

---

# Theorie der Programmierung I

---

Kurt Sieber

Fachbereich Mathematik/Theoretische Informatik  
Universität Siegen

Vorlesung vom 24.11.2004 (Stand: 24.11.2004)

## Small step Semantik

---

(UOP)  $op\ v \rightarrow op^{\mathcal{I}}(v)$  falls  $v \in dom(op^{\mathcal{I}})$

(BOP)  $op\ v_1\ v_2 \rightarrow op^{\mathcal{I}}(v_1, v_2)$  falls  $(v_1, v_2) \in dom(op^{\mathcal{I}})$

(PROJ)  $\#n\ (v_1, \dots, v_k) \rightarrow v_n$  falls  $n \in \{1, \dots, k\}$

(BETA-V)  $(\lambda id : \tau. e)\ v \rightarrow e[v/id]$

(APP-LEFT) 
$$\frac{e_1 \rightarrow e'_1}{e_1\ e_2 \rightarrow e'_1\ e_2}$$

(APP-RIGHT) 
$$\frac{e \rightarrow e'}{v\ e \rightarrow v\ e'}$$

## Small step Semantik

---

(TUPLE) 
$$\frac{e_i \rightarrow e'_i}{(v_1, \dots, v_{i-1}, e_i, \dots, e_k) \rightarrow (v_1, \dots, v_{i-1}, e'_i, \dots, e_k)}$$

(COND-EVAL) 
$$\frac{e_0 \rightarrow e'_0}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \rightarrow \text{if } e'_0 \text{ then } e_1 \text{ else } e_2}$$

(COND-TRUE) 
$$\text{if } \textit{true} \text{ then } e_1 \text{ else } e_2 \rightarrow e_1$$

(COND-FALSE) 
$$\text{if } \textit{false} \text{ then } e_1 \text{ else } e_2 \rightarrow e_2$$

(LET-EVAL) 
$$\frac{e_1 \rightarrow e'_1}{\text{let } id = e_1 \text{ in } e_2 \rightarrow \text{let } id = e'_1 \text{ in } e_2}$$

(LET-EXEC) 
$$\text{let } id = v \text{ in } e \rightarrow e[v/id]$$

## Satz 1.1 (Eindeutigkeit des Übergangsschrittes)

Die small step Semantik ist **deterministisch**, d.h. für jeden Ausdruck  $e$  gibt es **höchstens einen** small step  $e \rightarrow e'$  oder  $e \rightarrow \text{exn}$ .

### Satz 1.2 (Typerhaltung, “Preservation”)

Für jeden small step der Form  $e \rightarrow e'$  gilt:

1.  $free(e') \subseteq free(e)$ , insbesondere  $free(e') = \emptyset$ , wenn  $free(e) = \emptyset$ , d.h. die Abgeschlossenheit eines Ausdrucks bleibt bei jedem small step erhalten.
2. Wenn  $\Gamma \triangleright e :: \tau$ , dann gilt auch  $\Gamma \triangleright e' :: \tau$ , d.h. die Wohlgetyptheit und der Typ eines Ausdruck bleiben bei jedem small step erhalten.

### Satz 1.3 (Existenz des Übergangsschrittes, “Progress”)

Sei  $e$  abgeschlossen und wohlgetypt. Dann ist entweder  $e \in \text{Val}$  (d.h.  $e$  ist schon fertig ausgewertet) oder es existiert ein Übergangsschritt  $e \rightarrow e'$  bzw.  $e \rightarrow \text{exn}$ .

Man beachte, dass dieser Übergangsschritt nach Satz 1.1 dann eindeutig ist.

## Small step Semantik

---

**Satz 1.4 (Typsicherheit, “Safety”)** *Die Berechnung für einen abgeschlossenen wohlgetypten Ausdruck  $e$  kann nur*

1. *divergieren,*
2. *oder mit einem Wert  $v \in Val$  (vom gleichen Typ wie  $e$ ) terminieren,*
3. *oder mit einer Ausnahme  $exn \in Exn$  terminieren.*

Sie kann also *nicht* steckenbleiben. (“Well typed programs don’t go wrong.”)

### **Bemerkung:**

In der bisher definierten Programmiersprache kann auch Divergenz nicht auftreten (was noch zu beweisen wäre). Satz 1.4 ist schon so formuliert, dass er auch später noch gilt (in einer Sprache mit Rekursion).

---