Concurrency	Exam (2 hours)	December 17, 2003

Lambda Calculus

Recall the definitions of some classical λ-terms:

$$\begin{array}{lcll} I & = & \lambda x.x & \\ K & = & \lambda x.\lambda y.x & \\ S & = & \lambda x.\lambda y.\lambda z.xz(yz) & \\ \Omega & = & (\lambda x.x\,x)\,\lambda x.x\,x & \\ Y & = & \lambda f.B\,B & \text{where } B = \lambda b.f\,(b\,b) \end{array}$$

1. Comparing λ-terms

We consider the following equivalence relations between λ-terms:

$=_\beta$	β-convertibility
Λ^2	all pairs of λ-terms
$=_\alpha$	α-equivalence
$\cong$	operational equivalence (w.r.t. call-by-name strategy)

We make precise here the definition of the operational equivalence:

$$\begin{array}{lll} M \cong N & \text{iff} & \forall C.\ C[M] \Downarrow \Leftrightarrow C[N] \Downarrow \\ M \Downarrow & \text{iff} & \exists V.\ M \to^*_{\text{cbn}} V \end{array}$$

1. Order the equivalence relations given above from smallest to largest with respect to set inclusion.

2. Now, for each pair of λ-terms in the table below, choose among these equivalence relations the smallest one that contains this pair and fill the third column.

M	N	$(M,N) \in$
x	x	
x	y	
I	SKK	
YI	Ω	
$\lambda x.\lambda x.\lambda y.x$	$\lambda x.\lambda y.\lambda z.y$	
$\lambda x.\lambda x.\lambda y.x$	$\lambda x.\lambda y.\lambda z.x$	
$Y(\lambda f.\lambda x.Kx(fx))y$	Ω	
$\lambda x.fx$	f	
Y	Ω	

2. Normal forms

Let $\mathcal{V}$ be the set of variables and Λ the set of λ-terms.
Let $\mathcal{N} \subset \Lambda$ be the set of β-normal forms ($\mathcal{N} = \{M \in \Lambda \mid \nexists M' \in \Lambda : M \to_\beta M'\}$).
We define inductively the subset $\mathcal{N}'$ of Λ:

$$(\textsc{Var})\ \frac{x \in \mathcal{V}}{x \in \mathcal{N}'} \qquad (\textsc{Abs})\ \frac{x \in \mathcal{V} \quad M \in \mathcal{N}'}{\lambda x.M \in \mathcal{N}'} \qquad (\textsc{App}_n)\ \frac{x \in \mathcal{V} \quad M_1 \in \mathcal{N}' \quad \cdots \quad M_n \in \mathcal{N}'}{xM_1 \cdots M_n \in \mathcal{N}'}\ n \in \mathbb{N}, n > 0$$

Show that $\mathcal{N}' = \mathcal{N}$. (Hint: show that if $M \in \mathcal{N}$ then $M \in \mathcal{N}'$ by induction on $M \in \Lambda$ and that if $M \in \mathcal{N}'$ then $M \in \mathcal{N}$ by induction on the derivation $M \in \mathcal{N}'$).

Pi Calculus

Recall the π-calculus process syntax:

$$\begin{array}{rcl} P & ::= & M \,\big|\, P \,|\, P' \,\big|\, (\nu z)\, P \,\big|\, !\, P \\ M & ::= & \mathbf{0} \,\big|\, a(\tilde{x}).P \,\big|\, \overline{a}\langle\tilde{b}\rangle.P \,\big|\, \tau.P \,\big|\, M + M' \end{array}$$

3. Transition and transition graph

1. Give the derivation tree of an internal transition of $(\nu c)\, !\, (c(c).\overline{c}\langle z\rangle.\, \mathbf{0} + (\nu z)\, \overline{c}\langle z\rangle.z(z).\overline{z}.\, \mathbf{0})$.

2. Give the transition graph of $\mathrm{W}(l)$ where $\mathrm{W}(l) := (\nu t, f)\, \overline{l}\langle t, f\rangle.(t.\mathrm{W}(l) + f.\, \mathbf{0})$.

4. Church encoding

During the course, you have seen the Church encoding of booleans in the π-calculus and some more encodings in the λ-calculus. In this exercise, we will see an encoding of integers in the π-calculus. However, instead of using the standard unary encoding, we will see a more efficient encoding.
In the π-calculus encodings, the data we encode are located at some location l (which is where we can access the data). We give below the encoding $[\![n]\!]_l$ of an integer $n \in \mathbb{N}$ located at l:

$$\begin{array}{rcll} [\![n]\!]_l & := & \mathrm{Z}(l) & \text{if } n = 0 \\ [\![n]\!]_l & := & (\nu k)\, (\mathrm{SD}(l,k) \,|\, [\![p]\!]_k) & \text{if } n = 2 * p + 1 \\ [\![n]\!]_l & := & (\nu k)\, (\mathrm{D}(l,k) \,|\, [\![p]\!]_k) & \text{if } n = 2 * p \text{ and } p \neq 0 \end{array}$$

The three processes $\mathrm{Z}(l)$, $\mathrm{SD}(l,k)$ and $\mathrm{D}(l,k)$ performs more or less the same thing: they wait for three names z, sd and d on the channel l and reply on one of these three channels to indicate the kind of the integer (z for null, sd for odd and d for even but not null) and where the next integer can be found (at location k). For example, here is the definition of $\mathrm{Z}(l)$and $\mathrm{SD}(l,k)$:

$$\begin{array}{rcl} \mathrm{Z}(l) & := & l(z, sd, d).\overline{z}.\, \mathbf{0} \\ \mathrm{SD}(l,k) & := & l(z, sd, d).\overline{sd}\langle k\rangle.\, \mathbf{0} \end{array}$$

Fill the definition of $\mathrm{D}(l,k)$:

$$\mathrm{D}(l,k) \quad := \quad l(z, sd, d).\boxed{?}$$

To ease the writing of processes, we define three well known integers:

$$\begin{array}{rcl} \mathrm{Zero}(l) & := & \mathrm{Z}(l) \\ \mathrm{One}(l) & := & (\nu k)\, (\mathrm{SD}(l,k) \,|\, \mathrm{Zero}(k)) \\ \mathrm{Two}(l) & := & (\nu k)\, (\mathrm{D}(l,k) \,|\, \mathrm{One}(k)) \end{array}$$

You may have noticed that our integers are linear in the meaning that once they are used, they vanish. Write the definition of a process $\mathrm{Copy}(l,k)$ such that $\mathrm{Copy}(l,k) \,|\, [\![n]\!]_l$ reduces[1] to[2] $[\![n]\!]_l \,|\, [\![n]\!]_k$ for all $n \in \mathbb{N}$.

$$\mathrm{Copy}(l,k) \quad := \quad \boxed{?}$$

To see how we can compute with these integers, write now the definition of a process $\mathrm{Succ}(l,k)$ such that $\mathrm{Succ}(l,k) \,|\, [\![n]\!]_l$ reduces to $[\![n+1]\!]_k$ for all $n \in \mathbb{N}$.

$$\mathrm{Succ}(l,k) \quad := \quad \boxed{?}$$

[1] with some internal computation
[2] something equivalent to