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

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

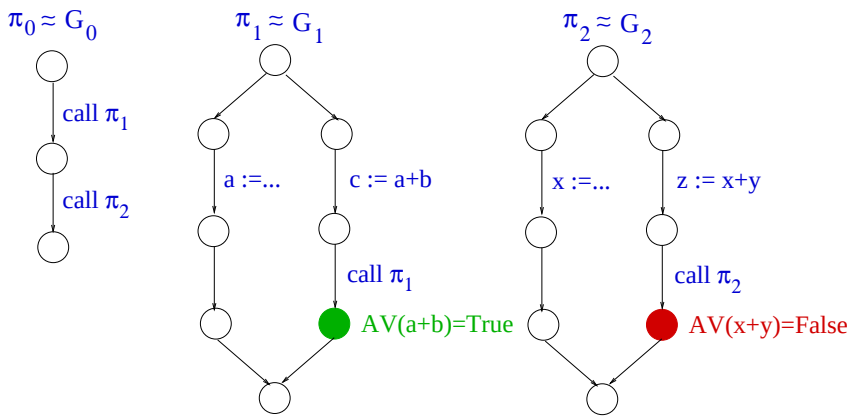
" Optimierende Compiler (185.A04, VU 2.0, ECTS 3.0) " WS 2020/21

Übungsblatt 8 10.12.2020

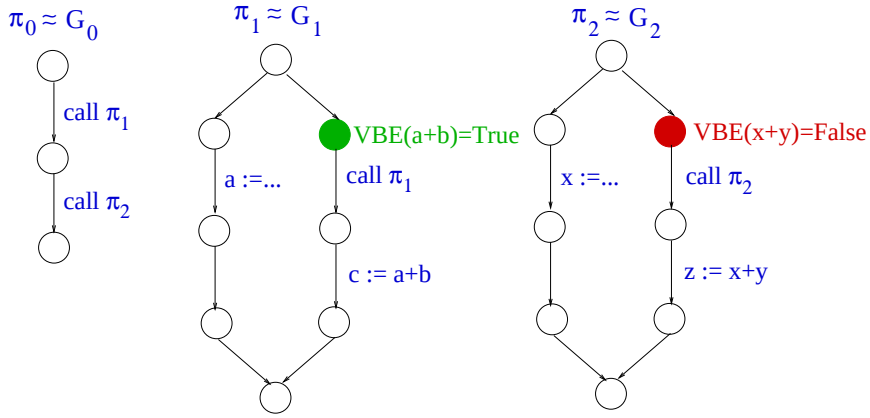
Aufgabe 1 : (4+4 Punkte)

In den Abbildungen a) und b) ist ein (Haupt-) Programm π_0 gegeben zusammen mit je zwei Prozeduren π_1 und π_2. Noch nicht angegeben ist, ob die Variablen a, b, c, x, y und z jeweils in π_0 oder in π_1 bzw. π_2 deklariert sind.

a)



b)



Geben Sie für die Programme in den Abbildungen a) und b) an, wo die Variablen a und b sowie x und y deklariert sein können, damit die Eigenschaften für *availability* bzw. *very busyness* an den farblich markierten Knoten in der angegebenen Weise gelten.

Aufgabe 2 : (4 Punkte)

Zeigen Sie anhand der interprozeduralen Flussgraphen zu den Flussgraphsystemen aus Aufgabe 1 und einzelner geeigneter Programmpfade auf, warum es für akkurate interprozedurale Datenflussanalyse nötig ist, DFA-Stacks (oder ausdrucksgleiche Datenstrukturen) zusammen mit darauf ausgedehnten lokalen abstrakten Semantik- und Rückkehrfunktionen einzuführen, um Eigenschaften wie in Aufgabe 1 berechnen zu können.

Aufgabe 3 : (5+5 Punkte)

Geben Sie Spezifikationen für die interprozedurale Analysen zur Berechnung von

1. *availability* ($\widehat{=}$ up-safety)
2. *very busyness* ($\widehat{=}$ down-safety)

Information für einen beliebig, aber fest gewählten Kandidatenausdruck t an.

In Kapitel 10.3.3 haben wir gesehen, dass die intraprozedural für sequentielle Programme erfolgreiche, zu berechnungsoptimalen Ergebnissen führende Strategie, Berechnungen so früh wie möglich in einem Programm zu platzieren, für parallele Programme nicht erfolgreich ist und zu einer unerwünschten Sequentialisierung des Programms führen kann.

In den Aufgaben 4, 5 und 6 wollen wir untersuchen, ob es beim Übergang von intra- zu interprozeduraler Elimination partiell redundanter Berechnungen ebenfalls 'überraschende' Effekte gibt.

Aufgabe 4 : (10 Punkte)

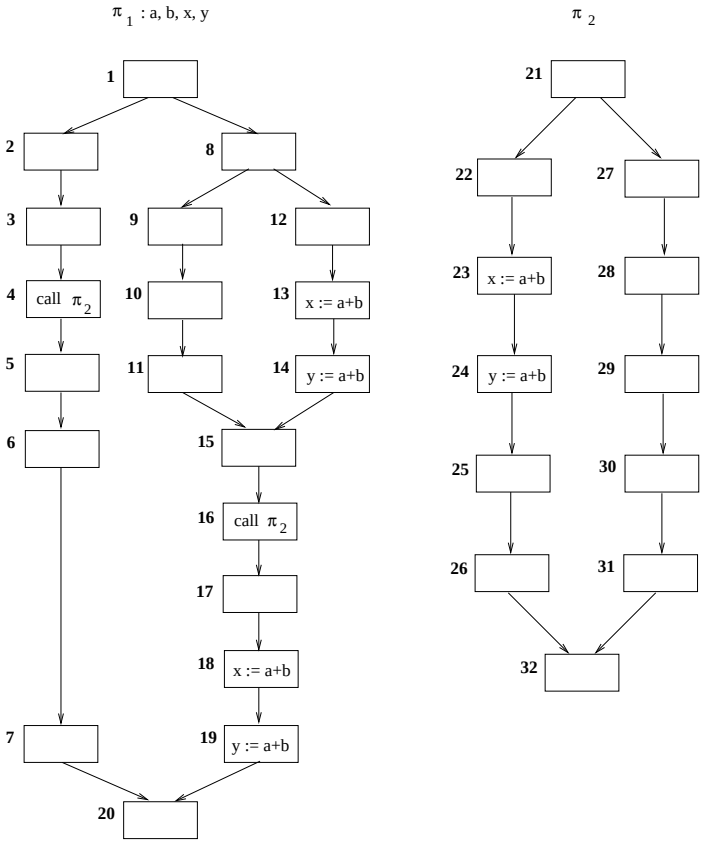
Intraprozedural gilt gemäß des Sicherheitslemmas 7.1.2.4 die folgende Äquivalenz über die Zerlegbarkeit von Sicherheit in Aufwärts- und Abwärtssicherheit:

$$\forall\, n \in N.\ \mathit{Safe}(n) \iff \mathit{U\text{-}Safe}(n) \vee \mathit{D\text{-}Safe}(n)$$

Gilt diese Äquivalenz interprozedural auch? Beweis oder Gegenbeispiel.

Aufgabe 5 : (6+2+2+2 Punkte)

Betrachten Sie folgendes Programm Π:



Programm Π enthält partiell redundante Berechnungen von $a + b$.

1. Beseitigen Sie diese partiell redundanten Berechnungen durch geeignete Initialisierungen und Benutzungen einer Hilfsvariablen h für $a + b$ so gut wie möglich.

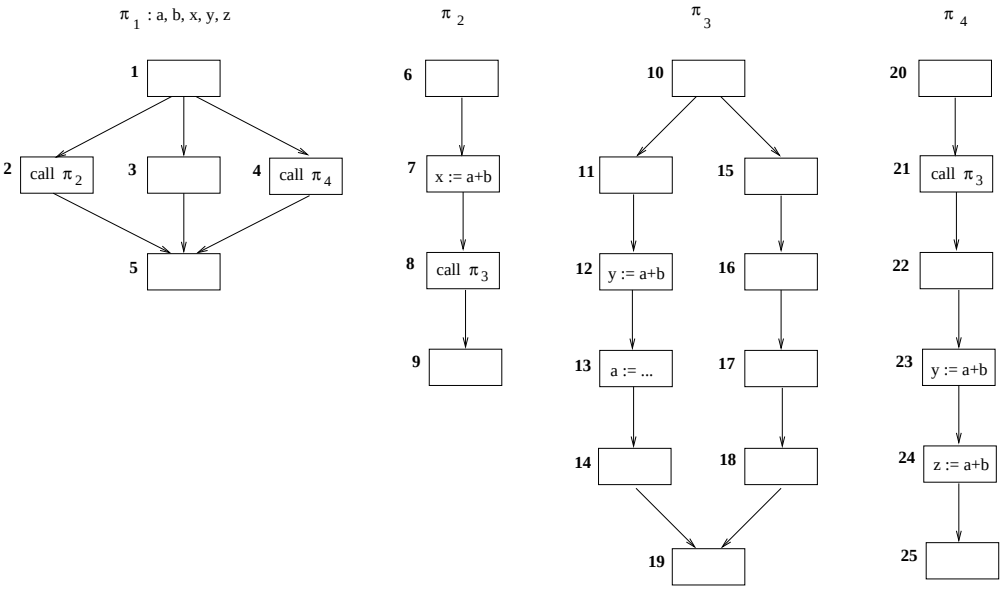
2. Ist Ihr resultierendes Programm (i.S.d. interprozeduralen Entsprechung von Definition 7.1.3.1) berechnungsoptimal? D.h. keine andere Initialisierung und Benutzung der Hilfsvariablen würde zu weniger Berechnungen von $a + b$ zur Laufzeit führen?

3. Ist Ihr resultierendes Programm kanonisch? Wir nennen das aus einer PRE-Transformation resultierende Programm (und die zugehörige Transformation selbst) *kanonisch*, wenn jede Initialisierung einer Hilfsvariablen auf jedem von der Initialisierungsstelle ausgehenden Programmpfad mindestens einmal benutzt wird.

4. Sind berechnungsoptimale intraprozedurale PRE-Transformationen kanonisch? Liegt hier ein Unterschied zum interprozeduralen Fall vor?

Aufgabe 6 : (10 Punkte)

Betrachten Sie folgendes Programm Π:



Programm Π enthält partiell redundante Berechnungen.

Zeigen Sie, dass sich diese Redundanzen nicht berechnungsoptimal beseitigen lassen. Geben Sie dazu zwei berechnungsminimale Programme Π' und Π'' an, die bezüglich der Relation 'berechnungsbesser' unvergleichbar sind und für die es 'offensichtlich' kein Programm gibt, das berechnungsbesser als beide ist.

Abgabe: Dienstag, den 12.01.2021, per e-mail an: `knoopcomplang.tuwien.ac.at`