Technische Universität Wien
Programmiersprachen und Übersetzer
Prof. Dr. J. Knoop

A-1040 Wien
Argentinierstr. 8
Tel.: 01-58801-18510

“ Analyse und Verifikation (185.276, VU 2.0, ECTS 3.0) ” SS 2020

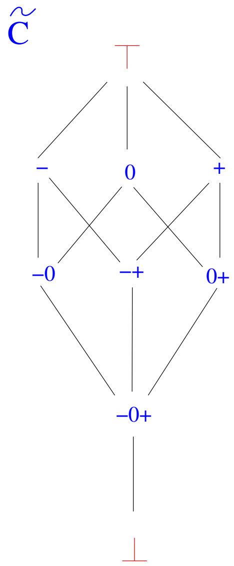
Übungsblatt 6 30.04.2020

Die Aufgaben beziehen sich auf *Kapitel 7, 8* und *9* der Vorlesung.

Gegeben ist der Verband $\widetilde{\mathcal{C}} = (C, \sqsubseteq, \sqcap, \sqcup, \perp, \top)$ für Vorzeichenanalyse mit

$$C = \{\perp, -, 0, +, -0, 0+, -0+, \top\}.$$

Dabei stehen die Symbole $-$, $+$ und 0 für die Menge aller echt negativen, aller echt positiven ganzen Zahlen und die einelementige Menge mit der Zahl null als Element. Entsprechend stehen die Symbole $-0+$ für alle ganzen Zahlen, -0, $0+$ für alle negativen bxw. positiven ganzen Zahlen einschließlich null usw. Das Symbol $-+$ steht somit für alle ganzen Zahlen mit Ausnahme der null und damit für die Information ungleich null. Entsprechend stehen die Symbole -0, $0+$ für die Information kleiner oder gleich null bzw. größer oder gleich null usw. Die Elemente $\perp$ und $\top$ schließlich stehen für keine Information bzw. die unerfüllbare, inkonsistente Allinformation.



Sei $V = \{x_1, x_2, \ldots, x_k\}$ eine endliche Menge von in Programmen vorkommen dürfender Variablen und

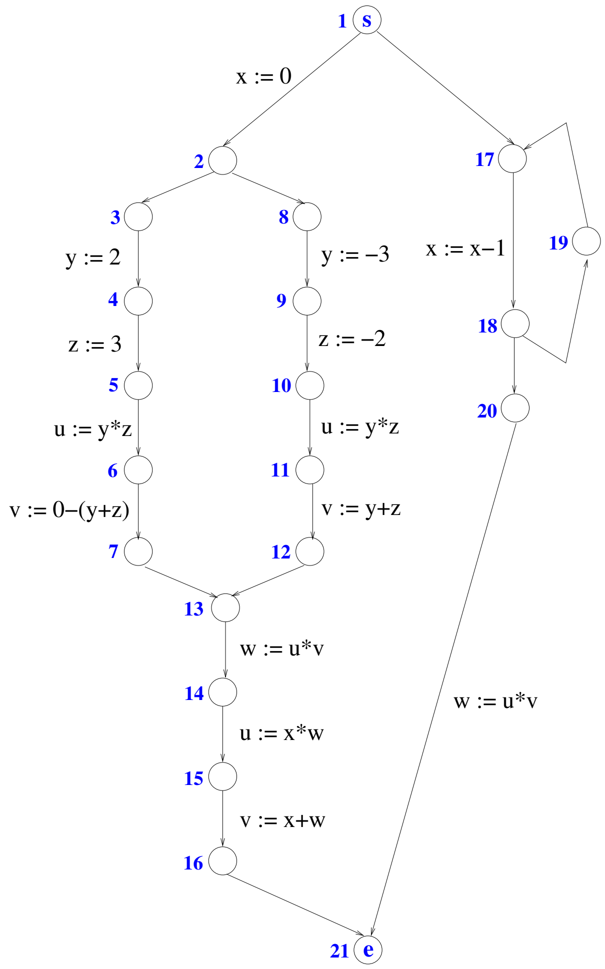
$$\Sigma = \{\sigma \mid \sigma : V \to C\}$$

die Menge abstrakter Programmzustände über $\mathcal{C}$.

Seien weiters $\mathbf{T}$ die Menge der Terme über V, der Menge $K = \{0, 1, -1, 2, -2, 3, -3, \ldots\}$ der Konstantensymbole für ganze Zahlen und der Menge $Op = \{+, -, *, /\}$ zweistelliger Operatoren (oder: Operationssymbole) mit der üblichen Bedeutung Addition, Subtraktion, Multiplikation und ganzzahlige Division auf der Menge der ganzen Zahlen.

<u>**Aufgabe 1**</u> : ((2+2)+2+(1+1)+(2+2) Punkte)

1. Definiere nach dem Vorbild von
   (a) Definition 7.10.2.2.4 die Auswertungsfunktion $\mathcal{A} : \mathbf{T} \to (\Sigma \to C)$.
   (b) Definition 7.10.2.2.5 die abstrakte Instruktionssemantik $\theta_i : \Sigma \to \Sigma$ für ι Instruktion der Form $x := t$ oder der Form *skip*.
2. Spezifiziere nach dem Vorbild der "einfachen-Konstanten"-Analyse aus Kapitel 7.10.2.3 die Vorzeichenanalyse, die für jede Programmstelle für jede Variable möglichst genau deren Vorzeichen bestimmt.
3. Ist die Vorzeichenanalyse (ohne Beweis)
   (a) monoton?
   (b) distributiv?
4. Wende die Vorzeichenanalyse auf folgendes Beispiel an:



und gib das Resultat der
   (a) *SUP*-Analyse
   (b) *MaxFP*-Analyse

bezüglich der Anfangszusicherung $\sigma_{\mathbf{S}} \in \Sigma$ mit $\sigma_{\mathbf{S}} = \lambda\, s.\, +$ durch Eintragen der entsprechenden DFA-Information an den Programmpunkten an.

Aufgabe 2 : (2+2)+2+(1+1)+(2+2) Punkte)

Wiederhole Aufgabe 1, aber ersetze den Zustandsverband Σ durch den 'auf den Kopf gestellten' Potenzmengenverband $\widetilde{\mathcal{P}(\Sigma)} = (\mathcal{P}(\Sigma), \supseteq, \cup, \cap, \Sigma, \emptyset)$ mit kleinstem Element Σ, größtem Element $\emptyset$ und Schnittoperation Mengenvereinigung $\cup$, wobei $\mathcal{P}$ den Potenzmengenoperator bezeichne. Im einzelnen:

1. Definiere nach dem Vorbild von Aufgabe 1
   (a) die Auswertungsfunktion $\mathcal{A} : \mathbf{T} \to (\mathcal{P}(\Sigma) \to C)$.
   (b) die abstrakte Instruktionssemantik $\theta_i : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ für ι Instruktion der Form $x := t$ oder der Form *skip*.
2. Spezifiziere nach dem Vorbild von Aufgabe 1 die Vorzeichenanalyse bezüglich der neuen Anweisungssemantik.
3. Ist die neue Vorzeichenanalyse (ohne Beweis)
   (a) monoton?
   (b) distributiv?
4. Wende die neue Vorzeichenanalyse auf das Beipiel aus Aufgabe 1 an und gib das Resultat der
   (a) *SUP*-Analyse
   (b) *MaxFP*-Analyse

   bezüglich der Anfangszusicherung $\{\sigma_{\mathbf{S}}\} \in \mathcal{P}(\Sigma)$, $\sigma_{\mathbf{S}} = \lambda\, s.\, +$, durch Eintragen der entsprechenden DFA-Information an den Programmpunkten an.

Aufgabe 3 : ((2+2)+(1+1)+2*(2+2)+(2+2) Punkte)

1. Gib die Spezifikationen der reversen Datenflussanalysen zu den Vorzeichenanalysen aus
   (a) Aufgabe 1
   (b) Aufgabe 2

   an.
2. Sind die reversen Datenflussanalysen (ohne Beweis)
   (a) monoton?
   (b) additiv?
3. Wende die beiden reversen Datenflussanalysen auf das Beispiel aus Aufgabe 1 an und gib jeweils die Resultate der
   (a) *RVUP*-Analyse
   (b) *RMinFP*-Analyse

bezüglich der DFA-Frage $\sigma_{\mathbf{e}} = \lambda\, s. \begin{cases} + & \text{falls } s = w \\ \perp & \text{sonst} \end{cases}$

bzw. $\{\sigma_{\mathbf{e}}\}$ als DFA-Anfrage am Endknoten **e** durch Eintragen der entsprechenden RDFA-Information an den Programmpunkten an.

4. Lassen sich die Ergebnisse der reversen Datenflussanalysen am Startknoten sinnvoll interpretieren?

Aufgabe 4 : (2+1+2 Punkte)

Der Erfolg paralleler Datenflussanalyse für unidirektionale Bitvektorprobleme beruht letztlich darauf, dass der Effekt eines Pfades ausschließlich vom Effekt der letzten Anweisung abhängt:

$$[\![\langle n_1, n_2, \ldots, n_k \rangle]\!] = [\![n_k]\!] \qquad (*)$$

1. Welche zentralen Probleme der Analyse paralleler Programme werden aufgrund der Gültigkeit von $(*)$ für unidirektionale Bitvektorprobleme einfach behandelbar?
2. Gilt $(*)$ auch für die "einfache-Konstanten"-Analyse?
3. Wie (weit) muss die "einfache-Konstanten"-Analyse abgeschwächt werden, damit für die abgeschwächte Konstantenanalyse (die weniger Terme als konstant erkennt als die "einfache-Konstanten"-Analyse) Eigenschaft $(*)$ gilt? D.h. wie muss die Termauswertungsfunktion $\mathcal{A}$ der "einfache-Konstanten"-Analyse aus Kapitel 7.10.2 abgeändert werden?

Iucundi acti labores.
Getane Arbeiten sind angenehm.
Cicero (106 - 43 v.Chr.)
röm. Staatsmann und Schriftsteller

Abgabe: Mittwoch, den 13.05.2020.