SECD-Maschine

Till Helge Helwig
Stefan Huster

Informatik II

April 2009

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:
  - **S**tack (Parameter und Auswertungsergebnisse)

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:
  - **S**tack (Parameter und Auswertungsergebnisse)
  - **E**nvironment (Abstraktionen/Variablenbelegungen)

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:
  - **S**tack (Parameter und Auswertungsergebnisse)
  - **E**nvironment (Abstraktionen/Variablenbelegungen)
  - **C**ode (Stapel mit Anweisungen)

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:
  - **S**tack (Parameter und Auswertungsergebnisse)
  - **E**nvironment (Abstraktionen/Variablenbelegungen)
  - **C**ode (Stapel mit Anweisungen)
  - **D**ump (Ablage zur Zustandsspeicherung)

SECD-Maschine

- Theoretisches **Modell** einer Rechenmaschine
- Stapelbasiert
- Zustand der Maschine wird durch **vier Komponenten** beschrieben:
  - **S**tack (Parameter und Auswertungsergebnisse)
  - **E**nvironment (Abstraktionen/Variablenbelegungen)
  - **C**ode (Stapel mit Anweisungen)
  - **D**ump (Ablage zur Zustandsspeicherung)
- Kann **Terme des Lambda-Kalküls** auswerten

$\Rightarrow$ **Neue Syntax** als Zwischenrepräsentation nötig

Grundidee der Zwischenrepräsentation

- **Alternative Notation** für Lambda-Terme
- Verringerung der impliziten Deklarationen
- Umformulierung in **stapelbasierte** Anweisungs-Reihenfolge

Grundidee der Zwischenrepräsentation

- **Alternative Notation** für Lambda-Terme
- Verringerung der impliziten Deklarationen
- Umformulierung in **stapelbasierte** Anweisungs-Reihenfolge

Beispiel

Lambda-Term: $(\lambda\, x\, y\, .\, (+\, x\, y)\,)\ 9\ 18$
Zwischenrepräsentation: $\langle x, \langle y,\ x\ y\ \mathit{prim}_{+}\rangle\rangle\ 9\ \mathit{ap}\ 18\ \mathit{ap}$

Grundidee der Zwischenrepräsentation

- **Alternative Notation** für Lambda-Terme
- Verringerung der impliziten Deklarationen
- Umformulierung in **stapelbasierte** Anweisungs-Reihenfolge

Beispiel

Lambda-Term: $(\lambda\, x\, y\, .\, (+\, x\, y)\,)\ 9\ 18$
Zwischenrepräsentation: $\langle x, \langle y,\ x\ y\ \mathit{prim}_{+}\rangle\rangle\ 9\ \mathit{ap}\ 18\ \mathit{ap}$

- **Vorbereitung** zur Umformung:
  1. Lambda-Term **vollständig klammern**
  2. **Abstraktionen** mit mehr als einem Parameter **aufsplitten**

Lambda-Term → Zwischenrepräsentation

Umformung erfolgt schrittweise nach diesen Regeln:

1. **Konstanten und Variablen:**
   $[\![C]\!]_{SECD} = C$

Lambda-Term → Zwischenrepräsentation

Umformung erfolgt schrittweise nach diesen Regeln:

1. **Konstanten und Variablen:**
   $[\![C]\!]_{SECD} = C$
2. **Abstraktionen:**
   $[\![(\lambda x.Y)]\!]_{SECD} = \langle x, [\![Y]\!]_{SECD} \rangle$

Lambda-Term → Zwischenrepräsentation

Umformung erfolgt schrittweise nach diesen Regeln:

1. **Konstanten und Variablen:**
   $[\![c]\!]_{SECD} = c$
2. **Abstraktionen:**
   $[\![(\lambda x.Y)]\!]_{SECD} = \langle x, [\![Y]\!]_{SECD} \rangle$
3. **Applikationen:**
   $[\![(Y\ x)]\!]_{SECD} = [\![Y]\!]_{SECD}\ [\![x]\!]_{SECD}\ ap$

Lambda-Term → Zwischenrepräsentation

Umformung erfolgt schrittweise nach diesen Regeln:

1. **Konstanten und Variablen:**
   $[\![c]\!]_{SECD} = c$
2. **Abstraktionen:**
   $[\![(\lambda x.Y)]\!]_{SECD} = \langle x, [\![Y]\!]_{SECD} \rangle$
3. **Applikationen:**
   $[\![(Y\ x)]\!]_{SECD} = [\![Y]\!]_{SECD}\ [\![x]\!]_{SECD}\ ap$
4. **Primitive Operationen:**
   $[\![(O^n\ b_1 \ldots b_n)]\!]_{SECD} = [\![b_1]\!]_{SECD} \ldots [\![b_n]\!]_{SECD}\ prim_O$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$[\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!]$$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$\begin{aligned} & [\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!] \\ \equiv\quad & [\![(\lambda x.((x\,23)\;13))]\!]\;[\![(\lambda x.(\lambda y.(+x\;y)))]\!]\;ap \end{aligned}$$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$
\begin{aligned}
& [\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!] \\
\equiv\; & [\![(\lambda x.((x\,23)\;13))]\!]\;[\![(\lambda x.(\lambda y.(+x\;y)))]\!]\;ap \\
\equiv\; & \langle x, [\![(x\,23)\;13]\!]\rangle\;\langle x, [\![(\lambda y.(+x\;y))]\!]\rangle\;ap
\end{aligned}
$$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$
\begin{aligned}
& [\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!] \\
\equiv\; & [\![(\lambda x.((x\,23)\;13))]\!]\;[\![(\lambda x.(\lambda y.(+x\;y)))]\!]\;ap \\
\equiv\; & \langle x,\, [\![(x\,23)\;13]\!]\rangle\;\langle x,\, [\![(\lambda y.(+x\;y))]\!]\rangle\;ap \\
\equiv\; & \langle x,\, [\![(x\,23)]\!]\;13\;ap\rangle\;\langle x,\, \langle y,\, [\![(+x\;y)]\!]\rangle\rangle\;ap
\end{aligned}
$$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$
\begin{aligned}
& [\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!] \\
\equiv\;& [\![(\lambda x.((x\,23)\;13))]\!]\;[\![(\lambda x.(\lambda y.(+x\;y)))]\!]\;ap \\
\equiv\;& \langle x,\,[\![(x\,23)\;13]\!]\rangle\;\langle x,\,[\![(\lambda y.(+x\;y))]\!]\rangle\;ap \\
\equiv\;& \langle x,\,[\![(x\,23)]\!]\;13\;ap\rangle\;\langle x,\,\langle y,\,[\![(+x\;y)]\!]\rangle\rangle\;ap \\
\equiv\;& \langle x,\,[\![x]\!]\;[\![23]\!]\;ap\;13\;ap\rangle\;\langle x,\,\langle y,\,[\![x]\!]\;[\![y]\!]\;prim_{+}\rangle\rangle\;ap
\end{aligned}
$$

Einfaches Beispiel zum Umformung

Lambda-Term: $(\lambda x.x\,23\,13)\;(\lambda xy.(+x\;y))$
Vorbereitung: $(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))$
Umformungen:

$$
\begin{aligned}
& [\![(\lambda x.((x\,23)\;13))\;(\lambda x.(\lambda y.(+x\;y)))]\!] \\
\equiv\;& [\![(\lambda x.((x\,23)\;13))]\!]\;[\![(\lambda x.(\lambda y.(+x\;y)))]\!]\;ap \\
\equiv\;& \langle x,\,[\![(x\,23)\;13]\!]\rangle\;\langle x,\,[\![(\lambda y.(+x\;y))]\!]\rangle\;ap \\
\equiv\;& \langle x,\,[\![(x\,23)]\!]\;13\;ap\rangle\;\langle x,\,\langle y,\,[\![(+x\;y)]\!]\rangle\rangle\;ap \\
\equiv\;& \langle x,\,[\![x]\!]\;[\![23]\!]\;ap\;13\;ap\rangle\;\langle x,\,\langle y,\,[\![x]\!]\;[\![y]\!]\;prim_{+}\rangle\rangle\;ap \\
\equiv\;& \langle x,\,x\,23\;ap\;13\;ap\rangle\;\langle x,\,\langle y,\,x\;y\;prim_{+}\rangle\rangle\;ap
\end{aligned}
$$

Beispiel Aufgabe

Lambda-Term: $(\lambda xyz.\,(-\,(x\;y\;5)\;z))\;(\lambda xy.\,+\,x\;y)\;(*\;8\;5)\;3$

Beispiel Aufgabe

$$[\![((((\lambda x.\,(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z))))\,(\lambda x.\,(\lambda y.\,(+\,x\ y))))\,(*\ 8\ 5))\ 3)]\!]$$

Beispiel Aufgabe

$$
\begin{aligned}
& [\![((((\lambda x.\,(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z))))\,(\lambda x.\,(\lambda y.\,(+x\ y))))\,(*\ 8\ 5))\ 3)]\!] \\
\equiv\ & [\![(((\lambda x.\,(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z))))\,(\lambda x.\,(\lambda y.\,(+x\ y))))\,(*\ 8\ 5))]\!]\ [\![3]\!]\ ap \\
\equiv\ & [\![((\lambda x.\,(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z))))\,(\lambda x.\,(\lambda y.\,(+x\ y))))]\!][\![(*\ 8\ 5)]\!]\ ap\ 3\ ap \\
\equiv\ & [\![(\lambda x.\,(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z))))]\!][\![(\lambda x.\,(\lambda y.\,(+x\ y)))]\!]\ ap\ [\![8]\!][\![5]\!]\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ [\![(\lambda y.\,(\lambda z.\,(-\,((x\ y)\ 5)\ z)))]\!]\rangle\langle x,\ [\![(\lambda y.\,(+x\ y))]\!]\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ [\![(\lambda z.\,(-\,((x\ y)\ 5)\ z))]\!]\rangle\rangle\langle x,\ \langle y,\ [\![(+x\ y)]\!]\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ \langle z,\ [\![(-\,((x\ y)\ 5)\ z)]\!]\rangle\rangle\rangle\langle x,\ \langle y,\ [\![x]\!][\![y]\!]prim_+\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ \langle z,\ [\![((x\ y)\ 5)]\!][\![z]\!]\ prim_-\rangle\rangle\rangle\langle x,\ \langle y,\ x\ y\ prim_+\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ \langle z,\ [\![(x\ y)]\!][\![5]\!]\ ap\ z\ prim_-\rangle\rangle\rangle\langle x,\ \langle y,\ x\ y\ prim_+\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ \langle z,\ [\![x]\!][\![y]\!]\ ap\ 5\ ap\ z\ prim_-\rangle\rangle\rangle\langle x,\ \langle y,\ x\ y\ prim_+\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap \\
\equiv\ & \langle x,\ \langle y,\ \langle z,\ x\ y\ ap\ 5\ ap\ z\ prim_-\rangle\rangle\rangle\langle x,\ \langle y,\ x\ y\ prim_+\rangle\rangle\ ap\ 8\ 5\ prim_*\ ap\ 3\ ap
\end{aligned}
$$

Auswertung mittels SECD-Maschine

Tipps zur Notation

- Mit falscher Notation wird es schnell **unübersichtlich**
- In jedem Schritt sind **alle vier Komponenten** relevant
- **Mögliche Notationen:**
  - Vektorschreibweise: $\langle S,\ E,\ C,\ D \rangle \rightarrow \langle S',\ E',\ C',\ D' \rangle \rightarrow \cdots$

Auswertung mittels SECD-Maschine

Tipps zur Notation

- Mit falscher Notation wird es schnell **unübersichtlich**
- In jedem Schritt sind **alle vier Komponenten** relevant
- **Mögliche Notationen:**
  - Vektorschreibweise: $\langle S,\ E,\ C,\ D \rangle \rightarrow \langle S',\ E',\ C',\ D' \rangle \rightarrow \cdots$
  - Jede Komponenten seperat in einer Zeile notieren:

Stack:	S
Environment:	E
Code:	C
Dump:	D

$\rightarrow$

Stack:	S'
Environment:	E'
Code:	C'
Dump:	D'

$\rightarrow \cdots$

Auswertung mittels SECD-Maschine

Tipps zur Notation

- Mit falscher Notation wird es schnell **unübersichtlich**
- In jedem Schritt sind **alle vier Komponenten** relevant
- **Mögliche Notationen:**
  - Vektorschreibweise: $\langle S, E, C, D \rangle \to \langle S', E', C', D' \rangle \to \cdots$
  - Jede Komponenten seperat in einer Zeile notieren:

Stack:	S
Environment:	E
Code:	C
Dump:	D

$\to$

Stack:	S'
Environment:	E'
Code:	C'
Dump:	D'

$\to \cdots$

  - Tabelle mit einer Spalte pro Komponente:

Stack	Environment	Code	Dump
S	E	C	D
S'	E'	C'	D'
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Auswertung mittels SECD-Maschine

Hinweise zur Auswertung

- Immer zuerst auf den **Code** schauen
- Nicht von den anderen Komponenten verwirren lassen
- Die **aktuelle Anweisung** entscheidet was zu tun ist und welche Komponenten beachtet werden müssen

Konstanten-Regel

Konstanten-Regel

$\langle S,\ E,\ bC,\ D\rangle \rightarrow_{SECD} \langle bS,\ E,\ C,\ D\rangle$

Konstanten werden einfach vom Code auf den Stack **geschoben**.

Konstanten-Regel

Konstanten-Regel

$\langle S,\ E,\ bC,\ D\rangle \rightarrow_{SECD} \langle bS,\ E,\ C,\ D\rangle$

Konstanten werden einfach vom Code auf den Stack **geschoben**.

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle\langle\langle x,\ x\ y\ prim_{+}\rangle, ()\rangle, & (y \rightarrow 23), & 15\ \langle z,z\rangle\ 17\ ap, & \epsilon\rangle \\
\langle 15\ \langle\langle x,\ x\ y\ prim_{+}\rangle, ()\rangle, & (y \rightarrow 23), & \langle z,z\rangle\ 17\ ap, & \epsilon\rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Variablen-Regel

Variablen-Regel

$\langle S,\ E,\ xC,\ D\rangle \rightarrow_{SECD} \langle E(x)\ S,\ E,\ C,\ D\rangle$

Variablen werden mittels Environment **ausgewertet** und auf den Stack befördert.

Variablen-Regel

Variablen-Regel

$\langle S,\ E,\ xC,\ D\rangle \rightarrow_{SECD} \langle E(x)\ S,\ E,\ C,\ D\rangle$

Variablen werden mittels Environment **ausgewertet** und auf den Stack befördert.

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle\langle\langle x,\ x\ y\ prim_{+}\rangle, ()\rangle, & (y \rightarrow 23), & y\ \langle z,z\rangle\ 17\ ap, & \epsilon\rangle \\
\langle 23\ \langle\langle x,\ x\ y\ prim_{+}\rangle, ()\rangle, & (y \rightarrow 23), & \langle z,z\rangle\ 17\ ap, & \epsilon\rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Abstraktions-Regel

Abstraktions-Regel

$$\langle S,\ E,\ \langle x,\ Y\rangle\ C,\ D\rangle \rightarrow_{SECD} \langle\langle\langle x,\ Y\rangle,\ E\rangle\ S,\ E,\ C,\ D\rangle$$

Abstraktionen werden als **Paar mit dem aktuellen Environment** auf den Stack geschoben.

Abstraktions-Regel

Abstraktions-Regel

$$\langle S,\ E,\ \langle x,\ Y\rangle\ C,\ D\rangle \rightarrow_{SECD} \langle\langle\langle x,\ Y\rangle,\ E\rangle\ S,\ E,\ C,\ D\rangle$$

Abstraktionen werden als **Paar mit dem aktuellen Environment** auf den Stack geschoben.

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle \epsilon, & (y \rightarrow 23), & \langle z, z\ y\ prim_{\star}\rangle\ 17\ ap, & \epsilon\rangle \\
\langle\langle\langle z,\ z\ y\ prim_{+}\rangle, (y \rightarrow 23)\rangle, & (y \rightarrow 23), & 17\ ap, & \epsilon\rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Primitivum-Regel

Primitivum-Regel

$$\langle T_n \ldots T_1 S,\ E,\ prim_{O^n} C,\ D \rangle \rightarrow_{SECD} \langle O(T_1 \ldots T_n)\ S,\ E,\ C,\ D \rangle$$

- Gemäß der Stelligkeit n werden n Operanden (T_i) vom Stack geholt
- Die Operation O wird auf diesen Operanden ausgeführt und das Ergebnis auf den Stack geschrieben

Primitivum-Regel

Primitivum-Regel

$\langle T_n \ldots T_1 S,\ E,\ prim_{O^n} C,\ D \rangle \rightarrow_{SECD} \langle O(T_1 \ldots T_n)\ S,\ E,\ C,\ D \rangle$

- Gemäß der Stelligkeit n werden n Operanden (T_i) vom Stack geholt
- Die Operation O wird auf diesen Operanden ausgeführt und das Ergebnis auf den Stack geschrieben

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle 5\ 3, & (y \rightarrow 23), & prim_{\star}\ \langle z, z \rangle\ 17\ ap, & \epsilon \rangle \\
\langle 15, & (y \rightarrow 23), & \langle z, z \rangle\ 17\ ap, & \epsilon \rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Applikations-Regel

Applikations-Regel

$$\langle T\ \langle\langle x,\ Y\rangle,\ E'\rangle\ S,\ E,\ ap\ C,\ D\rangle \rightarrow \langle \epsilon,\ E' \cup \{x \rightarrow T\},\ Y,\ \langle S,\ E,\ C,\ D\rangle\rangle$$

1. Vom Stack werden **Argument** (T) und **Abstraktion** geholt
2. Auf dem Dump werden restlicher Stack, Environment, restlicher Code und Dump **gesichert**
3. Das mitgelieferte Environment (E') wird um die neue Zuweisung $x \rightarrow T$ **ergänzt** und als **aktuelles eingesetzt**
4. Der Stack wird **geleert** und der Körper der Abstraktion zum **neuen Code**

Applikations-Regel

Applikations-Regel

$$\langle T\ \langle\langle x,\ Y\rangle,\ E'\rangle\ S,\ E,\ ap\ C,\ D\rangle \rightarrow \langle \epsilon,\ E' \cup \{x \rightarrow T\},\ Y,\ \langle S,\ E,\ C,\ D\rangle\rangle$$

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle 42\ \langle\langle x,\ x\ prim_!\rangle,\ (y \rightarrow 23)\rangle\ 25, & (), & ap\ 17\ ap, & \epsilon\rangle \\
\langle \epsilon, & (y \rightarrow 23,\ x \rightarrow 42), & x\ prim_!, & \langle 25,\ (),\ 17\ ap,\ \epsilon\rangle\rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Abstraktion-Terminierungs-Regel

Abstraktion-Terminierungs-Regel

$$\langle a\, S,\ E,\ \epsilon,\ \langle S',\ E',\ C',\ D' \rangle \rangle \rightarrow_{SECD} \langle a\, S',\ E',\ C',\ D' \rangle$$

- Der Code ist **abgearbeitet**, aber der Dump ist **nicht** leer
- Alle Einträge aus dem Dump werden **zurückgeschrieben**
- Der Stack wird dabei **nicht** überschrieben, sondern **ergänzt**

Abstraktion-Terminierungs-Regel

Abstraktion-Terminierungs-Regel

$$\langle a\, S,\ E,\ \epsilon,\ \langle S',\ E',\ C',\ D' \rangle \rangle \rightarrow_{SECD} \langle a\, S',\ E',\ C',\ D' \rangle$$

- Der Code ist **abgearbeitet**, aber der Dump ist **nicht** leer
- Alle Einträge aus dem Dump werden **zurückgeschrieben**
- Der Stack wird dabei **nicht** überschrieben, sondern **ergänzt**

Beispiel

$$\begin{array}{llll}
\vdots & \vdots & \vdots & \vdots \\
\langle 15, & (y \rightarrow 23), & \epsilon, & \langle 25,\ (),\ prim_+,\ \epsilon \rangle \rangle \\
\langle 15\ 25, & (), & prim_+, & \epsilon \rangle \\
\vdots & \vdots & \vdots & \vdots
\end{array}$$

Programm-Terminierungs-Regel

Programm-Terminierungs-Regel

$\langle S,\ E,\ \epsilon,\ \epsilon \rangle \rightarrow_{SECD} \langle S,\ E,\ \epsilon,\ \epsilon \rangle$

- Der Code ist **abgearbeitet** und der Dump **leer**
- Das Programm ist **fertig ausgewertet** und das **Ergebnis** liegt als oberstes Element auf dem Stack

Programm-Terminierungs-Regel

Programm-Terminierungs-Regel

$\langle S,\ E,\ \epsilon,\ \epsilon \rangle \rightarrow_{SECD} \langle S,\ E,\ \epsilon,\ \epsilon \rangle$

- Der Code ist **abgearbeitet** und der Dump **leer**
- Das Programm ist **fertig ausgewertet** und das **Ergebnis** liegt als oberstes Element auf dem Stack

Beispiel

$$\begin{array}{cccc} \vdots & \vdots & \vdots & \vdots \\ \langle 15, & (x \rightarrow 23), & \epsilon, & \epsilon \rangle \end{array}$$

Kurzes Beispiel

(1)

S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

Kurzes Beispiel

(1)

S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

---

(2)

S: $\langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: $23\ ap$
D: ϵ

Kurzes Beispiel

(1)
S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

---

(2)
S: $\langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: $23\ ap$
D: ϵ

---

(3)
S: $23\ \langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: ap
D: ϵ

Kurzes Beispiel

(1)
S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

---

(2)
S: $\langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: $23\ ap$
D: ϵ

---

(3)
S: $23\ \langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: ap
D: ϵ

(4)
S: ϵ
E: $(x \to 23)$
C: x
D: $\langle\epsilon,\ (),\ \epsilon,\ \epsilon\rangle$

Kurzes Beispiel

(1)
S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

(2)
S: $\langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: $23\ ap$
D: ϵ

(3)
S: $23\ \langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: ap
D: ϵ

(4)
S: ϵ
E: $(x \to 23)$
C: x
D: $\langle\epsilon,\ (),\ \epsilon,\ \epsilon\rangle$

(5)
S: 23
E: $(x \to 23)$
C: ϵ
D: $\langle\epsilon,\ (),\ \epsilon,\ \epsilon\rangle$

Kurzes Beispiel

(1)
S: ϵ
E: $()$
C: $\langle x,\ x\rangle\ 23\ ap$
D: ϵ

(2)
S: $\langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: $23\ ap$
D: ϵ

(3)
S: $23\ \langle\langle x,\ x\rangle,\ ()\rangle$
E: $()$
C: ap
D: ϵ

(4)
S: ϵ
E: $(x \to 23)$
C: x
D: $\langle\epsilon,\ (),\ \epsilon,\ \epsilon\rangle$

(5)
S: 23
E: $(x \to 23)$
C: ϵ
D: $\langle\epsilon,\ (),\ \epsilon,\ \epsilon\rangle$

(6)
S: 23
E: $()$
C: ϵ
D: ϵ

Aufgabe zum Selbstmachen

$$8\ \langle x,\ 4\ \langle y,\ x\ y\ prim_{+}\rangle\ 1\ ap\ prim_{+}\rangle\ 3\ ap\ prim_{\star}$$

Aufgabe zum Selbstmachen

$$8\ \langle x,\ 4\ \langle y,\ x\ y\ prim_{+}\rangle\ 1\ ap\ prim_{+}\rangle\ 3\ ap\ prim_{\star}$$

Ergebnis: 64

Eigenschaften der SECDH

- Die SECDH Maschine erlaubt die Modellierung von **Zuständen** im Lambda-Kalkül
- Erweitert die SECD Maschine um einen Speicher, der **Heap** genannt wird
- Die SECDH Maschine arbeitet mit **Adressen**

Adressen

- Eine **Speicherzelle** ist ein Ort im Speicher, der einen Wert aufnimmt, der auch wieder **verändert** werden kann.
- Jede Speicherzelle wird durch eine **Adresse** identifiziert.
- Bei einer Adresse handelt es sich i.d.R. um eine **Zahl**.
- Auf den folgenden Folien werden Adressen durch spitze Klammern gekennzeichnet, z.B. $< 1 >$.

Erweiterung der Zwischenrepäsentation

- Die bisher bestehenden Regeln für die Umwandlung von Lambda-Ausdrücken in die Zwischenrepäsentation werden um folgende Regel **ergänzt**:

5. **set!-Ausdrücke:**
   $[\![(set!\ v\ e')]\!]_{SECDH} = v\,[\![e']\!]_{SECDH}\ :=$

Auswertung mit SECDH

Die Auswertungsregeln für die SECDH-Maschine sind **analog** zu den Regeln für die SECD-Maschine, sie unterscheiden sich in **zwei Punkten**:

- Der Heap ist Bestandteil des **Zustandes**. Anders als das Environment, wird er nicht bei der Bildung neuer Closures gespeichert.
- Jedes neue **Zwischenergebis** wird bei einer neuen Adresse im Heap abgelegt. Auf dem Stack landen die **Adressen**der Zwischenergebnisse.

Auswertungsregeln SECDH

- **Konstanten-Regel:**
  $\langle S, E, bC, D, H \rangle \rightarrow_{SECDH} \langle aS, E, C, D, H[a \mapsto b] \rangle$

Auswertungsregeln SECDH

- **Konstanten-Regel:**
  $\langle S,\ E,\ bC,\ D,\ H\rangle \rightarrow_{SECDH} \langle aS,\ E,\ C,\ D,\ H[a \mapsto b]\rangle$

- **Variablen-Regel:**
  $\langle S,\ E,\ vC,\ D,\ H\rangle \rightarrow_{SECDH} \langle E(v)S,\ E,\ C,\ D,\ H\rangle$

Auswertungsregeln SECDH

- **Konstanten-Regel:**
  $\langle S,\ E,\ bC,\ D,\ H\rangle \rightarrow_{SECDH} \langle aS,\ E,\ C,\ D,\ H[a \mapsto b]\rangle$

- **Variablen-Regel:**
  $\langle S,\ E,\ vC,\ D,\ H\rangle \rightarrow_{SECDH} \langle E(v)S,\ E,\ C,\ D,\ H\rangle$

- **Primitivum-Regel:**
  $\langle a_n \ldots a_1 S,\ E,\ prim_{O^n} C,\ D,\ ,\ H\rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H[a \mapsto O(H(a_1) \ldots H(a_n))]\rangle$

Auswertungsregeln SECDH

- **Konstanten-Regel:**
  $\langle S,\ E,\ bC,\ D,\ H\rangle \rightarrow_{SECDH} \langle aS,\ E,\ C,\ D,\ H[a \mapsto b]\rangle$

- **Variablen-Regel:**
  $\langle S,\ E,\ vC,\ D,\ H\rangle \rightarrow_{SECDH} \langle E(v)S,\ E,\ C,\ D,\ H\rangle$

- **Primitivum-Regel:**
  $\langle a_n \ldots a_1 S,\ E,\ prim_{O^n} C,\ D,\ ,\ H\rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H[a \mapsto O(H(a_1) \ldots H(a_n))]\rangle$

- **Zuweisungs-Regel:**
  $\langle a_1 a_0 S,\ E,\ := C,\ D,\ H\rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H[a_0 \mapsto H(a_1)][a \mapsto void]\rangle$

Auswertungsregeln SECDH

- **Abstraktions-Regel:**
  $\langle S,\ E,\ \langle v,\ C' \rangle\ C,\ D,\ H \rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H\,[a \mapsto \langle \langle v,\ C' \rangle,\ E \rangle] \rangle$

Auswertungsregeln SECDH

- **Abstraktions-Regel:**
  $\langle S,\ E,\ \langle v,\ C' \rangle\ C,\ D,\ H \rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H\,[a \mapsto \langle \langle v,\ C' \rangle,\ E \rangle] \rangle$

- **Applikations-Regel:**
  $\langle a_1 a_0 S,\ E,\ apC,\ D,\ H\,[a_0 \mapsto \langle \langle v,\ C' \rangle,\ E \rangle] \rangle \rightarrow_{SECDH}$
  $\langle \varepsilon,\ E' \cup \{v \mapsto a\},\ C',\ (S, E, C, D),\ H\,[a \mapsto H(a_1)] \rangle$

Auswertungsregeln SECDH

- **Abstraktions-Regel:**
  $\langle S,\ E,\ \langle v,\ C' \rangle\ C,\ D,\ H \rangle \rightarrow_{SECDH}$
  $\langle aS,\ E,\ C,\ D,\ H\,[a \mapsto \langle \langle v,\ C' \rangle,\ E \rangle] \rangle$

- **Applikations-Regel:**
  $\langle a_1\, a_0 S,\ E,\ apC,\ D,\ H\,[a_0 \mapsto \langle \langle v,\ C' \rangle,\ E \rangle] \rangle \rightarrow_{SECDH}$
  $\langle \varepsilon,\ E' \cup \{v \mapsto a\},\ C',\ (S, E, C, D),\ H\,[a \mapsto H(a_1)] \rangle$

- **Abstraktions-Terminierungs-Regel:**
  $\langle a\, S,\ E,\ \epsilon,\ (S', E', C', D'),\ H \rangle \rightarrow_{SECDH} \langle a\, S',\ E',\ C',\ D',\ H \rangle$

Beispiel

	S:	ϵ
	E:	$()$
(1)	C:	$\langle a,\ a\ 23\ :=\ a\ 5\ prim_{+} \rangle\ 0\ ap$
	D:	ϵ
	H:	ϵ

Beispiel

	S:	ϵ
	E:	$()$
(1)	C:	$\langle a,\ a\ 23\ :=\ a\ 5\ prim_{+} \rangle\ 0\ ap$
	D:	ϵ
	H:	ϵ

	S:	$< 1 >$
	E:	$()$
(2)	C:	$0\ ap$
	D:	ϵ
	H:	$1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_{+} \rangle, ()\rangle$

Beispiel

(1)

S: ϵ
E: $()$
C: $\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle\ 0\ ap$
D: ϵ
H: ϵ

---

(2)

S: $< 1 >$
E: $()$
C: $0\ ap$
D: ϵ
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$

---

(3)

S: $< 2 >< 1 >$
E: $()$
C: ap
D: ϵ
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0$

Beispiel

(3)

S: $< 2 >< 1 >$
E: $()$
C: ap
D: ϵ
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+\rangle, ()\rangle$
$2 \mapsto 0$

---

(4)

S: ϵ
E: $(a \mapsto < 3 >)$
C: $a\ 23\ :=\ a\ 5\ prim_+\ a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+\rangle, ()\rangle$
$2 \mapsto 0$
$3 \mapsto 0$

Beispiel

(4)

S: ϵ
E: $(a \mapsto < 3 >)$
C: $a\ 23\ :=\ a\ 5\ prim_+\ a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0$
$3 \mapsto 0$

---

(5)

S: $< 3 >$
E: $(a \mapsto < 3 >)$
C: $23\ :=\ a\ 5\ prim_+\ a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0$
$3 \mapsto 0$

Beispiel

(5)

S: $< 3 >$
E: $(a \mapsto < 3 >)$
C: $23 \;:=\; a\ 5\ prim_+\ a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23 \;:=\; a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0$
$3 \mapsto 0$

---

(6)

S: $< 4 >< 3 >$
E: $(a \mapsto < 3 >)$
C: $:=\; a\ 5\ prim_+ a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23 \;:=\; a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0$
$3 \mapsto 0$
$4 \mapsto 23$

Beispiel

(6)

S: $< 4 >< 3 >$
E: $(a \mapsto < 3 >)$
C: $:= \ a\ 5\ prim_{+} a$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_{+} \rangle, () \rangle$
$2 \mapsto 0$
$3 \mapsto 0$
$4 \mapsto 23$

---

(7)

S: $< 5 >$
E: $(a \mapsto < 3 >)$
C: $a\ 5\ prim_{+}$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_{+} \rangle, () \rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23$
$4 \mapsto 23$

Beispiel

(7)

S: $< 5 >$
E: $(a \mapsto < 3 >)$
C: $a\ 5\ prim_+$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23$
$4 \mapsto 23$

---

(8)

S: $< 3 >< 5 >$
E: $(a \mapsto < 3 >)$
C: $5\ prim_+$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23$
$4 \mapsto 23$

Beispiel

(8)

S: $< 3 >< 5 >$
E: $(a \mapsto < 3 >)$
C: $5\ prim_+$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23$
$4 \mapsto 23$

---

(9)

S: $< 6 >< 3 >< 5 >$
E: $(a \mapsto < 3 >)$
C: $prim_+$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23 \quad 6 \mapsto 5$
$4 \mapsto 23$

Beispiel

(9)

S: $< 6 >< 3 >< 5 >$
E: $(a \mapsto < 3 >)$
C: $prim_+$
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23 \quad 6 \mapsto 5$
$4 \mapsto 23$

---

(10)

S: $< 7 >< 5 >$
E: $(a \mapsto < 3 >)$
C: ϵ
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle \langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, () \rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23 \quad 6 \mapsto 5$
$4 \mapsto 23 \quad 7 \mapsto 28$

Beispiel

(10)

S: $< 7 >< 5 >$
E: $(a \mapsto < 3 >)$
C: ϵ
D: $\langle \epsilon, (), \epsilon, \epsilon \rangle$
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23 \quad 6 \mapsto 5$
$4 \mapsto 23 \quad 7 \mapsto 28$

---

(11)

S: $< 7 >$
E: $()$
C: ϵ
D: ϵ
H: $1 \mapsto \langle\langle a,\ a\ 23\ :=\ a\ 5\ prim_+ \rangle, ()\rangle$
$2 \mapsto 0 \quad 5 \mapsto void$
$3 \mapsto 23 \quad 6 \mapsto 5$
$4 \mapsto 23 \quad 7 \mapsto 28$

Letzte Folie

Noch Fragen? ;)