Concurrent Object-Oriented Programming

Exercise session 7:
CCS and its semantics.

1. Syntax and Semantics of CCS:
The Calculus of Concurrent Systems (CCS) was proposed by Robin Milner in the early 80’s, as a model for concurrent, communicating processes. It distinguishes between resources, which are passive and represented in the calculus by names belonging to a given set $\mathbb{N}$, and processes, which are activities manipulating (accessing and creating) those resources.
The syntax of the calculus here follows:

$$P ::= 0 \mid a.P \mid \overline{a}.P \mid (\nu a)P$$
$$\mid P + P \mid P|P \mid M$$

The processes are called, from left to right, on the first line: the idle process, the input prefix, the output prefix, and the restriction; and on the second line the choice, the parallel, and the named process invocation. The input and output prefixes are used to model communication. The idle process models inactivity, the restriction models the privacy of a resource to a process, the choice models non-determinism, while the parallel operator models parallelism of activities. The named process invocation is useful to define recursive behaviors. It assumes the existence of a finite set process definitions, which we will suppose well-formed.

As a convention, restriction extends as far right as possible, and the choice operator binds more than the parallel operator. We will omit trailing idle processes on a general basis.

The semantics of any CCS term is given by a Labeled Transition System (LTS), defined using the Structural Operational Semantics (SOS) framework. The corresponding rules are given below. We will note:

$$\alpha \in \{a,\overline{a} \mid a \in \mathbb{N}\}$$

And:

$$\overline{\alpha} \triangleq \overline{\alpha} \text{ when } \alpha = a, \text{ and } \overline{\alpha} \triangleq a \text{ when } \alpha = a.$$
The prefix is the only axiom in the system.

\[ \alpha.p \xrightarrow{\alpha} p \]

The choice requires two symmetric rules.

\[ \begin{align*}
\frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'} \\
\frac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}
\end{align*} \]

So does parallel composition.

\[ \begin{align*}
\frac{p \xrightarrow{\alpha} p'}{p | q \xrightarrow{\alpha} p' | q} \\
\frac{q \xrightarrow{\alpha} q'}{p | q \xrightarrow{\alpha} p | q'}
\end{align*} \]

Communication (a.k.a. interaction) may happen when compatible prefixes are allowed.

\[ \begin{align*}
\frac{p \xrightarrow{\alpha} p'}{p | q \xrightarrow{\tau} p' | q'} \\
\frac{q \xrightarrow{\alpha} q'}{p | q \xrightarrow{\alpha} p | q'}
\end{align*} \]

Restriction does not prevent non-restricted (free) names to appear, but the bound name(s).

\[ \begin{align*}
\frac{p \xrightarrow{\alpha} p'}{(\nu b)p \xrightarrow{\alpha} (\nu b)p' | q}
\end{align*} \]

\[ a \text{ syntactically different from } b \]

Name invocation just replaces a named process by its definition.

\[ \begin{align*}
\frac{p \xrightarrow{\alpha} p'}{M \xrightarrow{\alpha} p'} (M \triangleq p)
\end{align*} \]

2. Syntax:

Give the semantics of the following terms:

\[ \begin{align*}
a.b.c.0, a.c.0 + b.c.0, a.b + a.c, a.(b + c), a \mid b, a.b + a.c \mid \bar{a}.\bar{b} \\
(\nu a)((a + b) \mid \bar{a}), P \triangleq a.((\tau.P + b) + \tau.a.P), Q \triangleq \tau.(\nu a)(a \mid \bar{a} + b) + c.Q, R \triangleq a.R
\end{align*} \]

3. Modeling example:

Write the process terms for the following Network, Sender and Receiver connected as described in the diagram below. The network exhibits no failures. The sender waits for something to send, then emits it and waits for the acknowledgement. Conversely does the receiver.
Now describe a lossy network. Implement the *alternating bit protocol*: where each message and acknowledgement bears a tag, either 0 or 1, that alternates with each new message sending. If the sender receives the wrong number with the acknowledgement, then this acknowledgement is ignored. Otherwise, the sender can proceed and send a new message (with the bit duly alternated, of course).