Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Christoph Baumann

Subvariablen und Speicher der C_0-Maschine

Änderung der Definition

Mit der aktuellen Definition von Subvariablen (inkl. Variablen) und Speichern treten Probleme bei der Ausführung von Zuweisungen und Heapallokationen auf.

Zum Einen haben derzeit Bindungen die Form $(c.m, i)$. Dies ist problematisch, da somit Variablen an Speicher in bestimmten C_0-Konfigurationen gebunden werden. Im statischen Teil der Semantik verwendeten wir Notation (m, i) für Variable i im Speicher m. Für den dynamischen Teil jetzt m durch $c.m$ zu ersetzen, bringt Probleme mit sich. So ist dann formal nicht mehr gesichert, dass sich die Bindungen von Variablen bei Zuweisungen nicht ändern. $bind(c', id) = bind(c, id)$ wäre nicht mehr gewährleistet, da sich ja der Inhalt des Speichers m, in dem id an Stelle i liegt, verändern könnte. Somit würde gelten $bind(c', id) = (c'.m, i) \neq (c.m, i) = bind(c, id)$. Wenn sich Bindungen durch Zuweisungen ändern können, würde dies insbesondere heißen, dass die Werte von Pointern, nämlich Bindungen, veralten könnten. Die Pointer würden auf eine Speicherstelle in einer ganz bestimmten Konfiguration zeigen. Dies wäre zwar an sich kein Problem, da sich bei Speichern zur Laufzeit nur die Inhalte, nicht aber die Symboltabellen ändern. Die Pointer würden also immer noch auf die gleichen Variablen zeigen, dennoch müsste man zusätzliche Argumentation auffahren, um dies zu zeigen. Eine ähnliche Situation tritt bei der Zuweisung und Allokation von Variablen auf dem heap memory auf. Hier würden Pointer ab einem gewissen Zeitpunkt ebenfalls auf veraltete Versionen des Heaps zeigen.

Die zweite Problematik betrifft die Notation der va-Funktion für (Sub)variablen u. In der statischen Semantik genügt es über den Wert von Variablen mit $va(u)$ zu sprechen. Betrachtet man aber eine Folge von Konfigurationen, so könnte man mit der vorliegenden Notation nicht definieren, was der Wert einer Variable in einer bestimmten Konfiguration der C_0-Maschine ist. Es fehlt einfach der Bezug zu c in der Definition von $va(u)$.

Lösungsvorschlag: 1. Aufteilen von Speichern in ihren Inhalt und ihre Symboltabelle

m	$\in$	$\{gm, hm\} \cup \{lms(j) \mid j \in \mathbb{N}\}$	Namen von Speichern
$st.m$	$=$	$(st.m.n, st.m.name, st.m.typ)$	Symboltabellen
$ct.m$	$:$	$[0 : size(c.st.m) - 1] \to \{\text{einfache Werte}\}$	Speicherinhalte

Wir definieren Variablen und Subvariablen dann wie gewohnt über die Namen der Speicher.

- (m, i) Variable i im Speicher m
- $(m, i)s$ Subvariable im Speicher m mit Selektoren $s = s_1 \cdots s_j$

Die Definitionen für Größe und Basisadresse werden entsprechend angepasst. Der Wert von Subvariablen $u = (m, i)s$ ergibt sich zu:

$$va(u) = ct.m_{size(u)} ba(u)$$

Für die dynamische C_0-Semantik lautet die Konfiguration c nun

$$c = (c.pr, c.rd, c.rbs, c.st, c.ct),$$

wobei gilt:

$$\begin{aligned} c.st &= (c.st.gm, c.st.hm, c.st.lms) \\ c.ct &= (c.ct.gm, c.ct.hm, c.ct.lms) \end{aligned}$$

2. Wir erweitern die Notation von va um einen Bezug zur aktuellen Konfiguration c.

$$va(c,u) = c.ct.m_{size(u)}ba(u)$$

$c^i.ct.m$ verweist dabei auf die Speicherinhalte $c^i.ct.gm$, $c^i.ct.hm$, $c^i.ct.lms$ der Konfiguration c^i. Man beachte die erneute Überladung der va-Funktion.

Wir definieren die Bindung und den Wert von Bezeichnern X bei der Ausdrucksauswertung wie folgt.

$$\begin{aligned} bind(c,X) &= \begin{cases} (top(c),j) & : \quad top(c).name(j) = X \\ (gm,j) & : \quad (\nexists i : top(c).name(i) = X) \wedge c.gm.name(j) = X \end{cases} \\ va(c,X) &= va(c, bind(c,X)) \end{aligned}$$

Für Zuweisungen $id = e$ soll dann gelten.

$$va(c',id) = va(c,e)$$

Bezeichne id eine Subvariable $u = (m,i)s = bind(c,id)$ in Speicher m.

$$\begin{aligned} va(c',id) &= va(c', bind(c',id)) \\ &= va(c', bind(c,id)) \\ &= va(c',u) \\ &= c'.ct.m_{size(u)}ba(u) \end{aligned}$$

Wir erkennen, dass insbesondere $bind(c',id) = bind(c,id)$ für Zuweisungen gilt. Dann erhalten wir die Semantik der Zuweisung $id = e$.

$$c'.ct.m_{size(bind(c,id))}ba(bind(c,id)) = va(c,e)$$

Für Allokationen von Variablen des Typs t auf dem heap memory und die Zuweisung an Pointer p mittels der Anweisung $p = new\ t$ wird der heap memory wie folgt erweitert.

$$\begin{aligned} c'.st.hm.n &= c.st.hm.n + 1 = x \\ c'.st.hm.name(x) &= \Omega \\ c'.st.hm.typ(x) &= \varepsilon\{i \mid tt.name(i) = t\} \end{aligned}$$

Dann weisen wir p seinen neuen Wert zu.

$$va(c',p) = (hm,x)$$