Pure CCS

Content

- Syntax
- Semantics
- Counter

What is CCS about?

- CCS, The Calculus of Communicating Systems
  - a formal language that models the operational behaviors of the communications of concurrent objects

How is CCS Defined?

- The definition of CCS follows the standard approach of defining a programming language
- The standard approach is widely used in computer science to define various calculi

Definition of Calculus

- Defining Programming Languages
- Syntax
  - Syntactical entity
  - Alphabet
  - BNF (Backus-Naur Form 巴科斯范式)
- Semantics
  - Operational behavior
  - Labeled Transition System, LTS

Operational Semantics

- A way to give meaning to computation in a mathematically rigorous way
  - Other approaches to providing a formal semantics of computation include *axiomatic semantics* and *denotational semantics*
- Classified into two categories
  - structural operational semantics (small-step semantics)
    - formally describe how the *individual steps* of a computation take place in a computer-based system
  - natural semantics (big-step semantics)
    - describe how the *overall results* of the executions are obtained

Intuition about the Semantics

- $A \stackrel{def}{=} a.A'$, $A' \stackrel{def}{=} \overline{c}.A$;
- $B \stackrel{def}{=} c.B'$, $B' \stackrel{def}{=} \overline{b}.B$
- Example

$Na + Cl \rightarrow NaCl$

Preliminary Notations

Some Notations:

$\mathcal{N}$ is a Set of Names, ranged over by small letters

$\overline{\mathcal{N}}$ is a Set $\{\overline{a} \mid a \in \mathcal{N}\}$ of Co-Names

$\mathcal{L}$ is the Set $\mathcal{N} \cup \overline{\mathcal{N}}$ of all names, ranged over by α

$\mathcal{A}$ is the Set $\mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$ of Labels, ranged over by λ

Let $\overline{\alpha}$ be defined as follows:

$$\overline{\alpha} \stackrel{\text{def}}{=} \begin{cases} a, & \text{if } \alpha = \overline{a} \\ \overline{a}, & \text{if } \alpha = a \end{cases}$$

A function $f : \mathcal{A} \longrightarrow \mathcal{A}$ is a relabeling function if $f(\tau) = \tau$ and $f(\overline{\alpha}) = \overline{f(\alpha)}$

Action

- External
- Internal

Action

- External
  - Potential to communicate
- Internal

Action

- External
  - Potential to communicate
- Internal
  - Communication in a process
  - Communication between processes

Syntax of CCS

The abstract grammar is defined as follows:

$E := \mathbf{0}$	nil process
X	process variable
$\lambda.E$	prefix
$E \mid E'$	concurrent composition
$E \backslash L$	restriction
$E[f]$	re-labeling
$E + E'$	non-determinism
A	recursion

Where λ is either a or $\overline{a}$ or τ, $L \subseteq \mathcal{N}$

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

Let $\mathcal{E}$ be the set of process expressions and $\mathcal{P}$ be the set of processes

Intuitive Explanation

- ***0***
  - The process that can never do anything in any environment
- X
  - Process variable
- $\lambda.E$
  - The process that must first perform the action λ and then evolves as E
- $E|E'$
  - The processes E and E' may evlove independently or communicate through common channels

Intuitive Explanation

- $E\backslash L$
  - The process that can do whatever E may do as long as the action are not restricted by L
- $E[f]$
  - Whatever E can do, $E[f]$ can do with action renamed by f
- $E+E'$
  - The process that acts either as E or as E'
- A
  - The recursive process that may never stop

What is an LTS?

An LTS is a Triple

$$\langle \mathcal{S}, \mathcal{T}, \rightarrow \rangle$$

where

$\mathcal{S}$ is a Set of States

$\mathcal{T}$ is a Set of Transitions

$\rightarrow \subseteq \mathcal{S} \times \mathcal{T} \times \mathcal{S}$

We write $a \xrightarrow{t} b$ if $\langle a, t, b \rangle \in \rightarrow$ and the intuition is that

a evolves to b by making a t transition

E.G. automata

An Example: The λ-Calculus

- Term

$$
\begin{array}{lll}
t ::= & x & \text{variable} \\
& tt' & \text{application} \\
& \lambda x.t & \text{abstraction}
\end{array}
$$

- Reduction

$$(\lambda x.t)s \to t\{s/x\}$$

If $t \to t'$ then $st \to st'$

If $t \to t'$ then $ts \to t's$

If $t \to t'$ then $\lambda x.t \to \lambda x.t'$

The LTS of CCS

Prefix

$$\frac{}{\lambda.E \xrightarrow{\lambda} E}$$

Composition

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

Restriction

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

Relabelling

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

The LTS of CCS, Continued

Choice

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

Recursion

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

Using the Rules

- $((a.E+b.0)|\bar{a}.F)\backslash a \xrightarrow{\tau} (E|F)\backslash a$
- $A \overset{def}{=} a.A+\tau.b.A$. Show that $A \xrightarrow{\tau} b.A$
- $A \overset{def}{=} a.A'$, $A' \overset{def}{=} \bar{c}.A$; $B \overset{def}{=} c.B'$, $B' \overset{def}{=} \bar{b}.B$

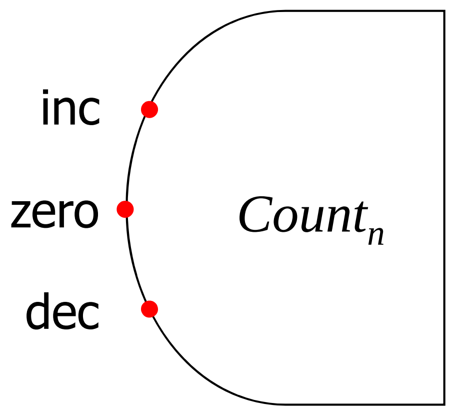
  Show that $(A|B)\backslash c \xrightarrow{a} (A'|B)\backslash c \xrightarrow{\tau} (A|B')\backslash c$
- A[f], where f maps a onto b

Examples

- Readers-Writers
- Dining philosophers problem

Counter

- A counter is able to hold any natural number
- The nature numbers as a state of a counter
- Increment and decrement actions



Counter

Specification

$$Count_0 \stackrel{\text{def}}{=} inc.Count_1 + zero.Count_0$$

$$Count_{n+1} \stackrel{\text{def}}{=} inc.Count_{n+2} + dec.Count_n \quad (n \geq 0)$$

Counter

Implementation

$$C^{(n)} = \overbrace{C^\frown \cdots {}^\frown C}^{n \text{ times}}{}^\frown B$$

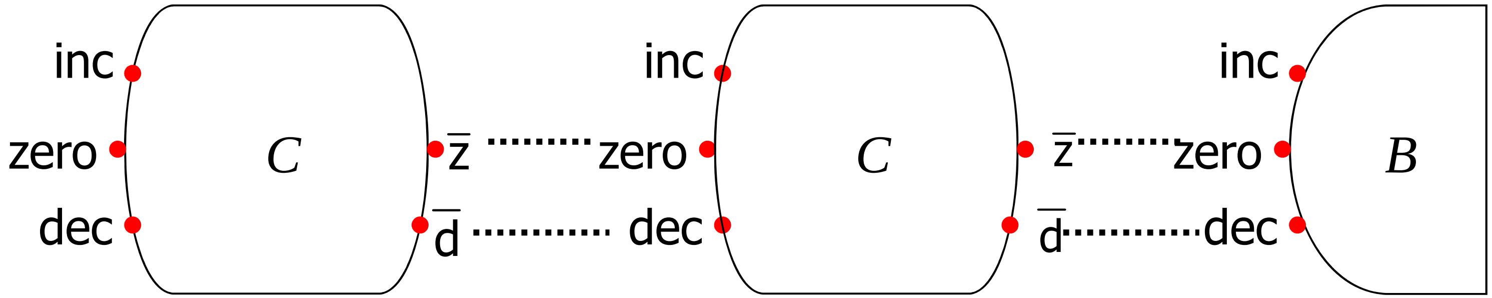
where

$$C \stackrel{\text{def}}{=} inc.(C^\frown C) + dec.D$$

$$D \stackrel{\text{def}}{=} \overline{d}.C + \overline{z}.B$$

$$B \stackrel{\text{def}}{=} inc.(C^\frown B) + zero.B$$

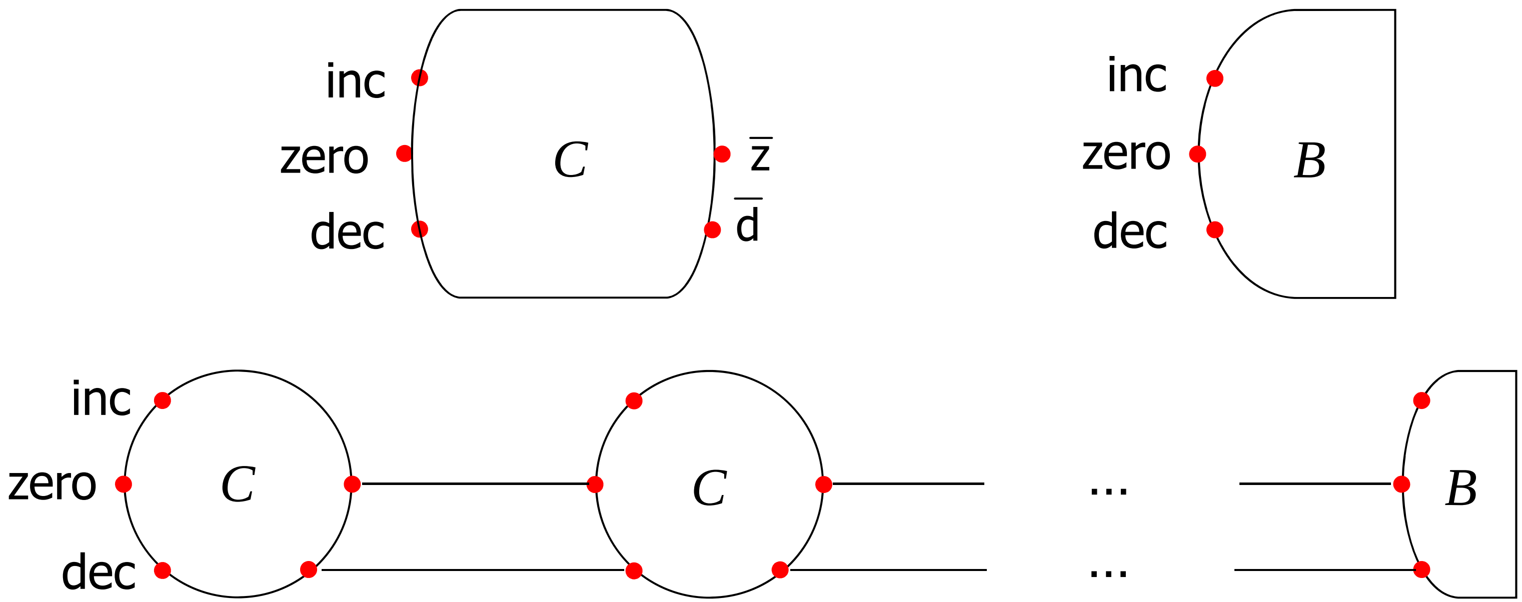
Counter



$$C \frown C \stackrel{\text{def}}{=} (C\{z'/z, d'/d\} \mid C\{z'/zero, d'/dec, i'/inc\}) \backslash \{z', d', i'\}$$

$$C \frown B \stackrel{\text{def}}{=} (C\{z'/z, d'/d\} \mid B\{z'/zero, d'/dec, i'/inc\}) \backslash \{z', d', i'\}$$

Counter



Exercise: Prove that for all $n \geq 0$, $C^{(n)} = Count_n$.