G54FOP: Lecture 6

Operational Semantics III: State

Henrik Nilsson

University of Nottingham, UK

Small Expression Language: Terms

t	$\rightarrow$		*terms:*
		`true`	*constant true*
	$\mid$	`false`	*constant false*
	$\mid$	`if` t `then` t `else` t	*conditional*
	$\mid$	`0`	*constant zero*
	$\mid$	`succ` t	*successor*
	$\mid$	`pred` t	*predecessor*
	$\mid$	`iszero` t	*zero test*

Small Expression Language: Values

v	$\rightarrow$		*values:*
		`true`	*constant true*
	$\mid$	`false`	*constant false*
	$\mid$	nv	*numeric value*
nv	$\rightarrow$		*numeric values:*
		`0`	*zero value*
	$\mid$	`succ` nv	*successor 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.
- $[l \mapsto v]\mu$: 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:*
		$\ldots$	
	$\mid$	`unit`	*constant unit*
	$\mid$	`new` t	*allocation*
	$\mid$	`!` t	*dereferencing*
	$\mid$	t `:=` t	*assignment*
	$\mid$	l	*store location*
	$\mid$	t `;` t	*sequencing*

Small Imperative Language (2)

New values; extends the values of Small Expression Language:

v	$\rightarrow$		*values:*
		$\ldots$	
	$\mid$	`unit`	*unit value*
	$\mid$	l	*store location*

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:*
		$\ldots$	
	$\mid$	`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?