RECITATION 14: ORDERED PROGRAMMING

RYAN KAVANAGH

Please not that the final exam will be on Tuesday, December 12, 5:30pm–8:30pm, in BH A51. This exam is closed book, closed notes. Tomorrow's lecture will be dedicated to reviewing material for the final exam. There will also be office hours at their regularly slated times this Friday, Saturday, and Monday. Please make use of these resources!

We also posted links to course FCEs and the TA evaluation forms on piazza. Please fill these out: it is the only feedback we get.

Homework 10 was due yesterday, but if you still have grace days leftover, you have until 13:30 tomorrow to hand it in. We will do our best to finish grading it by Friday so that you can get review our feedback before the final. We will make a piazza post once it is graded, and you will be able to collect it during our office hours.

1. Ordered Logic Term Assignment Rules

We review the rules assigning terms to ordered logic. In subsingleton logic, we had that contexts were generated by $\omega ::= \cdot \mid A$ and the judgment $\omega \vdash P : B$. We generalize this to the judgment

$$\Omega \vdash P :: (c : A)$$

where Ω is an ordered context generated by the grammar $\Omega ::= \cdot \mid c : A \mid \Omega_1 \cdot \Omega_2$. We read the judgment

$$(d_1 : A_1) \cdots (d_n : A_n) \vdash P :: (c : A)$$

as meaning the process P provides a service of type A along channel c, and uses channels d_i of type A_i. To emphasise that $d_1, \ldots, d_n$ and c are bound in P, we will instead notate process declarations as:

$$c \leftarrow P \leftarrow d_1 \ldots d_n = P(c, d_1, \ldots, d_n).$$

The rule driving computation is

$$\frac{\Omega \vdash P_x :: (x : A) \quad \Omega_L\ (x : A)\ \Omega_R \vdash Q_x :: (z : C)}{\Omega_L\ \Omega\ \Omega_R \vdash (x \leftarrow P_x\ ;\ Q_x) :: (z : C)}\ \mathsf{cut}$$

where we use subscripts to denote which variables are bound in the processes. The other rules are available on the last page of this document, excerpted from the 15-816 Lecture 8 notes from fall 2016. We remark that we have not seen and will not use the rules for the " $\circ$ " twist connective.

Date: 6 December 2017.

Based in part on Frank Pfenning's 15-816 lecture notes from fall 2016.

2. Examples

We will spend the remainder of the recitation exploring examples involving lists and list segments. First, we recall the definition of list_A from Lecture 24:

$$\mathsf{list}_A = \oplus\{\mathsf{cons} : A \bullet \mathsf{list}_A, \mathsf{nil} : \mathbf{1}\}.$$

A list segment is a list prefix, that is to say, a process of the type

$$\mathsf{seg}_A = \mathsf{list}_A / \mathsf{list}_A$$

expecting to receive a tail from the right, and which then provides the concatenation of the prefix with the tail. We will implement several natural list segments.

The first is the *empty* list segment which, given a list on its right, simply produces that list. This leads us to believe that it should be typed $\cdot \vdash \mathit{empty} :: (s : \mathsf{seg}_A)$. Its implementation is then:

$$\begin{array}{lll} s \leftarrow \mathit{empty} = & & \% \cdot \vdash s : \mathsf{list}_A / \mathsf{list}_A \\ \quad t \leftarrow \mathsf{recv}\ s; & & \% \ t : \mathsf{list}_A \vdash s : \mathsf{list}_A \\ \quad s \leftarrow t & & \end{array}$$

In the second line, we receive the tail $t : \mathsf{list}_A$ oven the channel s, and in the third line, we forward from t to s. The comments following the "%" sign describe the type of the term we must provide on the remaining lines. For example, on the first line, it denotes that we must provide a body $\mathit{empty}(s)$ such that $\cdot \vdash \mathit{empty}(s) :: (s : \mathsf{seg}_A)$, and on the second line that we must provide a term T such that $t : \mathsf{list}_A \vdash T :: (s : \mathsf{list}_A)$.

Another thing we can do is *concatenate* segments. Its type should be:

$$(s_1 : \mathsf{seg}_A)(s_2 : \mathsf{seg}_A) \vdash \mathit{concat} :: (s : \mathsf{seg}_A).$$

We implement it as follows:

$$\begin{array}{lll} s \leftarrow \mathit{concat} \leftarrow s_1\ s_2 = & & \\ \quad t \leftarrow \mathsf{recv}\ s; & & \% \ (s_1 : \mathsf{seg}_A)(s_2 : \mathsf{seg}_A)(t : \mathsf{list}_A) \vdash s : \mathsf{list}_A \\ \quad \mathsf{send}\ s_1\ t; & & \% \ (s_1 : \mathsf{seg}_A)(s_2 : \mathsf{list}_A) \vdash s : \mathsf{list}_A \\ \quad \mathsf{send}\ s_1\ s_2; & & \% \ s_1 : \mathsf{list}_A \vdash s : \mathsf{list}_A \\ \quad s \leftarrow s_1 & & \end{array}$$

We may wish to convert segments to lists by simply providing them with nil as a tail. The corresponding type ought to be: $s : \mathsf{seg}_A \vdash \mathit{toList} :: (l : \mathsf{list}_A)$. The implementation is:

$$\begin{array}{lll} l \leftarrow \mathit{toList} \leftarrow s = & & \\ \quad n \leftarrow \mathit{nil}; & & \%(s : \mathsf{seg}_A)(n : \mathsf{list}_A) \vdash l : \mathsf{list}_A \\ \quad \mathsf{send}\ s\ n; & & \% \ s : \mathsf{list}_A \vdash l : \mathsf{list}_A \\ \quad l \leftarrow s & & \end{array}$$

(Recall from lecture: $\cdot \vdash \mathit{nil} :: (l : \mathsf{list}_A)$ given by $l \leftarrow \mathit{nil} = l.\mathsf{nil}; \mathsf{close}\ l$.)

Conversely, we may wish to convert a list to a list segment. The corresponding type is $l : \mathsf{list}_A \vdash \mathit{toSeg} :: (s : \mathsf{seg}_A)$. The implementation is:

$$\begin{array}{lll} s \leftarrow \mathit{toSeq} \leftarrow l = & & \\ \quad t \leftarrow \mathsf{recv}\ l & & \% \ (l : \mathsf{list}_A)(t : \mathsf{list}_A) \vdash s : \mathsf{list}_A \\ \quad \mathit{concat} & & \end{array}$$

where we make use of the helper function $(\mathsf{q} : \mathsf{list}_A)(\mathsf{r} : \mathsf{list}_A) \vdash \mathit{concat} :: (\mathsf{s} : \mathsf{list}_A)$ that concatenates two lists:

$$
\begin{array}{l}
\mathsf{s} \leftarrow \mathit{concat} \leftarrow \mathsf{q}\ \mathsf{r} = \\
\quad \mathsf{case}\ \mathsf{q}\ (\mathsf{nil} \Rightarrow \mathsf{wait}\ \mathsf{q}; \mathsf{s} \leftarrow \mathsf{r} \\
\qquad\quad \mid \mathsf{cons} \Rightarrow \mathsf{h} \leftarrow \mathsf{recv}\ \mathsf{q}; \mathsf{s.cons}; \mathsf{send}\ \mathsf{s}\ \mathsf{h}; \mathit{concat})
\end{array}
$$

Judgmental Rules

$$\frac{\Omega \vdash P_x :: (x{:}A) \quad \Omega_L\ (x{:}A)\ \Omega_R \vdash Q_x :: (z{:}C)}{\Omega_L\ \Omega\ \Omega_R \vdash (x \leftarrow P_x\ ;\ Q_x) :: (z{:}C)}\ \mathsf{cut} \qquad \frac{}{y{:}A \vdash x \leftarrow y :: (x{:}A)}\ \mathsf{id}$$

Additive Connectives

$$\frac{\Omega \vdash P :: (x{:}A_k) \quad (k \in I)}{\Omega \vdash (x.l_k\ ;\ P) :: (x : \oplus\{l_i{:}A_i\}_{i \in I})}\ \oplus R_k \qquad \frac{\Omega_L\ (x{:}A_i)\ \Omega_R \vdash Q_i :: (z{:}C) \quad (\forall i \in I)}{\Omega_L\ (x{:}\oplus\{l_i{:}A_i\}_{i \in I})\ \Omega_R \vdash \mathsf{case}\ x\ (l_i \Rightarrow Q_i)_{i \in I} :: (z{:}C)}\ \oplus L$$

$$\frac{\Omega \vdash P_i :: (x{:}A_i) \quad (\forall i \in I)}{\Omega \vdash \mathsf{case}\ x\ (l_i \Rightarrow P_i)_{i \in I} :: (x{:}\&\{l_i{:}A_i\}_{i \in I}))}\ \&R \qquad \frac{\Omega_L\ (x{:}A_k)\ \Omega_R \vdash P :: (z{:}C) \quad (k \in I)}{\Omega_L\ (x : \&\{l_i{:}A_i\}_{i \in I})\ \Omega_R \vdash (x.l_k\ ;\ Q) :: (z{:}C)}\ \&L_k$$

Multiplicative Connectives

$$\frac{}{\cdot \vdash \mathsf{close}\ x :: (x{:}\mathbf{1})}\ \mathbf{1}R \qquad \frac{\Omega_L\ \Omega_R \vdash Q :: (z{:}C)}{\Omega_L\ (x{:}\mathbf{1})\ \Omega_R \vdash (\mathsf{wait}\ x\ ;\ Q) :: (z{:}C)}\ \mathbf{1}L$$

$$\frac{\Omega\ (y{:}B) \vdash P_y :: (x{:}A)}{\Omega \vdash (y \leftarrow \mathsf{recv}\ x\ ;\ P_y) :: (x{:}A\ /\ B)}\ /R \qquad \frac{\Omega_L\ (x{:}A)\ \Omega_R \vdash Q :: (z{:}C)}{\Omega_L\ (x{:}A\ /\ B)\ (w{:}B)\ \Omega_R \vdash (\mathsf{send}\ x\ w\ ;\ Q) :: (z{:}C)}\ /L^*$$

$$\frac{(y{:}B)\ \Omega \vdash P_y :: (x{:}A)}{\Omega \vdash (y \leftarrow \mathsf{recv}\ x\ ;\ P_y) :: (x{:}B \setminus A)}\ \setminus R \qquad \frac{\Omega_L\ (x{:}A)\ \Omega_R \vdash Q :: (z{:}C)}{\Omega_L\ (w{:}B)\ (x{:}B \setminus A)\ \Omega_R \vdash (\mathsf{send}\ x\ w\ ;\ Q) :: (z{:}C)}\ \setminus L^*$$

$$\frac{\Omega \vdash P :: (x{:}B)}{(w{:}A)\ \Omega \vdash (\mathsf{send}\ x\ w\ ;\ P) :: (x{:}A \bullet B)}\ \bullet R^* \qquad \frac{\Omega_L\ (y{:}A)\ (x{:}B)\ \Omega_R \vdash Q_y :: (z{:}C)}{\Omega_L\ (x{:}A \bullet B)\ \Omega_R \vdash (y \leftarrow \mathsf{recv}\ x\ ;\ Q_y) :: (z{:}C)}\ \bullet L$$

$$\frac{\Omega \vdash P :: (x{:}B)}{\Omega\ (w{:}A) \vdash (\mathsf{send}\ x\ w\ ;\ P) :: (x{:}A \circ B)}\ \circ R^* \qquad \frac{\Omega_L\ (x{:}B)\ (y{:}A)\ \Omega_R \vdash Q_y :: (z{:}C)}{\Omega_L\ (x{:}A \bullet B)\ \Omega_R \vdash (y \leftarrow \mathsf{recv}\ x\ ;\ Q_y) :: (z{:}C)}\ \circ L$$