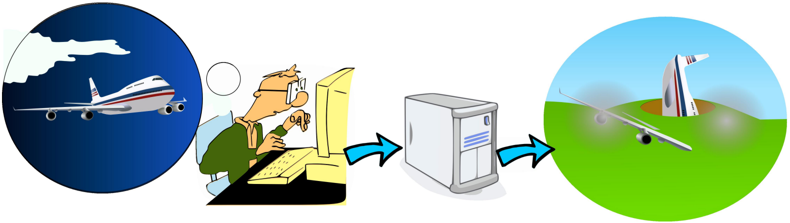
GOETHE UNIVERSITÄT
FRANKFURT AM MAIN

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

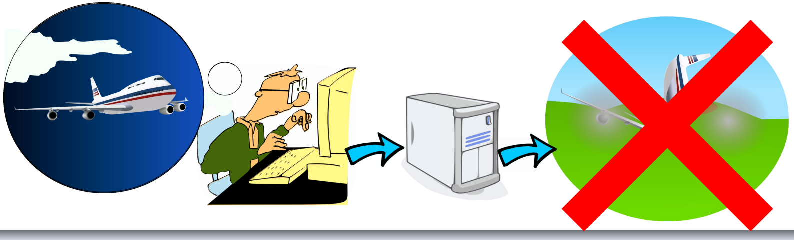
David Sabel

Goethe-Universität, Frankfurt

Motivation



Motivation



Compiler-Korrektheit für nebenläufige Programmiersprachen

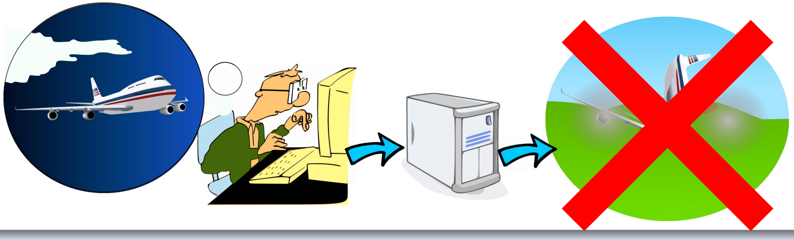
```
map ::  (a -> b) -> [a] -> [b]
map f []     =[]
map f (x:xs) = (f x):(map f xs)
...
```

$$\overset{?}{=}$$

```
100010110011100011000101000010111
111011001010110011000101000100000
100110101011110111111000100011011
...
```

$$L_1 \xrightarrow{\text{korrekt?}} L_2$$

Motivation



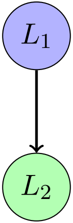
Compiler-Korrektheit für nebenläufige Programmiersprachen

```
map ::  (a -> b) -> [a] -> [b]
map f []      =[]
map f (x:xs) = (f x):(map f xs)
...
```

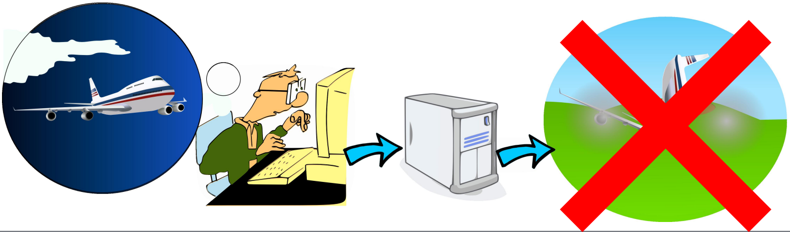
$\overset{?}{=}$

semantisch

```
10001011001110001100010100001011
11101100101011001100010100010000
10011010101111011111100010001101
...
```



Motivation



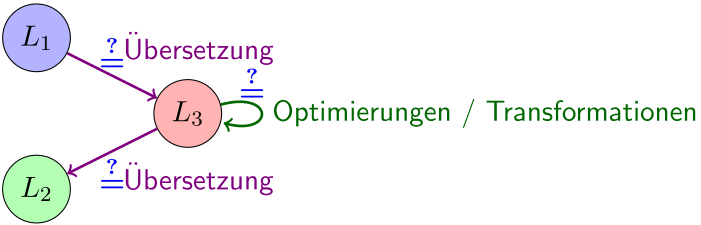
Compiler-Korrektheit für nebenläufige Programmiersprachen

```
map ::  (a -> b) -> [a] -> [b]
map f []      =[]
map f (x:xs) = (f x):(map f xs)
...
```

$\stackrel{?}{=}$

semantisch

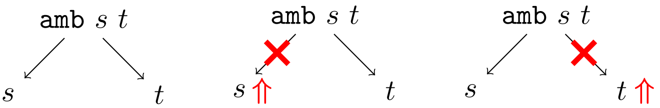
```
10001011001110001100010100001011
11101100101011001100010100010000
10011010101111011111100010001101
...
```



Untersuchung

Modell

- 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`

Wichtigste Vorarbeiten

- [Moran, 1998]: call-by-need & `amb`, may- und must-Konvergenz
- [Schmidt-Schauß, 2003]: Diagramme, faire must-Konvergenz, Kontextlemma
- [Carayol et al., 2005]: `amb`, faire must-Konvergenz

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

Syntax

$$E ::= V \mid (\lambda V.E) \mid (E_1\ E_2) \mid (\mathtt{letrec}\ V_1 = E_1, \ldots V_n = E_n\ \mathtt{in}\ E)$$
$$\mid (\mathtt{case}_T\ E\ Alt_1 \ldots Alt_{|T|}) \mid (c_{T,i}\ E_1 \ldots E_{\mathrm{ar}(c_{T,i})}) \mid (\mathtt{seq}\ E_1\ E_2) \mid (\mathtt{amb}\ E_1\ E_2)$$
$$Alt ::= ((c_{T,i}\ V_1 \ldots V_{\mathrm{ar}(c_{T,i})})\ \rightarrow\ E)$$

Normalordnungsreduktion $\xrightarrow{no}$

Call-by-need, small-step Reduktion:

Anwenden von Rewriting-Regeln in Reduktionskontexten

(lbeta) $(\lambda x.s)\ t \rightarrow \mathtt{letrec}\ x = t\ \mathtt{in}\ s$

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

(amb-l-c) $\mathtt{amb}\ v\ s \rightarrow v$

(amb-r-c) $\mathtt{amb}\ s\ v \rightarrow v$

... ...

Nichtdeterminismus auch bei Wahl des Reduktionskontextes

Faire N.O. $\xrightarrow{fno}$

Variante von $\xrightarrow{no}$:

Ressourcen an den amb-Operatoren:

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

Rewriting passt Ressourcen an (Scheduling)

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

Programmgleichheit: Kontextuelle Äquivalenz

May-Konvergenz

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

"reduzibel zu einer WHNF"

Must-Konvergenz

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

"jeder Nachfolger ist may-konvergent"

Kontextuelle Präordnung

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

Kontextuelle Äquivalenz

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

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

Für alle Ausdrücke s: $s\downarrow \iff s\downarrow_F$ und $s\Downarrow \iff s\Downarrow_F$

Programmgleichheit: Kontextuelle Äquivalenz

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

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

Kontextuelle Äquivalenz

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

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

Für alle Ausdrücke s: $s{\downarrow} \iff s{\downarrow}_F$ und $s{\Downarrow} \iff s{\Downarrow}_F$

Methoden zum Korrektheitsnachweis

Kontextlemma

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

Gabel- & Vertauschungsdiagramme

Vollst. Darstellung der Reduktions- und Transformations-Überlappungen

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

ermöglichen ...

Induktive Konstruktion von erfolgreichen und fehlschlagenden Reduktionsfolgen

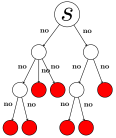
$$\begin{tikzcd} s \arrow[r, "{\mathcal{S},P}"] \arrow[d, "\mathbf{no}"'] & t \arrow[d, dashed, "\mathbf{no}"] \\ \cdot \arrow[r, dashed, "{\mathcal{S},P}"'] \arrow[d, "{\mathbf{no},*}"'] & \cdot \arrow[d, dashed, "{\mathbf{no},*}"] \\ s' \arrow[r, dashed, "{\mathcal{S},P}"'] & t' \end{tikzcd} \qquad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

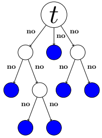
Standardisierungstheorem

(1) $t \xrightarrow{\mathcal{C},(\mathrm{ptc} \vee \mathrm{amb}),*} t'$, t' WHNF $\implies t{\downarrow}$

(2) $t \xrightarrow{(\mathcal{C},\mathrm{ptc}) \vee (\mathcal{S},\mathrm{ambs}),*} t'$, $t'{\Uparrow} \implies t{\uparrow}$

Endliche Simulation





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

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

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

Korrektheitsresultate

Korrekte Programmtransformationen

- Alle 16 deterministischen Kalkülregeln d.h. partielle Auswertung
- Garbage collection (letrec Env in $s \to s$)
- Kopieren von Variablen, Konstruktoren
- Kopieren von Ausdrücken mit nur einem Vorkommen und Ω-Termen
- Alle Ω-Terme sind kontextuell gleich
- Gesetze für amb und kodierte Operatoren:

$$(\texttt{amb}\ s_o\ s) \sim_c s \sim_c (\texttt{amb}\ s\ s_o) \text{ wenn } s_o\ \Omega\text{-Term}$$
$$(\texttt{dchoice}\ v_1\ v_2) \sim_c (\texttt{amb}\ v_1\ v_2) \sim_c (\texttt{choice}\ v_1\ v_2)$$
$$(\texttt{amb}\ w\ w) \sim_c w,\ \ (\texttt{amb}\ s_1\ s_2) \sim_c (\texttt{amb}\ s_2\ s_1),\ \ (\texttt{amb}\ v_1\ (\texttt{amb}\ v_2\ v_3)) \sim_c (\texttt{amb}\ (\texttt{amb}\ v_1\ v_2)\ v_3),$$
$$(\texttt{choice}\ t_1\ t_2) \sim_c (\texttt{choice}\ t_2\ t_1),\ \ (\texttt{choice}\ t\ t) \sim_c t,\ \ (\texttt{choice}\ t_1\ (\texttt{choice}\ t_2\ t_3)) \sim_c (\texttt{choice}\ (\texttt{choice}\ t_1\ t_2)\ t_3)$$
$$(\texttt{pconv}\ t_1\ t_2\ r) \sim_c r \text{ wenn } t_1{\Downarrow} \vee t_2{\Downarrow}$$
$$(\texttt{por}\ s\ \texttt{True}) \sim_c \texttt{True} \sim_c (\texttt{por}\ \texttt{True}\ s)$$
$$(\texttt{por}\ \texttt{False}\ \texttt{False}) \sim_c \texttt{False}$$
$$v, v_i, w, w_i \text{ Werte, } w, w_i, t, t_i \text{ geschlossen}$$

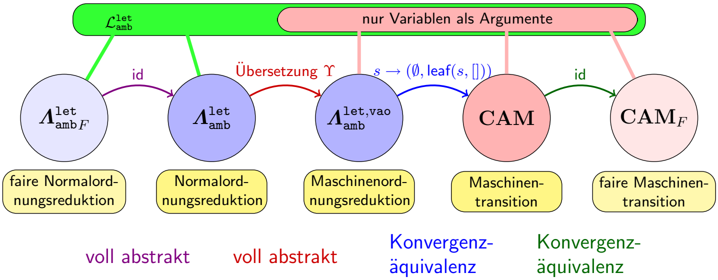
Eigenschaft der kontextuellen Äquivalenz

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

Abstrakte Maschine

- Erweiterung der mark-1 [Sestoft,1997] um Nebenläufigkeit
- Korrektur der Maschine von [Moran,1998]
- Faire und unfaire Variante

Korrektheitsbeweis $\Lambda^{\mathtt{let}}_{\mathtt{amb}} \rightarrow \mathrm{CAM}_F$



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

[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)

Fazit und Ausblick

- Operationaler Ansatz (auch) für nebenläufige Programmiersprachen erfolgreich
- Kontextuelle Äquivalenz basierend auf may- & must-Konvergenz ergibt erwartete Gleichungen
- Entwickelte Techniken erlauben Korrektheitsbeweis vieler Transformationen
- Sowohl Transformationen als auch Übersetzungen
- Moran's Resulate verbessert bzw. zum erfolgreichen Abschluss geführt

Ausblick

- Ansatz / Techniken auf viele Kalküle anwendbar
  - nur small-step Reduktion, Wertbegriff notwendig
  - für nebenläufigen Prozesskalkül mit Speicher und Futures bereits angewendet [Niehren, Sabel, Schmidt-Schauß, Schwinghammer, 2007]
- getypte Kalküle betrachten
- Diagramm-Methode automatisieren