

G54FOP: Lecture 6

Operational Semantics III: State

Henrik Nilsson

University of Nottingham, UK

Small Expression Language: Terms

$t \rightarrow$

true

false

if t **then** t **else** t

0

succ t

pred t

iszero t

terms:

constant true

constant false

conditional

constant zero

successor

predecessor

zero test

Small Expression Language: Values

$v \rightarrow$

	true	<i>constant true</i>
	false	<i>constant false</i>
	nv	<i>numeric value</i>

$nv \rightarrow$

	0	<i>zero value</i>
	succ nv	<i>successor value</i>

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores: Without state:

$$t \longrightarrow t'$$

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

$$t \mid \mu \longrightarrow t' \mid \mu'$$

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

$$t \mid \mu \longrightarrow t' \mid \mu'$$

- $l \in \mathcal{L}$: Uninterpreted set of locations (addresses) with equality.

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

$$t \mid \mu \longrightarrow t' \mid \mu'$$

- $l \in \mathcal{L}$: Uninterpreted set of locations (addresses) with equality.
- $\mu \in \mathcal{L} \rightarrow v$: Store: map from location to value.

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

$$t \mid \mu \longrightarrow t' \mid \mu'$$

- $l \in \mathcal{L}$: Uninterpreted set of locations (addresses) with equality.
- $\mu \in \mathcal{L} \rightarrow v$: Store: map from location to value.
- $\mu(l)$: Lookup value at location l .

Introducing State: Key Ideas

- μ : Store (state, memory); can be read/updated.
- Evaluation relation relates terms **and stores** to new terms and stores:

$$t \mid \mu \longrightarrow t' \mid \mu'$$

- $l \in \mathcal{L}$: Uninterpreted set of locations (addresses) with equality.
- $\mu \in \mathcal{L} \rightarrow v$: Store: map from location to value.
- $\mu(l)$: Lookup value at location l .
- $[l \mapsto v]\mu$: Update; like μ except $([l \mapsto v]\mu)(l) = v$

Small Imperative Language (1)

New terms; extends the terms of Small Expression Language:

$t \rightarrow$ *terms:*

...

	unit	<i>constant unit</i>
	new t	<i>allocation</i>
	! t	<i>dereferencing</i>
	t := t	<i>assignment</i>
	l	<i>store location</i>
	t ; t	<i>sequencing</i>

Small Imperative Language (2)

New values; extends the values of Small Expression Language:

$v \rightarrow$ *values:*
...
| **unit** *unit value*
| *l* *store location*

Small Imperative Language (2)

New values; extends the values of Small Expression Language:

v	\rightarrow	<i>values:</i>
\dots		
	unit	<i>unit value</i>
	l	<i>store location</i>

Note: Still an expression language in that every term is an expression that evaluates to a value, even if some expressions have side effects. No separate category of commands.

Homework Lecture 6 (1)

1. Consider the Small Imperative Language. Add a loop construct:

$t \rightarrow$ *terms:*
...
| **while** t **do** t *while loop*

Provide evaluation rule(s) for this construct, assuming the usual semantics of a while loop: repetition of loop body **zero** or more times as long as loop condition is true.

Hint: Make use of what you have!

Homework Lecture 6 (2)

2. As mentioned, our language is still an expression language where expressions may have side effects. Design a new language (syntax and op. sem.) by separating the terms into

- expressions: do not have side effects
- commands: have side effects

and making any other changes you see fit.

Don't worry too much if the resulting language isn't "useful" (the Small Imperative Language isn't really). Can the evaluation rules for expressions somehow be simplified by exploiting that expressions do not have side effects?