Value Passing CCS

Content

- Syntax
- Semantics
- Examples

Syntax of Value Passing CCS

The abstract grammar is defined as follows

E := **0**	nil process
X	process variable
λ.E	prefix
E | E'	concurrent composition
E\L	restriction
E[f]	re-labeling
E + E'	non-determinism
if b **then** E	condition
A	recursion

Where λ is either a(x) or $\overline{a}$(e) or τ, L $\subseteq$ N

f: $\mathcal{A} \to \mathcal{A}$, a re-labeling function from $\mathcal{A}$ to $\mathcal{A}$

V: a value set

The LTS of Value Passing CCS

Prefix

$$\frac{}{a(x).E \xrightarrow{a(v)} E\{v/x\}} \qquad \frac{v = val(e)}{\overline{a}(e).E \xrightarrow{\overline{a}(v)} E} \qquad \frac{}{\tau.E \xrightarrow{\tau} E}$$

Composition

$$\frac{E \xrightarrow{\lambda} E'}{E \mid F \xrightarrow{\lambda} E' \mid F} \qquad \frac{F \xrightarrow{\lambda} F'}{E \mid F \xrightarrow{\lambda} E \mid F'} \qquad \frac{E \xrightarrow{\alpha(v)} E' \quad F \xrightarrow{\overline{\alpha}(v)} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

Restriction

$$\frac{E \xrightarrow{\lambda} E' \quad n(\lambda) \cap L = \emptyset}{E \backslash L \xrightarrow{\lambda} E' \backslash L}$$

The LTS of Value Passing CCS

Relabelling

$$\frac{E \xrightarrow{\lambda} E'}{E[f] \xrightarrow{f(\lambda)} E'[f]}$$

Choice

$$\frac{E \xrightarrow{\lambda} E'}{E+F \xrightarrow{\lambda} E'} \qquad \frac{F \xrightarrow{\lambda} F'}{E+F \xrightarrow{\lambda} F'}$$

Condition

$$\frac{E \xrightarrow{\lambda} E' \qquad \top = val(b)}{\textbf{if } \mathrm{b} \textbf{ then } \mathbf{E} \xrightarrow{\lambda} \mathbf{E'}}$$

Recursion

$$\frac{E\{A/X\} \xrightarrow{\lambda} E'}{A \xrightarrow{\lambda} E'} A \stackrel{\text{def}}{=} E\{A/X\}$$

Examples

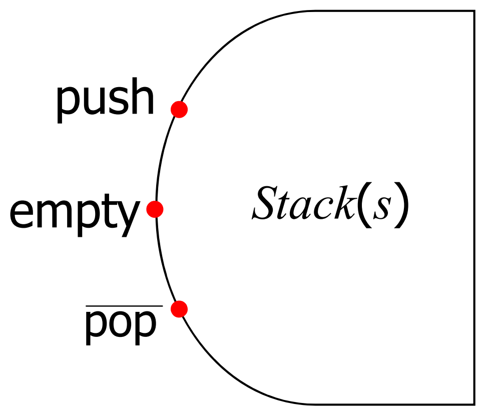
- a(x).if x=0 then b(y).P else c(z).Q
- (a(x).if x=0 then b(y).P else c(z).Q) | $\bar{a}$0.$\bar{b}$3

- Note： the two-armed conditional if b then E else E′ can be defined as

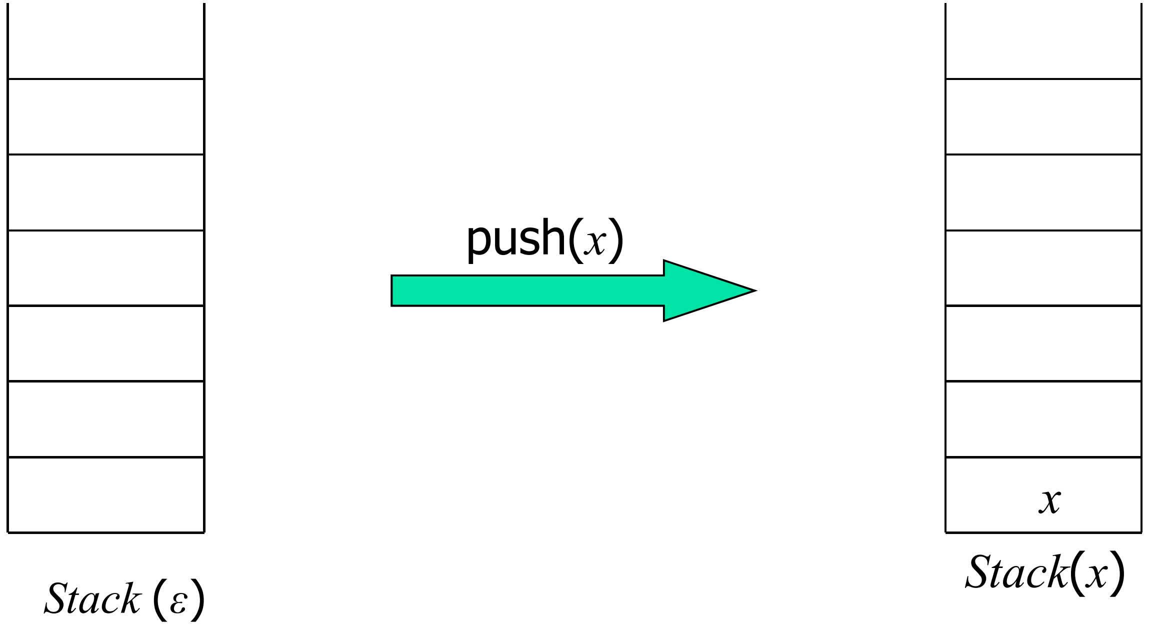
  (if b then E) + (if $\neg$b then E′)

Stack

- A state of a stack is a sequence of values
- Assume that Stack(v_1, … , v_n) specifies a stack containing n values, with v_1 topmost

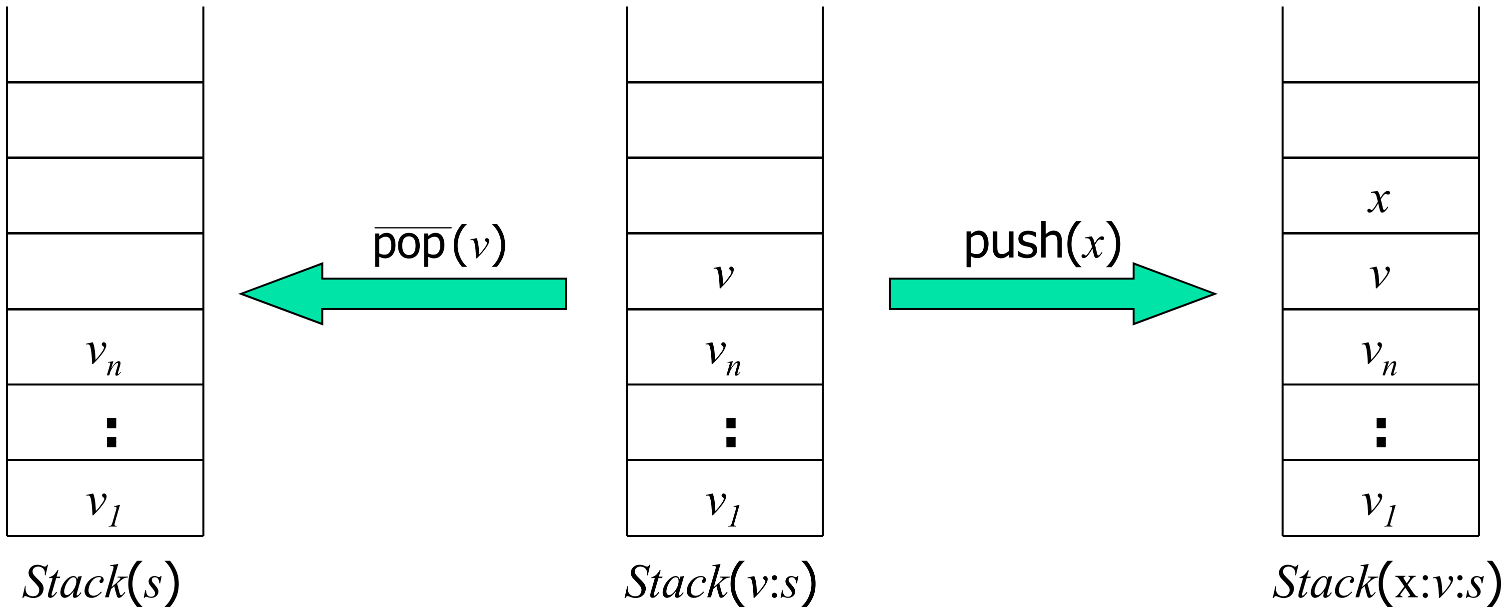


Stack



where ε stands for the empty sequence

Stack



where s for the value sequence v_1, … , v_n ;

: for sequence concatenation

Stack

Specification

$$Stack(\varepsilon) \stackrel{\text{def}}{=} push(x).Stack(x) + empty.Stack(\varepsilon)$$

$$Stack(v : s) \stackrel{\text{def}}{=} push(x).Stack(x : v : s) + \overline{pop}(v).Stack(s)$$

where ε for the empty sequence;

s for an arbitrary value sequence;

: for sequence concatenation

Stack

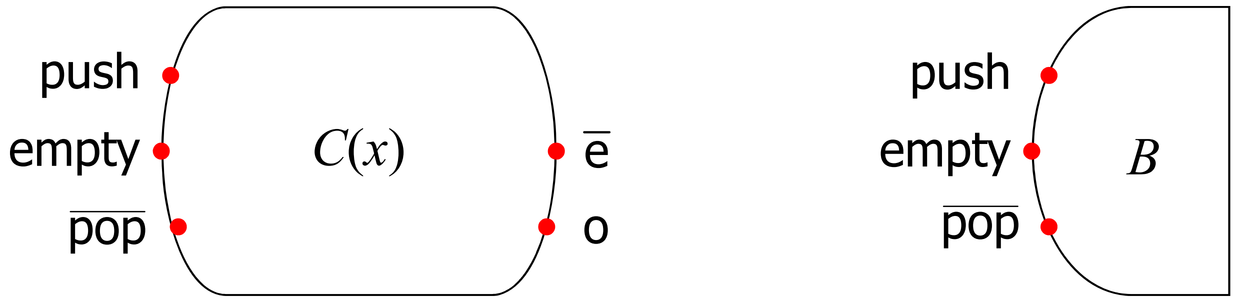
Implementation

$$Cells\langle v_1, \ldots, v_n \rangle \stackrel{\mathrm{def}}{=} C(v_1)^\frown C(v_2)^\frown \cdots {}^\frown C(v_n)^\frown B$$

where

$$\begin{aligned} C(x) &\stackrel{\mathrm{def}}{=} push(y).(C(y)^\frown C(x)) + \overline{pop}(x).D \\ D &\stackrel{\mathrm{def}}{=} o(x).C(x) + \overline{e}.B \\ B &\stackrel{\mathrm{def}}{=} push(y).(C(y)^\frown B) + empty.B \end{aligned}$$

Stack



$$C(y)^\frown C(x) \stackrel{\text{def}}{=} (C(y)\{e'/e, o'/o\} \mid C(x)\{e'/empty, o'/pop, i'/push\})\backslash\{e', o', i'\}$$

$$C(x)^\frown B \stackrel{\text{def}}{=} (C(x)\{e'/e, o'/o\} \mid B\{e'/empty, o'/pop, i'/push\})\backslash\{e', o', i'\}$$