

Übung zu “Automatische Analyse und Verifikation von Programmen”

Aufgabe 10 *Lösen allgemeiner Gleichungssysteme*

Gegeben sei ein Verband (L, \sqsubseteq) , der die Ascending Chain Condition erfüllt, und ein Gleichungssystem, bestehend aus n Gleichungen der Form $X_i = f_i(X_1, \dots, X_n)$ für $i \in \{1, \dots, n\}$. Dabei sind X_1, \dots, X_n Variable und f_1, \dots, f_n monotone Funktionen der Form $f_i: L^n \rightarrow L$.

Eine Funktion $f: L^n \rightarrow L$ heißt *unabhängig* vom i -ten Parameter, falls $f(l_1, \dots, l_i, \dots, l_n) = f(l_1, \dots, l'_i, \dots, l_n)$ für alle $l_1, \dots, l_i, l'_i, \dots, l_n \in L$.

Eine Lösung des Gleichungssystems ist eine Abbildung $\sigma: \{X_1, \dots, X_n\} \rightarrow L$ mit $\sigma(X_i) = f_i(\sigma(X_1), \dots, \sigma(X_n))$ für alle $1 \leq i \leq n$.

- Beschreiben Sie, analog zum Worklist-Algorithmus, ein effizientes Verfahren, das die kleinste Lösung eines solchen Gleichungssystems berechnet.
- Wie löst man ein System von Ungleichungen der Form $X_i \sqsupseteq f_i(X_1, \dots, X_n)$?

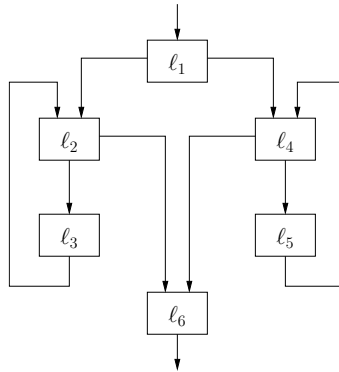
Aufgabe 11 *Varianten des Worklist-Algorithmus*

Es spielt eine große Rolle in welcher Reihenfolge der Worklist-Algorithmus die Paare der Flussrelation F abarbeitet. Es gibt in dieser Hinsicht verschiedene Optimierungsmöglichkeiten, von denen im folgenden zwei betrachtet werden.

Wir nehmen an, dass es im Flussgraph einen Block r (die Wurzel) gibt, von dem alle anderen Blöcke aus erreichbar sind. Außerdem nehmen wir an, dass der Flussgraph genau n Blöcke enthält.

- Eine Idee ist, einen gerichteten Spannbaum des Flussgraphen mit Wurzel r zu finden und dessen Knoten in *Reverse Postorder* zu ordnen, d.h., jedem Knoten bzw. Block ℓ wird eine eindeutige Zahl $n(\ell) \in \{1, \dots, n\}$ zugeordnet, so dass folgende Eigenschaft erfüllt ist:
 - Falls (ℓ, ℓ') eine *Baumkante* ist, d.h., Teil des Spannbaums ist, so gilt $n(\ell) < n(\ell')$.
 - Falls (ℓ, ℓ') eine *Vorwärtskante* ist, d.h., ℓ' ist Nachfolger von ℓ im Spannbaum, so gilt ebenfalls $n(\ell) < n(\ell')$.
 - Falls (ℓ, ℓ') eine *Rückwärtskante* ist, d.h., ℓ ist Nachfolger von ℓ' im Spannbaum, so stellen wir keine Anforderungen.
 - In allen anderen Fällen, d.h., bei sogenannten *Cross-Kanten* (ℓ, ℓ') , muss auch $n(\ell) < n(\ell')$ gelten.

- (i) Betrachten Sie den folgenden Flussgraph. Bestimmen Sie einen Spannbaum, dessen Knoten in Reverse Postorder nummeriert sind.



- (ii) Angenommen, eine solche Reverse Postorder ist gegeben. Wie sollte man dann die Worklist abarbeiten, um möglichst wenig Schritte machen zu müssen?
- (iii) Modifizieren Sie das Verfahren der Tiefensuche, um für einen gegebenen Flussgraph einen Spannbaum mit einer Nummerierung in Reverse Postorder zu erhalten.
- (b) Ein *stark zusammenhängender Subgraph* ist eine Menge von Knoten, so dass es von jedem Knoten einen Pfad zu jedem anderen Knoten gibt. Eine *starke Zusammenhangskomponente* ist ein maximaler stark zusammenhängender Subgraph.
- (i) Bestimmen Sie die starken Zusammenhangskomponenten des Graphen aus Teilaufgabe (a)(i).
- (ii) Wie kann man starke Zusammenhangskomponenten nutzen, um den Worklist-Algorithmus zu beschleunigen?

Aufgabe 12 *Java Bytecode*

Auf den Studentenrechnern ist die Software `jasmin` installiert, ein Assembler für die Java Virtual Machine. Mit `jasmin file.j` kann man eine Bytecode-Datei `file.j` in ein Java-Class-File `file.class` umwandeln. Dieses Class-File kann dann durch Aufruf von `java file.class` ausgeführt werden. Vergessen Sie nicht, vorher den Pfad `“.”` mit in `$CLASSPATH` aufzunehmen.

Das Format einer Bytecode-Datei wird unter <http://jasmin.sourceforge.net/> beschrieben, eine dokumentierte Vorlage ist hier erhältlich:

<http://www.fmi.uni-stuttgart.de/szs/teaching/ss2005/analyse/vorlage.j>

Ändern Sie die Vorlage `vorlage.j` so ab, dass Java folgende Fehlermeldungen ausgibt:

- `Inconsistent stack height 0 != 1` (Verschiedene Stack-Längen an der gleichen Programmstelle)
- `Accessing value from uninitialized register`
- `Stack size too large`
- `Unable to pop operand off an empty stack`
- `Expecting to find integer on stack`