Exercise 6

1. Recall the variable agent example on lecture notes. If the read process is changed into

$$\mathbf{Reader}(a,b) \stackrel{\text{df}}{=} (\nu\, s)\overline{a}\langle s\rangle.s \lhd \mathsf{read}.s(y).\overline{b}\langle y\rangle$$

   (a) Write the reduction of $\mathbf{Var}\langle a,5\rangle \mid \mathbf{Read}\langle a,b\rangle$.

   (b) Write all the possible reductions of $\mathbf{Var}\langle a,5\rangle \mid \mathbf{Read}\langle a,b\rangle \mid \mathbf{Write}\langle a,6\rangle$.

2. Write a small variable agent $\mathbf{Var}(a, v_1, v_2, ...v_n)$ (storing value $v_1, ..., v_n$ at a), which satisfies the following specification:

   - If read_i is chosen, then it returns the stored value v_i and recurs to the same variable;
   - If write_i is chosen, then it receives a value w and returns to the variable with the new state $\mathbf{Var}(a, v_1, ..., v_{i-1}, w, v_{i+1}, ..., v_n)$;
   - If quit is chosen, then it ends the loop (i.e. it terminates with 0).

   Also write the expressions for reading, writing and quit process, denoted as $\mathbf{Reader}_i(a)$, $\mathbf{Writer}_i(a,w)$, and $\mathbf{Quit}(a)$ respectively.

3. An ATM agent offers three services: balance, deposit or exit. If *balance* is chosen, then it shows a balance of the account, and recurs to the menu with the same amount. If *deposit* is chosen, then it receives a deposited amount z, and returns to the menu with the new state as their sum $y+z$. If *exit* is chosen, it exits the service. The following is the implementation of the ATM.

$$\begin{array}{lll}
\mathbf{ATM}(a,y) \stackrel{\text{df}}{=} !a(z). \quad \mathbf{ATM_1}\langle y,z\rangle & & \\
\mathbf{ATM_1}(y,s) \stackrel{\text{df}}{=} \quad s \rhd [& \mathsf{balance}: & \overline{s}\langle y\rangle.\mathbf{ATM_1}\langle y,s\rangle \;[\!]\; \\
& \mathsf{deposit}: & s(z).\overline{s}\langle y+z\rangle.\mathbf{ATM_1}\langle y+z,s\rangle \\
& \mathsf{exit}: & \mathbf{0}\ ]
\end{array}$$

   (a) Explain the role of s in the definition of the ATM agent.

   (b) Define a Customer agent: $Customer(a,z)$ that deposit z pounds and exits.

   (c) Check that the following reduces correctly (without inferferences):

$$\mathbf{ATM}\langle a,10000\rangle \mid \mathbf{Customer}\langle a,100\rangle \mid \mathbf{Customer}\langle a,1000\rangle$$

4. The following expressions show the bahaviour of an FTP server.

$$\begin{array}{l}
\mathbf{Client}(pid) \stackrel{\text{df}}{=} (\nu\, s)\overline{pid}\langle s\rangle.\overline{s}\langle userid, passwd\rangle.s \rhd \{\mathsf{sorry}: ... \;[\!]\; \mathsf{welcome}: ...\} \\
\mathbf{Init}(pid, nis) \stackrel{\text{df}}{=} \quad (\nu\, b)(\mathbf{Ftpd}\langle pid, b\rangle \mid \mathbf{FtpThread}\langle b, nis\rangle \mid \mathbf{Authenticator}\langle nis\rangle)
\end{array}$$

$\mathbf{Ftpd}(pid, b) \stackrel{\text{df}}{=} pid(z).(\nu\, s')\overline{b}\langle s'\rangle.\overline{s'}\langle z\rangle.\mathbf{Ftpd}\langle pid, b\rangle$

$\mathbf{FtpThread}(b, nis) \stackrel{\text{df}}{=} b(s').s'(z).z(userid, passwd).$
$(\nu\, s'')\overline{nis}\langle s''\rangle.s'' \lhd \mathsf{checkUser}.\overline{s''}\langle userid, passwd\rangle.$
$s'' \rhd \{\mathsf{invalid} : z \lhd \mathsf{sorry} \,[\!]\, \mathsf{valid} : z \lhd \mathsf{welcome}.\mathbf{Action}\langle z, b\rangle\}$

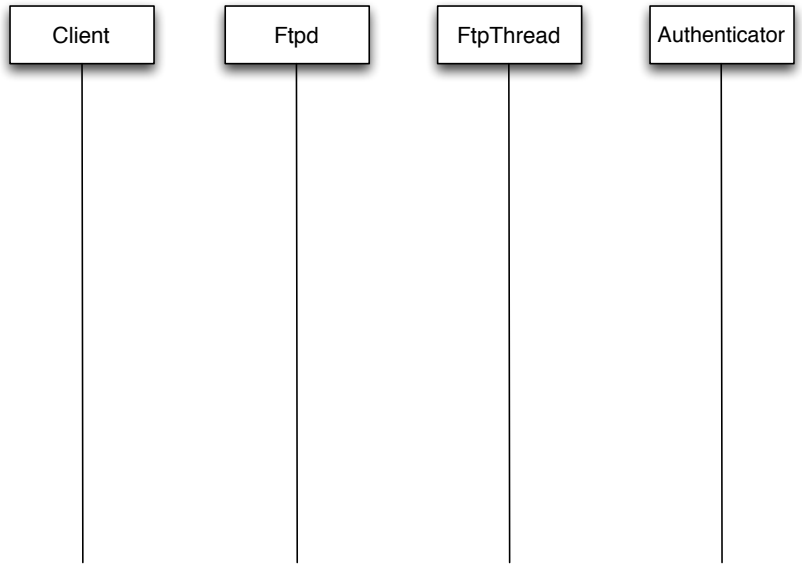
$\mathbf{Actions}(s, b) \stackrel{\text{df}}{=} s \rhd \{ \mathsf{get} : \mathbf{Action}\langle s, b\rangle \,[\!]$
$\mathsf{put} : \mathbf{Actions}\langle s, b\rangle \,[\!]$
$\mathsf{bye} : \mathbf{FtpThread}\langle b, nis\rangle\}$

$\mathbf{Authenticator}(nis) \stackrel{\text{df}}{=} nis(x).x \rhd \{ \mathsf{checkUser} : x(userid, passwd).$
$\textbf{if } passwd = pw \textbf{ then } x \lhd \mathsf{valid}.\mathbf{Authenticator}\langle nis\rangle$
$\textbf{else } x \lhd \mathsf{invalid}.\mathbf{Authenticator}\langle nis\rangle \,[\!]...\}$

where we assume that pw is the valid password of the user with $userid$.

(a) Briefly explain how the system works and illustrate the process in the message passing diagram. The template is shown below. We assume that there is an authenticator for validating users on the FTP server.

(b) Comment the usage of the delegation in this example.



5. (Optional) In Domain Name System (DNS), there are two ways of querying the IP address for a specific domain name: iterative queries and recursive queries. (1) In iterative queries, name servers return the best information they have. Although a DNS server may not know the IP address for a given friendly name, it might know the IP

address of another name server likely to have the IP address being sought, so it sends that information back. (2) When a client system sends a recursive query to a local name server, that local name server must return the IP address for the domain name entered. If it does not know the IP address, it will recursively ask other DNS servers until the IP address is found and returned. The process is shown in Figure 1 (a) for iterative query and in Figure 1 (b) for recursive query.

(a) Write the expression for Client(a, s) that wants to query the IP address for a specific domain name.

(b) Write the expressions for LocalDNS(a) and OtherDNS(b) in iterative queries.

(c) Write the expressions for LocalDNS'(a) and OtherDNS'(b) in recursive queries.

6. Session types:

(a) What is the dual of the following session types?

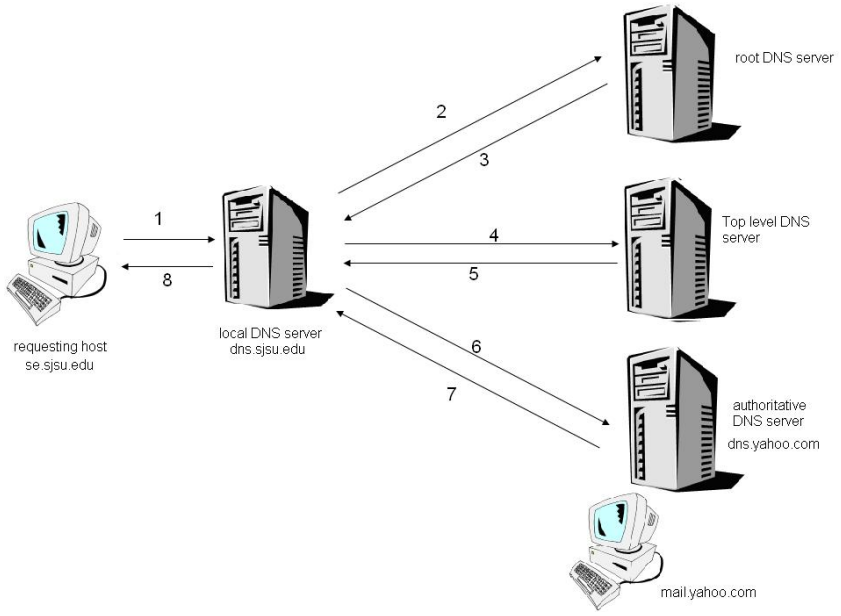
i. $![\texttt{nat}]; \texttt{end}$

ii. $\mu t.(?[\texttt{string}]; ?[\texttt{string}]; ![\texttt{boolean}]; t)$

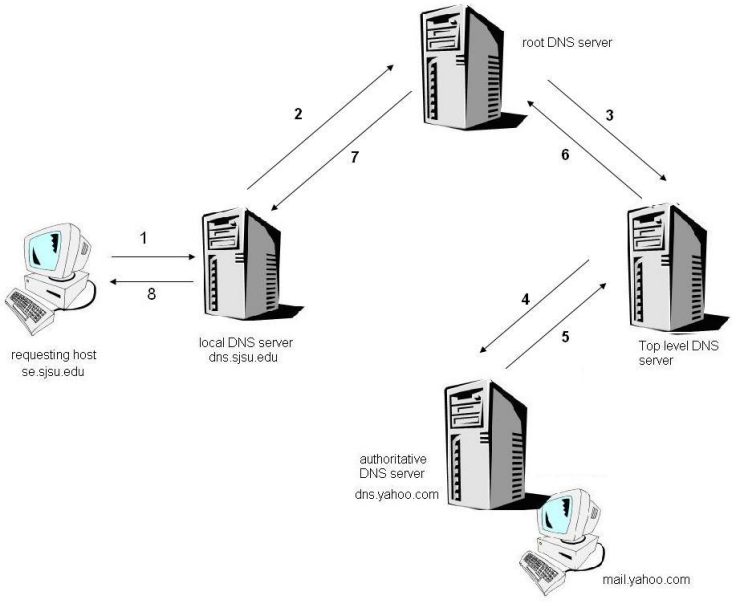
iii. $\mu t.(![\texttt{nat}]; ?[\texttt{string}]; \oplus\{\texttt{result} :![\texttt{double}]; t, \texttt{retry} :?[\texttt{string}]; t, \texttt{quit} : \texttt{end}\})$

iv. $\oplus\{\texttt{apple} :![\texttt{nat}]; ?[\texttt{boolean}]; \texttt{end}, \ \texttt{orange} : \&\{ \ \texttt{pear} :?[\texttt{string}]; \texttt{end}, \ \texttt{banana} : ![\texttt{nat}]; \texttt{end}\}\}$

(b) Give the session types of the ATM and the client in Question 3. You need to use the subtyping for typing the client.



(a) Iterative Queries



(b) Recursive Queries

Figure 1: Two ways of name resolution