Semantics of a Call-by-Need Lambda Calculus with McCarthy's amb for Program Equivalence

David Sabel

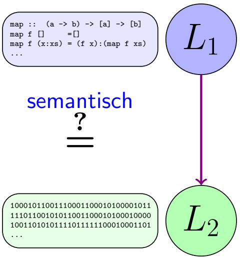
Institut für Informatik
Fachbereich Informatik und Mathematik
Goethe-Universität Frankfurt

Kolloquium zum GI-Dissertationspreis 2008
Dagstuhl, 20. Mai 2009

Motivation

Ziel: **korrekte** Compiler

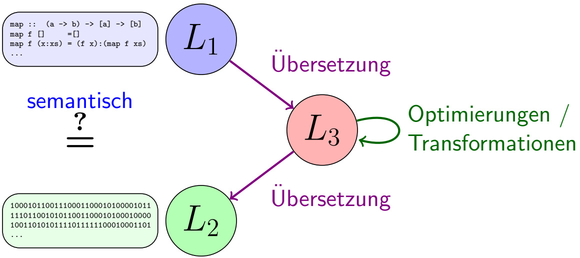
für moderne insb. nebenläufige Programmiersprachen



Motivation

Ziel: **korrekte** Compiler

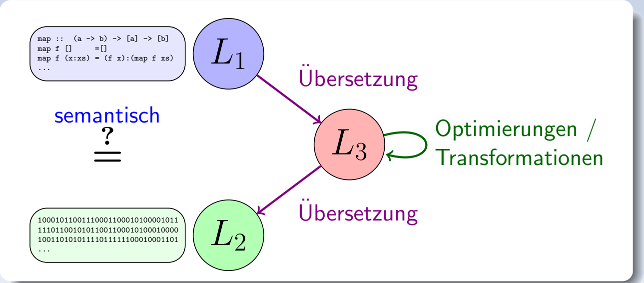
für moderne insb. nebenläufige Programmiersprachen



Motivation

Ziel: **korrekte** Compiler

für moderne insb. nebenläufige Programmiersprachen



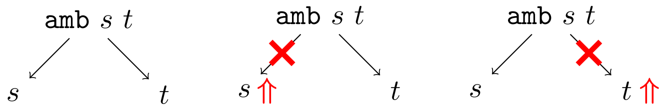
Gesucht:

- Formales Modell (Semantik): kanonisch, allgemein verwendbar
- Gleichheits- bzw. Korrektheitsbegriff
- Techniken / Hilfsmittel zum Korrektheitsnachweis

Untersuchung

Modell einer nebenläufigen Programmiersprache

- Kernsprache einer verzögert auswertenden higher-order FP mit
  - McCarthy's [McCarthy, 1963] `amb`-Operator:



  - *nebenläufige Auswertung von s und t unter Beachtung von Fairness*
  - *viele nd. Operatoren kodierbar:* `por`, `pconv`, `choice`, `dchoice`, `ndmerge`

Gleichheitstheorie basierend auf **Operationaler Semantik**

Wichtigste Vorarbeiten

- [Moran, 1998]: call-by-name / need & amb
- [Schmidt-Schauß, 2003]: Kernsprache mit I/O
- [Carayol et al., 2005]: call-by-name `amb`, faire must-Konvergenz

Der $\Lambda^{\mathtt{let}}_{\mathtt{amb}}$-Kalkül – Syntax

Syntax

$$
\begin{array}{lll}
E & ::= V & (\textit{Variable}) \\
 & \mid (\lambda V.E) & (\textit{Abstraktion}) \\
 & \mid (E_1\ E_2) & (\textit{Applikation}) \\
 & \mid (\mathtt{letrec}\ V_1 = E_1, \ldots V_n = E_n\ \mathtt{in}\ E) & (\mathtt{letrec}\text{-}\textit{Ausdruck}) \\
 & \mid (\mathtt{case}_T\ E\ Alt_1 \ldots Alt_{|T|}) & (\mathtt{case}\text{-}\textit{Ausdruck}) \\
 & \mid (c_{T,i}\ E_1 \ldots E_{\mathrm{ar}(c_{T,i})}) & (\textit{Konstruktorapplikation}) \\
 & \mid (\mathtt{seq}\ E_1\ E_2) & (\mathtt{seq}\text{-}\textit{Ausdruck}) \\
 & \mid (\mathtt{amb}\ E_1\ E_2) & (\mathtt{amb}\text{-}\textit{Ausdruck}) \\
Alt & ::= ((c_{T,i}\ V_1 \ldots V_{\mathrm{ar}(c_{T,i})})\ \rightarrow\ E) & (\mathtt{case}\text{-}\textit{Alternative})
\end{array}
$$

klassischer Lambdakalkül
+rekursive `let`-Ausdrücke
+case & Konstruktoren
+sequentielle Auswertungen
+nichtdeterministisches `amb`

if s_1 then s_2 else s_3

$\mathtt{case}_{\mathtt{Bool}}\ s_1\ (\mathtt{True} \rightarrow s_2)\ (\mathtt{False} \rightarrow s_3)$

Der $\Lambda^{\texttt{let}}_{\texttt{amb}}$-Kalkül – Operationale Semantik

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Normalordnungsreduktion $\xrightarrow{\text{no}}$

Call-by-need, small-step Reduktion:

Anwenden von Rewriting-Regeln in Reduktionskontexten

(lbeta) $(\lambda x.s)\ t \to \texttt{letrec}\ x = t\ \texttt{in}\ s$

(cp) $\texttt{letrec}\ x_0 = \lambda y.s,\ \{x_i = x_{i-1}\}_{i=1}^{m} \ldots R^{-}[x_m] \ldots$
$\to \texttt{letrec}\ x_0 = \lambda y.s,\ \{x_i = x_{i-1}\}_{i=1}^{m} \ldots R^{-}[\lambda y.s] \ldots$

(amb-l-c) $\texttt{amb}\ v\ s \to v$, wenn v ein Wert

(amb-r-c) $\texttt{amb}\ s\ v \to v$, wenn v ein Wert

... ...

Nichtdeterminismus auch bei Wahl des Reduktionskontextes

Auswertung bis zur WHNF $= \lambda x.s,\ (c\ \overrightarrow{s_i}),\ (\texttt{letrec}\ Env\ \texttt{in}\ v),$
$(\texttt{letrec}\ x_1 = (c\ \overrightarrow{s_i}), \{x_i = x_{i-1}\}_{i=2}^{m}, Env\ \texttt{in}\ x_m)$

Faire N.O. $\xrightarrow{\text{fno}}$: Variante von $\xrightarrow{\text{no}}$ mit Ressourcen an den amb-Operatoren:

$(\texttt{amb}_{\langle m,n\rangle}\ s\ t)$ mit $m,n \in \mathbb{N}_0$

Rewriting passt Ressourcen an (Scheduling)

Programmgleichheit: Kontextuelle Äquivalenz

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

May-Konvergenz

$s{\downarrow}$ gdw. $\exists t : s \xrightarrow{\mathbf{no},*} t, t$ WHNF

"reduzibel zu einer WHNF"

Must-Konvergenz

$s{\Downarrow}$ gdw. $\forall t : s \xrightarrow{\mathbf{no},*} t \implies t{\downarrow}$

"jeder Nachfolger ist may-konvergent"

Kontextuelle Präordnung und Äquivalenz

$$s \leq_c t \quad \text{gdw.} \quad \underbrace{\forall C : C[s]{\downarrow} \implies C[t]{\downarrow}}_{\leq_c^{\downarrow}} \wedge \underbrace{\forall C : C[s]{\Downarrow} \implies C[t]{\Downarrow}}_{\leq_c^{\Downarrow}}$$

$$\sim_c \; = \; \leq_c \cap \geq_c$$

Theorem (Konvergenzäquivalenz $\xrightarrow{\text{no}}, \xrightarrow{\text{fno}}$)

Für alle Ausdrücke s: $s{\downarrow} \iff s{\downarrow}_{\text{fair}}$ und $s{\Downarrow} \iff s{\Downarrow}_{\text{fair}}$

Programmgleichheit: Kontextuelle Äquivalenz

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

May-Konvergenz

$s\downarrow$ gdw. $\exists t : s \xrightarrow{\mathbf{no},*} t, t$ WHNF

"reduzibel zu einer WHNF"

Must-Konvergenz

$s\Downarrow$ gdw. $\forall t : s \xrightarrow{\mathbf{no},*} t \implies t\downarrow$

"jeder Nachfolger ist may-konvergent"

Kontextuelle Präordnung und Äquivalenz

$$s \leq_c t \quad \text{gdw.} \quad \underbrace{\forall C : C[s]\downarrow \implies C[t]\downarrow}_{\leq_c^{\downarrow}} \wedge \underbrace{\forall C : C[s]\Downarrow \implies C[t]\Downarrow}_{\leq_c^{\Downarrow}}$$

$$\sim_c \; = \; \leq_c \cap \geq_c$$

Theorem (Konvergenzäquivalenz $\xrightarrow{\text{no}}$, $\xrightarrow{\text{fno}}$)

Für alle Ausdrücke s: $s\downarrow \iff s\downarrow_{\text{fair}}$ und $s\Downarrow \iff s\Downarrow_{\text{fair}}$

Nachweis von Korrektheiten

Programmtransformation P ist korrekt gdw. $P \subseteq \sim_c$

- Korrektheit widerlegen: Einfach (ein Gegenbeispiel genügt)
  z.B. True $\not\sim_c$ False, denn für
  $$C = \texttt{case}_{Bool}\ [\cdot]\ (\texttt{True} \to \texttt{True})\ (\texttt{False} \to \bot)$$
  gilt $C[\texttt{True}]\downarrow$ aber $C[\texttt{False}]\Uparrow$
- Korrektheit beweisen: Schwer (alle Kontexte)

$\Longrightarrow$ benötigt: Hilfsmittel für Korrektheitsbeweis

Kontextlemma

$$\leq_{c,\mathcal{R}} \subseteq \leq_c$$

wobei: $s \leq_{c,\mathcal{R}} t$ gdw. $\forall R : (R[s]\downarrow \Longrightarrow R[t]\downarrow) \wedge (R[s]\Downarrow \Longrightarrow R[t]\Downarrow)$

P Programmtransformation: Für $P \subseteq \leq_c$ zeige

$$\forall (s,t) \in P : \forall R : R[s]\downarrow \implies R[t]\downarrow \wedge R[s]\Downarrow \implies R[t]\Downarrow$$

Methoden zum Korrektheitsnachweis

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Gabel- & Vertauschungsdiagramme

Vollst. Darstellung der Reduktions- und Transformations-Überlappungen und Zusammenführbarkeit

$$
\begin{tikzcd}
\cdot \arrow[r, "\mathcal{S}{,}\widehat{P}"] \arrow[d, "\widehat{\mathbf{no}}{,}\leq 1"'] & \cdot \arrow[d, dashed, "\widetilde{\mathbf{no}}{,}k_2"] \\
\cdot \arrow[r, dashed, "rel{,}k_1"'] & \cdot
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[r, "i\mathcal{S}{,}\widehat{P}"] \arrow[d, dashed, "\widetilde{\mathbf{no}}{,}k_2"'] & \cdot \arrow[d, "\widehat{\mathbf{no}}{,}\leq 1"] \\
\cdot \arrow[r, dashed, "rel{,}k_1"'] & \cdot
\end{tikzcd}
$$

ermöglichen: Induktive Konstruktion von Reduktionsfolgen

- $S[s]{\downarrow} \implies S[t]{\downarrow}$
- $S[t]{\uparrow} \implies S[s]{\uparrow}$
  (äquiv. zu $S[s]{\Downarrow} \implies S[t]{\Downarrow}$)

$\xRightarrow[\text{Lemma}]{\text{Kontext}} P \subseteq \leq_c$

$$
\begin{tikzcd}
S[s] \arrow[r, "\mathcal{S}{,}P"] \arrow[d, "\mathbf{no}"'] & S[t] \arrow[d, dashed, "\mathbf{no}"] \\
\cdot \arrow[r, dashed, "\mathcal{S}{,}P"'] \arrow[d, "\mathbf{no}{,}*"'] & \cdot \arrow[d, dashed, "\mathbf{no}{,}*"] \\
s' \arrow[r, dashed, "\mathcal{S}{,}P"'] & t' \\
\text{WHNF} & \text{WHNF}
\end{tikzcd}
\qquad
\begin{tikzcd}
S[s] \arrow[r, "\mathcal{S}{,}P"] \arrow[d, dashed, "\mathbf{no}"'] & S[t] \arrow[d, "\mathbf{no}"] \\
\cdot \arrow[r, dashed, "\mathcal{S}{,}P"'] \arrow[d, dashed, "\mathbf{no}{,}*"'] & \cdot \arrow[d, "\mathbf{no}{,}*"] \\
s' \arrow[r, dashed, "\mathcal{S}{,}P"'] & t' \\
\Uparrow & \Uparrow
\end{tikzcd}
$$

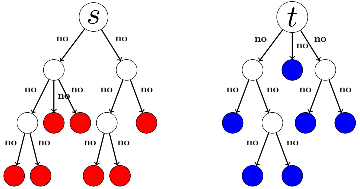
Weitere Methoden zum Korrektheitsnachweis

Standardisierungstheorem

$$t \xrightarrow{(\mathcal{C},\sim_c)\vee(\mathcal{S},\text{ambs}),*} t'$$

(1) t' WHNF $\Longrightarrow$ $t\downarrow$ (2) $t'\Uparrow$ $\Longrightarrow$ $t\uparrow$

Endliche Simulation



Theorem:

$M \in \mathcal{CSS}(s)$, $N \in \mathcal{CSS}(t)$:

$$M \ \langle\leq\rangle\ N \Longrightarrow s \leq_c t$$

N.B.: M, N endlich, s, t geschlossen

$\langle\leq\rangle$ gdw. $(\forall s \in M : \exists t \in N : s \leq_c^{\downarrow} t) \wedge (\forall t \in N : \exists s \in M : s \leq_c^{\Downarrow} t)$

Beispiel: $s = (\texttt{amb True False})$ und $t = (\texttt{amb False True})$.
$\{\texttt{True}, \texttt{False}\} \in \mathcal{CSS}(s) \cap \mathcal{CSS}(t) \implies s \sim_c t$.

Ergebnisse: Korrekte Programmtransformationen

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- Alle 16 deterministischen Kalkülregeln d.h. partielle Auswertung
- Garbage collection
  - $\texttt{letrec}\ x_1 = s_1, \ldots, x_n = s_n, Env\ \texttt{in}\ t \rightarrow \texttt{letrec}\ Env\ \texttt{in}\ t$ wenn $x_i \notin FV(t)$
- Kopieren von Variablen und Konstruktoren
  - $\texttt{letrec}\ x = y, \ldots C[x] \ldots \rightarrow \texttt{letrec}\ x = y, \ldots C[y] \ldots$
  - $\texttt{letrec}\ x = (c\ \overrightarrow{s_i}), \ldots C[x] \ldots \rightarrow \texttt{letrec}\ x = (c\ \overrightarrow{y_i}), \{y_i = s_i\}_{i=1}^{\mathrm{ar}(c)}, \ldots C[(c\ \overrightarrow{y_i})] \ldots$
- Kopieren von Ausdrücken mit nur einem Vorkommen
  - $\texttt{letrec}\ x = s\ \texttt{in}\ S[x] \rightarrow S[s]$, wenn x nur einmal vorkommt
- Ω-Terme sind kontextuell gleich und Ω-Terme dürfen kopiert werden
  - s ist Ω-Term gdw. für alle Env: $(\texttt{letrec}\ Env\ s){\Uparrow}$.

Weitere Ergebnisse

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Charakteristische Gesetze

- $(\texttt{amb}\ s\ t) \sim_c t \sim_c (\texttt{amb}\ t\ s)$, wenn s ein Ω-Term
- $(\texttt{dchoice}\ v_1\ v_2) \sim_c (\texttt{amb}\ v_1\ v_2) \sim_c (\texttt{choice}\ v_1\ v_2)$, geschlossene Werte v_1, v_2
- $(\texttt{amb}\ v\ v) \sim_c v$, v Wert
- $(\texttt{amb}\ s\ t) \sim_c (\texttt{amb}\ t\ s)$
- $(\texttt{amb}\ v_1\ (\texttt{amb}\ v_2\ v_3)) \sim_c \texttt{amb}((\texttt{amb}\ v_1\ v_2)\ v_3)$, v_i geschlossene Werte
- $(\texttt{choice}\ s\ t) \sim_c (\texttt{choice}\ t\ s)$, s, t geschlossen
- $(\texttt{choice}\ s\ s) \sim_c s$, s geschlossen
- $(\texttt{choice}\ s_1\ (\texttt{choice}\ s_2\ s_3)) \sim_c (\texttt{choice}(\texttt{choice}\ s_1\ s_2)\ s_3)$, s_i geschlossen
- $(\texttt{pconv}\ s\ t\ r) \sim_c r$, s, t geschlossen, $s{\Downarrow} \vee t{\Downarrow}$
- $(\texttt{por}\ s\ \texttt{True}) \sim_c \texttt{True} \sim_c (\texttt{por}\ \texttt{True}\ s)$
- $(\texttt{por}\ \texttt{False}\ \texttt{False}) \sim_c \texttt{False}$

Eigenschaft der kontextuellen Äquivalenz

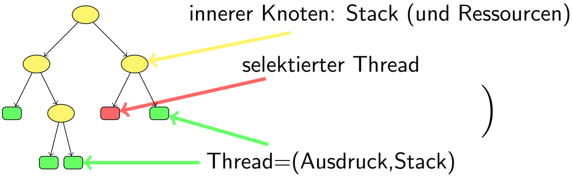
Theorem: $s \sim_c t$ gdw. $\forall C : C[s]{\Downarrow} \iff C[t]{\Downarrow}$

Die Concurrent Abstract Machine

- Maschinenmodell passend zum $\Lambda^{\mathtt{let}}_{\mathtt{amb}}$-Kalkül (mit eingeschr. Syntax)
- Basis:
  - Sestofts **mark 1** [Sestoft, 1997] für verzögerte Auswertung
  - Moran's Erweiterung um **Nebenläufigkeit** [Moran, 1998]
- unfaire Variante $\mathbf{CAM}$ und faire Variante $\mathbf{CAM}_F$ mit Ressourcen

Zustand

$$\left(\Gamma = \left\{ \begin{array}{l} x_1 \mapsto e_1, \\ \ldots \\ x_n \mapsto e_n \end{array} \right\}, \right.$$



$\big)$

Heap Prozessbaum

- Heap: Abbildung von Variablen auf Ausdrücke
- Prozessbaum: Hierarchie von Threads
- Selektierter Thread steuert Transition

Korrektheit der Übersetzung / Abstrakten Maschine

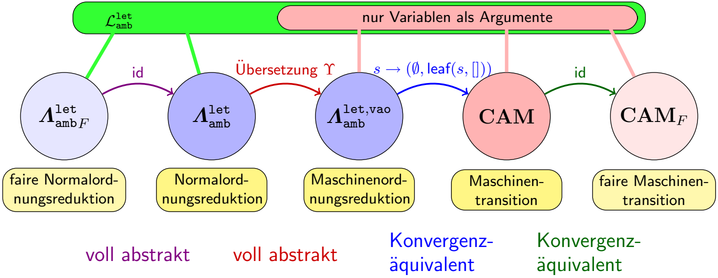
Korrektheit von Übersetzungen $T :: \Lambda_A \to \Lambda_B$

[Schmidt-Schauß, Niehren, Schwinghammer, Sabel, 2008]

- Konvergenzäquivalenz: $T(s){\downarrow_B} \iff s{\downarrow_A}$ und $T(s){\Downarrow_B} \iff s{\Downarrow_A}$
- Volle Abstraktheit: $T(s) \leq_B T(t) \Leftrightarrow s \leq_A t$

(äquivalente Gleichheitstheorien)

Korrektheitsbeweis $\Lambda^{\texttt{let}}_{\texttt{amb}} \to \mathrm{CAM}_F$



Fazit

- Ansatz aufbauend auf operationaler Semantik (auch) für nebenläufige Programmiersprachen erfolgreich
- Kontextuelle Äquivalenz basierend auf may- & must-Konvergenz ergibt erwartete Gleichungen
- Entwickelte Techniken erlauben Korrektheitsbeweis vieler Transformationen
- Korrektheit sowohl für Transformationen als auch für Übersetzungen

Ausblick

- Operationaler Ansatz auf viele Programmiersprachen anwendbar
  - nur small-step Reduktion, Wertbegriff notwendig
  - generisches Kontextlemma bereits entwickelt [Schmidt-Schauß & Sabel, 2007]
- Auf nebenläufigen, higher-order, call-by-value Prozesskalkül mit Speicher und Futures bereits angewendet [Niehren, Sabel, Schmidt-Schauß, Schwinghammer, 2007]
- Korrektheit von Implementierungen [Schwinghammer, Sabel, Schmidt-Schauß, Niehren, 2009]
- Kontextuelle Äquivalenz für getypte Sprachen [Sabel, Schmidt-Schauß, Harwath, 2009]