Communication and Concurrency
Lecture 12

Colin Stirling (cps)

School of Informatics

28th October 2013

Notation

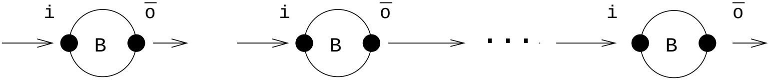
- Assume P contains port $\overline{b}$ and Q contains port a

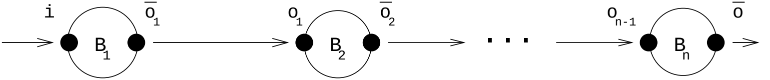
Notation

- Assume P contains port $\overline{b}$ and Q contains port a
- Define $P \frown Q$ linking $\overline{b}$ to a: $(P[c/b] \mid Q[c/a])\backslash\{c\}$
  where c is a new port (not contained in P or Q)

Notation

- Assume P contains port $\overline{b}$ and Q contains port a
- Define $P \frown Q$ linking $\overline{b}$ to a: $(P[c/b] \mid Q[c/a])\backslash\{c\}$ where c is a new port (not contained in P or Q)
- Example: buffers $\mathtt{B} \stackrel{\text{def}}{=} \mathtt{i}(x).\overline{\mathtt{o}}(x).\mathtt{B}$

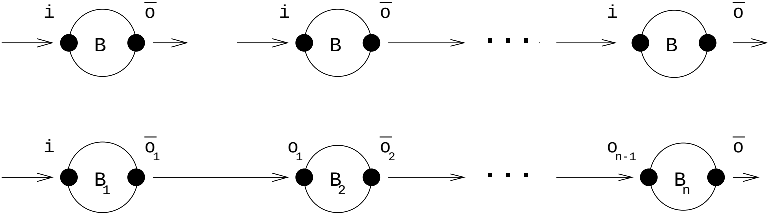




$$
\begin{array}{lcl}
\mathtt{B}_1 & \equiv & \mathtt{B}[\mathtt{o}_1/\mathtt{o}] \\
\mathtt{B}_{j+1} & \equiv & \mathtt{B}[\mathtt{o}_j/\mathtt{i}, \mathtt{o}_{j+1}/\mathtt{o}] \quad 1 \leq j < n-1 \\
\mathtt{B}_n & \equiv & \mathtt{B}[\mathtt{o}_{n-1}/\mathtt{i}]
\end{array}
$$

Notation

- Assume P contains port $\overline{b}$ and Q contains port a
- Define $P \frown Q$ linking $\overline{b}$ to a: $(P[c/b] \mid Q[c/a])\backslash\{c\}$
  where c is a new port (not contained in P or Q)
- Example: buffers $\mathtt{B} \stackrel{\text{def}}{=} \mathtt{i}(x).\overline{\mathtt{o}}(x).\mathtt{B}$



$$
\begin{array}{lcll}
\mathtt{B}_1 & \equiv & \mathtt{B}[\mathtt{o}_1/\mathtt{o}] & \\
\mathtt{B}_{j+1} & \equiv & \mathtt{B}[\mathtt{o}_j/\mathtt{i}, \mathtt{o}_{j+1}/\mathtt{o}] & 1 \leq j < n-1 \\
\mathtt{B}_n & \equiv & \mathtt{B}[\mathtt{o}_{n-1}/\mathtt{i}] &
\end{array}
$$

- Redo as n Bs with $\overline{\mathtt{o}}$ linking $\mathtt{i}$: $B \frown B \frown \ldots \frown B$

Sorting machine example

- Where a system of size $n + 1$ is defined in terms of a system of size n. (From Milner's book 136ff.)

Sorting machine example

- Where a system of size $n+1$ is defined in terms of a system of size n. (From Milner's book 136ff.)
- Want a sorter $\texttt{Sorter}_n$, $n \geq 0$, capable of sorting n-length sequences of positive integers

Sorting machine example

- Where a system of size $n + 1$ is defined in terms of a system of size n. (From Milner's book 136ff.)
- Want a sorter $\mathtt{Sorter}_n$, $n \geq 0$, capable of sorting n-length sequences of positive integers
- Assume $\mathtt{Sorter}_n$ has ports $\mathtt{in}$, $\overline{\mathtt{out}}$

Sorting machine example

- Where a system of size $n+1$ is defined in terms of a system of size n. (From Milner's book 136ff.)
- Want a sorter $\texttt{Sorter}_n$, $n \geq 0$, capable of sorting n-length sequences of positive integers
- Assume $\texttt{Sorter}_n$ has ports $\texttt{in}$, $\overline{\texttt{out}}$
- It accepts exactly n integers one by one at port $\texttt{in}$;

Sorting machine example

- Where a system of size $n+1$ is defined in terms of a system of size n. (From Milner's book 136ff.)
- Want a sorter $\mathtt{Sorter}_n$, $n \geq 0$, capable of sorting n-length sequences of positive integers
- Assume $\mathtt{Sorter}_n$ has ports $\mathtt{in}$, $\overline{\mathtt{out}}$
- It accepts exactly n integers one by one at port $\mathtt{in}$;
- Then it delivers them one by one in descending order at $\overline{\mathtt{out}}$, terminated by a zero

Sorting machine example

- Where a system of size $n+1$ is defined in terms of a system of size n. (From Milner's book 136ff.)
- Want a sorter $\texttt{Sorter}_n$, $n \geq 0$, capable of sorting n-length sequences of positive integers
- Assume $\texttt{Sorter}_n$ has ports $\texttt{in}$, $\overline{\texttt{out}}$
- It accepts exactly n integers one by one at port $\texttt{in}$;
- Then it delivers them one by one in descending order at $\overline{\texttt{out}}$, terminated by a zero
- And returns to start state

Sorting machine specification

- A multiset is a set with possibly multiple elements

$$\{1,2,1\} = \{2,1,1\} \neq \{1,2\}$$

S ranges over multisets of integers and $\max(S)$ $\min(S)$ are maximum and minimum elements of S

Sorting machine specification

- A multiset is a set with possibly multiple elements

$$\{1,2,1\} = \{2,1,1\} \neq \{1,2\}$$

  S ranges over multisets of integers and $\max(S)$ $\min(S)$ are maximum and minimum elements of S

- Specification of sorter

$$\begin{array}{lcl}
\mathtt{Spec}_n & \stackrel{\text{def}}{=} & \mathtt{in}(x_1)\ldots\mathtt{in}(x_n).\mathtt{Hold}_n(\{x_1,\ldots,x_n\}) \\
\mathtt{Hold}_n(S) & \stackrel{\text{def}}{=} & \overline{\mathtt{out}}(\max(S)).\mathtt{Hold}_n(S - \{\max(S)\}) \\
 & & S \neq \emptyset \\
\mathtt{Hold}_n(\emptyset) & \stackrel{\text{def}}{=} & \overline{\mathtt{out}}(0).\mathtt{Spec}_n
\end{array}$$

Sorting machine specification

- A multiset is a set with possibly multiple elements

$$\{1,2,1\} = \{2,1,1\} \neq \{1,2\}$$

S ranges over multisets of integers and $\max(S)$ $\min(S)$ are maximum and minimum elements of S

- Specification of sorter

$$\begin{array}{lcl}
\mathtt{Spec}_n & \stackrel{\text{def}}{=} & \mathtt{in}(x_1)\ldots\mathtt{in}(x_n).\mathtt{Hold}_n(\{x_1,\ldots,x_n\}) \\
\mathtt{Hold}_n(S) & \stackrel{\text{def}}{=} & \overline{\mathtt{out}}(\max(S)).\mathtt{Hold}_n(S-\{\max(S)\}) \\
 & & S \neq \emptyset \\
\mathtt{Hold}_n(\emptyset) & \stackrel{\text{def}}{=} & \overline{\mathtt{out}}(0).\mathtt{Spec}_n
\end{array}$$

- Alternatively assuming $y_1 \geq \ldots \geq y_n$

$\mathtt{Hold}_n(\{y_1,\ldots,y_n\}) \stackrel{\text{def}}{=} \overline{\mathtt{out}}(y_1)\ldots\overline{\mathtt{out}}(y_n).\overline{\mathtt{out}}(0).\mathtt{Spec}_n$

Sorting machine implementation I

- Use n simple cells C and a barrier cell B

Sorting machine implementation I

- Use n simple cells C and a barrier cell B
- C has ports `in`, $\overline{\texttt{down}}$, `up`, $\overline{\texttt{out}}$; B just has `in`, $\overline{\texttt{out}}$

Sorting machine implementation I

- Use n simple cells C and a barrier cell B
- C has ports $\mathtt{in}$, $\overline{\mathtt{down}}$, $\mathtt{up}$, $\overline{\mathtt{out}}$; B just has $\mathtt{in}$, $\overline{\mathtt{out}}$
- Notation: $C \frown C$ where $\overline{\mathtt{down}}$ in first C is linked to $\mathtt{in}$ of second C and $\mathtt{up}$ of first C is linked to $\overline{\mathtt{out}}$ of second C and then these ports are internalised (restricted upon)

Sorting machine implementation I

- Use n simple cells C and a barrier cell B
- C has ports $\mathtt{in}$, $\overline{\mathtt{down}}$, $\mathtt{up}$, $\overline{\mathtt{out}}$; B just has $\mathtt{in}$, $\overline{\mathtt{out}}$
- Notation: $C \frown C$ where $\overline{\mathtt{down}}$ in first C is linked to $\mathtt{in}$ of second C and $\mathtt{up}$ of first C is linked to $\overline{\mathtt{out}}$ of second C and then these ports are internalised (restricted upon)
- $\mathtt{Sorter}_n \stackrel{\mathrm{def}}{=} C \frown \ldots \frown C \frown B$ (n Cs)

Sorting machine implementation I

- Use n simple cells C and a barrier cell B
- C has ports $\mathtt{in}$, $\overline{\mathtt{down}}$, $\mathtt{up}$, $\overline{\mathtt{out}}$; B just has $\mathtt{in}$, $\overline{\mathtt{out}}$
- Notation: $C \frown C$ where $\overline{\mathtt{down}}$ in first C is linked to $\mathtt{in}$ of second C and $\mathtt{up}$ of first C is linked to $\overline{\mathtt{out}}$ of second C and then these ports are internalised (restricted upon)
- $\mathtt{Sorter}_n \stackrel{\text{def}}{=} C \frown \ldots \frown C \frown B$ (n Cs)
- We need to define B and C so that: $\mathtt{Sorter}_n \approx \mathtt{Spec}_n$

Sorting machine implementation I

- Use n simple cells C and a barrier cell B
- C has ports $\mathtt{in}$, $\overline{\mathtt{down}}$, $\mathtt{up}$, $\overline{\mathtt{out}}$; B just has $\mathtt{in}$, $\overline{\mathtt{out}}$
- Notation: $C \frown C$ where $\overline{\mathtt{down}}$ in first C is linked to $\mathtt{in}$ of second C and $\mathtt{up}$ of first C is linked to $\overline{\mathtt{out}}$ of second C and then these ports are internalised (restricted upon)
- $\mathtt{Sorter}_n \stackrel{\mathrm{def}}{=} C \frown \ldots \frown C \frown B$ (n Cs)
- We need to define B and C so that: $\mathtt{Sorter}_n \approx \mathtt{Spec}_n$
- Do it inductively
  1. Base Case: $B \approx \mathtt{Spec}_0$
  2. General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n$
  3. Why? $\mathtt{Sorter}_{n+1} \stackrel{\mathrm{def}}{=} C \frown \mathtt{Sorter}_n$

Sorting machine implementation II

- B is straightforward: $B \stackrel{\text{def}}{=} \overline{\texttt{out}}(0).B$

Sorting machine implementation II

- B is straightforward: $B \stackrel{\text{def}}{=} \overline{\texttt{out}}(0).B$
- C is more involved

$$
\begin{aligned}
C &\stackrel{\text{def}}{=} \texttt{in}(x).C'(x) \\
C'(x) &\stackrel{\text{def}}{=} \overline{\texttt{down}}(x).C + \texttt{up}(y).D(x, y) \\
D(x, y) &\stackrel{\text{def}}{=} \overline{\texttt{out}}(\max(\{x, y\})).C''(\min(\{x, y\})) \\
C''(x) &\stackrel{\text{def}}{=} \textbf{if } x = 0 \textbf{ then } \overline{\texttt{out}}(0).C \textbf{ else } C'(x)
\end{aligned}
$$

Sorting machine implementation II

- B is straightforward: $B \stackrel{\text{def}}{=} \overline{\texttt{out}}(0).B$
- C is more involved

$$
\begin{array}{lcl}
C & \stackrel{\text{def}}{=} & \texttt{in}(x).C'(x) \\
C'(x) & \stackrel{\text{def}}{=} & \overline{\texttt{down}}(x).C + \texttt{up}(y).D(x,y) \\
D(x,y) & \stackrel{\text{def}}{=} & \overline{\texttt{out}}(\max(\{x,y\})).C''(\min(\{x,y\})) \\
C''(x) & \stackrel{\text{def}}{=} & \textbf{if } x = 0 \textbf{ then } \overline{\texttt{out}}(0).C \textbf{ else } C'(x)
\end{array}
$$

- Example: $\texttt{Sorter}_3$: $C \frown C \frown C \frown B$

Proof of correctness

- Base Case: $B \approx \texttt{Spec}_0$

Proof of correctness

- Base Case: $B \approx \texttt{Spec}_0$
- General Step: $\texttt{Spec}_{n+1} \approx C \frown \texttt{Spec}_n \approx$

Proof of correctness

- Base Case: $B \approx \mathtt{Spec}_0$
- General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n \approx$
- $(\mathtt{in}(x_1).C'(x_1)) \frown (\mathtt{in}(z_1) \ldots \mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1, \ldots, z_n\})) \approx$

Proof of correctness

- Base Case: $B \approx \mathtt{Spec}_0$
- General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n \approx$
- $(\mathtt{in}(x_1).C'(x_1)) \frown (\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).(\overline{\mathtt{down}}(x_1).C + \ldots) \frown$
  $(\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$

Proof of correctness

- Base Case: $B \approx \mathtt{Spec}_0$
- General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n \approx$
- $(\mathtt{in}(x_1).C'(x_1)) \frown (\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).(\overline{\mathtt{down}}(x_1).C + \ldots) \frown$
  $(\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).\tau.(C \frown (\mathtt{in}(z_2)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{x_1,z_2,\ldots,z_n\})))$

  $\approx \quad \vdots$

Proof of correctness

- Base Case: $B \approx \mathtt{Spec}_0$
- General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n \approx$
- $(\mathtt{in}(x_1).C'(x_1)) \frown (\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).(\overline{\mathtt{down}}(x_1).C + \ldots) \frown$
  $(\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).\tau.(C \frown (\mathtt{in}(z_2)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{x_1,z_2,\ldots,z_n\})))$

  $\approx \quad \vdots$
- $\mathtt{in}(x_1)\ldots\mathtt{in}(x_n).\mathtt{in}(x_{n+1}).(C'(x_{n+1}) \frown \mathtt{Hold}_n(\{x_1,\ldots,x_n\}))$

Proof of correctness

- Base Case: $B \approx \texttt{Spec}_0$
- General Step: $\texttt{Spec}_{n+1} \approx C \frown \texttt{Spec}_n \approx$
- $(\texttt{in}(x_1).C'(x_1)) \frown (\texttt{in}(z_1)\ldots\texttt{in}(z_n).\texttt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\texttt{in}(x_1).(\overline{\texttt{down}}(x_1).C + \ldots) \frown$
  $(\texttt{in}(z_1)\ldots\texttt{in}(z_n).\texttt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\texttt{in}(x_1).\tau.(C \frown (\texttt{in}(z_2)\ldots\texttt{in}(z_n).\texttt{Hold}_n(\{x_1,z_2,\ldots,z_n\})))$

  $\approx \quad \vdots$
- $\texttt{in}(x_1)\ldots\texttt{in}(x_n).\texttt{in}(x_{n+1}).(C'(x_{n+1}) \frown \texttt{Hold}_n(\{x_1,\ldots,x_n\}))$
- Result follows using following lemma where if S is any multiset of size k and $\{x\} \cup S = \{y_1,\ldots,y_{k+1}\}$ and $y_1 \geq \ldots \geq y_{k+1}$ then

Proof of correctness

- Base Case: $B \approx \mathtt{Spec}_0$
- General Step: $\mathtt{Spec}_{n+1} \approx C \frown \mathtt{Spec}_n \approx$
- $(\mathtt{in}(x_1).C'(x_1)) \frown (\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).(\overline{\mathtt{down}}(x_1).C + \ldots) \frown$ $(\mathtt{in}(z_1)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{z_1,\ldots,z_n\})) \approx$
- $\mathtt{in}(x_1).\tau.(C \frown (\mathtt{in}(z_2)\ldots\mathtt{in}(z_n).\mathtt{Hold}_n(\{x_1,z_2,\ldots,z_n\})))$

  $\approx \quad \vdots$
- $\mathtt{in}(x_1)\ldots\mathtt{in}(x_n).\mathtt{in}(x_{n+1}).(C'(x_{n+1}) \frown \mathtt{Hold}_n(\{x_1,\ldots,x_n\}))$
- Result follows using following lemma where if S is any multiset of size k and $\{x\} \cup S = \{y_1,\ldots,y_{k+1}\}$ and $y_1 \geq \ldots \geq y_{k+1}$ then
- $C'(x) \frown \mathtt{Hold}_n(S) \approx$
  $\tau.\overline{\mathtt{out}}(y_1)\ldots\overline{\mathtt{out}}(y_{k+1}).\overline{\mathtt{out}}(0).(C \frown \mathtt{Spec}_n)$