ein initialisiertes zweidimensionales Array zum Speichern Ihrer Lösung.

    `solver : ISolver` eine Instanz des SAT-Solver (SAT4J). Eine Dokumentation des Interfaces findet sich in der SAT4J-Dokumentation

  - Der Name der Klasse `MyNonogramSolver`, sowie die Signatur der Methoden `solve()`, `getSolution()` und des Konstruktors sind fest vorgegeben, um Ihre Abgabe automatisch überprüfen zu können.

- `nonogram-1.0.jar`: Eine Java-Bibliothek (inkl. SAT4J) um:

  - Spielbrettobjekte aus String-Repräsentationen oder 1-bit Bilder zu laden (`Nonogram.from`),

– sowie einer Datenstruktur (`Nonogram`) zur Repräsentation eines Spielbrettes.

Hinweis: Indizierung der Zeile und Reihe ab 0. `solution[1][3] == true` bedeutet, dass das Feld in der zweiten Reihe, vierte Spalte schwarz ist.

Probelauf

Nach dem Entpacken des bereitgestellten Archivs sollten Sie einen Ordner mit folgenden Dateien vorfinden:

- `nonogram-1.0.jar`
- `nonograms.txt`
- `Makefile`
- `MyNonogramSolver.java`

Testen Sie Ihre Systemkonfiguration, indem Sie die Implementierung von `MyNonogramSolver` zu übersetzen versuchen:

```
javac -cp nonogram-1.0.jar MyNonogramSolver.java
```

Sie können einen Testlauf starten mit:

```
java -cp .:nonogram-1.0.jar edu.kit.iti.formal.nonogram.Runner
```

Dabei wird Ihre Implementierung auf alle Spielbretter aus der Datei `nonograms.txt` (Umgebungsvariable: `NONOGRAMS`) angewendet. Sie sollten mehrere Ausgaben folgender Art sehen:

```
I am using the nonograms in file './nonograms.txt'
== u_a ================================================
Riddle is not satisfiable
Your answer is sat
> SOLUTION u_a
-x
x-
<END SOLUTION u_a
NOT SOLVED :(
```

Weiße Spielfelder erscheinen als "x", schwarze als "-". In dieser Beispielausabge gibt das System an, dass dieses Puzzle unlösbar ist, aber die Implementierung dazu die falsche Angabe gemacht hat.

Tipp: Beginnen Sie Ihre Tests mit wenigen Spielbrettern in der Datei `nonograms.txt` und legen Sie nach und nach weitere Spielbretter dort hinein. Sollten Sie Problemfälle Ihrer Implementierung erkennen, erzeugen Sie sich für diese Fälle minimale Spielbretter, um diese Probleme isoliert zu lösen.

Kodierung der Spielbretter

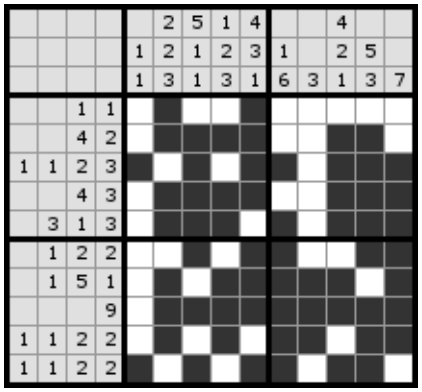
Spielbretter werden in Zeichenketten kodiert und können über das bereitgestellte Framework eingelesen werden. Die Zeichenkette ist dabei wie folgt aufgebaut. Jedes Spielbrett eine Namen ($\langle id \rangle$) gefolgt von einem Doppelpunkt. Dann kommen die Läufe für jede Zeile, danach für jede Spalte getrennt durch einen Punkt. Die Zeile bzw. Spalten sind getrennt durch die Semikolon, sowie die Läufe durch Kommata. Formal wie folgt:

$$S \to \langle id \rangle \;\; ':' \;\; \langle L \rangle^? \;\; '.' \;\; \langle L \rangle^? \;\; '!' \tag{1}$$

$$L \to \langle R^? \rangle \;\; (';' \;\; \langle R \rangle)^* \tag{2}$$

$$R \to \langle \mathbb{N} \rangle \;\; (',' \;\; \langle \mathbb{N} \rangle)^* \tag{3}$$

$$\tag{4}$$



```
1241 :1,1;4,2;1,1,2,3;4,3;3,1,3;1,2,2;1,5,1;9;1,1,2,2;1,1,2,2.
1,1;2,2,3;5,1,1;1,2,3;4,3,1;1,6;3;4,2,1;5,3;7!
```

Abbildung 2: Beispiel für eine Kodierung:

Im Archiv sind bereits einige Spielbretter mitgeliefert. Das Framework stellt Ihnen die Spielbretter als Läufe der Reihen (bzw. Spalten) als Liste von Listen von Integer (`List<List<Integer>>`) zur Verfügung.

Der SAT-Solver SAT4J

Für diese Aufgabe verwenden wir das Werkzeug SAT4J[1]. SAT4J ist eine reine Java-Implementierung eines SAT-Solvers. SAT4J wird in vielen Java-basierten Systemen eingesetzt, u.a. im Alloy Analyzer[2] oder auch im Eclipse Framework[3].

Eine Klausel wird in SAT4J als Liste von Ganzzahlen repräsentiert, dabei entsprechen positive (negative) Zahlen positiven (negativen) Literalen. Beispielsweise entspricht die aussagenlogische Formel $(\neg P_1 \vee P_2) \wedge (P_1 \vee \neg P_3 \vee \neg P_2)$ etwa folgenden Aufrufen der Methode `addClause` eines `SATSolver`-Objekts `solver`:

```
solver.addClause(new VecInt(new int[]{-1, 2}));
solver.addClause(new VecInt(new int[]{1, -3, -2}));
```

Die Erfüllbarkeit der so gesetzten Klauselmenge überprüft man mittels SAT-Solver durch Aufruf der Methode `findModel()` der Klasse `ISolver` – diese Methode liefert `null` zurück, falls die Klauselmenge unerfüllbar ist; gibt es eine erfüllende Interpretation, so wird ein Array von Ganzzahlen zurückgegeben – dabei werden diejenigen AL-Variablen, die als wahr interpretiert werden, durch eine positive ganze Zahl und diejenigen, die zu falsch ausgewertet werden, durch eine negative dargestellt. Das zurückgelieferte Modell ist partiell, nur AL-Variablen, für die der SAT-Solver eine Entscheidung getroffen hat, sind aufgeführt. Ist eine Variable nicht im Modell enthalten, ist diese beliebig belegbar.

Abgabe der Lösung

Bitte reichen Sie den Quelltext Ihres Java-Programms (*eine* Datei als ASCII-Text) bis spätestens 07.01.2018, 23:59 Uhr im **ILIAS-Portal** unter dem Punkt „Praxisblatt 1“ ein. **Auch für Programme, die nicht vollständig korrekte Ergebnisse liefern, werden Bonuspunkte anteilig vergeben.**

[1] http://www.sat4j.org/
[2] http://alloy.mit.edu/alloy4/
[3] http://mail-archive.ow2.org/sat4j-dev/2008-03/msg00001.html