1 Concurrency

The semantics of the langauges we have seen so far in this course are sequential, and often even deterministic. That is, each program only has a single thread of execution, and if we model the semantics operationally, the evaluation relation can be modeled using a function.

However, many systems today are concurrent: they have multiple threads of execution that interact with each other, either through message passing or by manipulating shared state.

The next few lectures will introduce core languages with concurrency expressed using message passing. A key feature of these languages, often called *process calculi*, are that equivalence is characterized in terms of *bisimulation*. We will start with CCS, a simple process calculus introduced by Milner around 1980.

2 CCS

Let $\mathcal{L}$ be a set of labels $\lambda, \mu, \ldots$. An action α is either a label λ, its complement $\overline{\lambda}$, or τ. Intuitively λ and $\overline{\lambda}$ will denote communication on λ while τ will denote internal computation within a process.

The syntax of CCS processes is defined as follows:

$P ::=$	$A\langle a_1, \ldots, a_n\rangle$	Instantiation
$\mid$	$\sum_{i \in I} \alpha_i.P_i$	Summation
$\mid$	$P_1 \mid P_2$	Parallel Composition
$\mid$	$\mathsf{new}\ a\ P$	Restriction

A program is a sequence of top-level definition $A(a_1, \ldots, a_n) = P_A$ and a "main" process P.

Note that in $\mathsf{new}\ a\ P$ is a binder for a. As usual, we work up to α-equivalence and assume standard capture-avoiding substitution.

3 Reduction

A semantics for CCS can be formulated in terms of *reductions*:

$$\frac{}{(a.P + M) \mid (\overline{a}.Q + N) \to P \mid Q}\ \text{React} \qquad \frac{P \to P'}{P \mid Q \to P' \mid Q}\ \text{Par} \qquad \frac{P \to P'}{\mathsf{new}\ a\ P \to \mathsf{new}\ a\ P'}\ \text{Res}$$

$$\frac{}{\tau.P + M \to P}\ \text{Tau} \qquad \frac{Q \equiv P \quad P \to P' \quad P' \equiv Q'}{Q \to Q'}\ \text{Struct}$$

These rules make use of an equivalence known as *structural congruence*, which is defined as follows:

1. α-conversion

2. Re-ordering of sums
3. $P|0 \equiv P$, $P|Q \equiv Q|P$, $P|(Q|R) \equiv (P|Q)|R$
4. $\mathsf{new}\ a\ (P|Q) \equiv (\mathsf{new}\ a\ Q)|P$ if $a \notin \mathsf{fv}(P)$, $\mathsf{new}\ a\ 0 \equiv 0$, $\mathsf{new}\ a, b\ P \equiv \mathsf{new}\ b, a\ P$
5. $A\langle\vec{b}\rangle \equiv \{\vec{b}/\vec{a}\}P_A$ where $A(\vec{a}) = P_A$

4 Labeled Transition System

We can also give a semantics for CCS as a *labeled transition system* or CCS.

$$\frac{}{M + \alpha.P + N \xrightarrow{\alpha} P}\ \text{L-Sum} \qquad \frac{P \xrightarrow{\lambda} P' \quad Q \xrightarrow{\overline{\lambda}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}\ \text{L-React} \qquad \frac{P \xrightarrow{\alpha} P' \quad \alpha \notin \{a, \overline{a}\}}{\mathsf{new}\ a\ P \xrightarrow{\alpha} \mathsf{new}\ a\ P'}\ \text{L-Res}$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ \text{L-Par L} \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}\ \text{L-Par R} \qquad \frac{\{\vec{b}/\vec{a}\}P_A \xrightarrow{\alpha} P' \quad A(\vec{a}) = P_A}{A\langle\vec{b}\rangle \xrightarrow{\alpha} P'}\ \text{L-Ident}$$

Note that we no longer make use of structural congruence (although we do still work up to α-equivalence). Also, in the L-React rule, since λ is internal to the process, we label the transition with τ so that λ is hidden from any external processes.

5 Properties

We can show that even though we no longer have a structural congruence rule, structural congruence in fact still holds. We therefore have the following theorem:

Theorem 1. *If $P \xrightarrow{\alpha} P'$ and $P \equiv Q$, then $\exists Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $Q' \equiv P'$.*

Proof. Here is a partial proof, containing only a few subcases. Proof by induction on $P \xrightarrow{\alpha} P'$. Case L-Par L: $P = P_1|P_2$, $P_1 \xrightarrow{\alpha} P_1'$, $P' = P_1'|P_2$. Consider $P \equiv Q$. We now look at all of the ways Q could be structurally congruent to P:

Subcase $Q = P_2|P_1$. Then let $Q' = P_2|P_1'$. By L-Par R, $Q \xrightarrow{\alpha} Q'$. ✓

Subcase $Q = Q_1|P_2$, $Q_1 \equiv P_1$. By the induction hypothesis, $\exists Q_1'$ such that $Q_1 \xrightarrow{\alpha} Q_1'$, $Q_1' \equiv P_1$. By L-Par L, $Q \xrightarrow{\alpha} Q_1'|P_2 \equiv P'$. Then let $Q' = Q_1'|P_2$. ✓ □

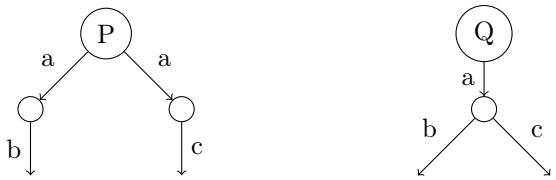
We can also show that the transitions in this system correspond to those in the original CCS:

Theorem 2. *$P \to P'$ iff $P \xrightarrow{\tau}\!\equiv P'$ (where $\xrightarrow{\tau}\!\equiv$ indicates relational composition of $\xrightarrow{\tau}$ and $\equiv$).*

6 Bisimulation

One "obvious" notion of equvialence we might be tempted to use for a process would be one that equates the traces it generates. However, this would not be satisfactory because it would equate processes such as the following:

Viewed as automata, we have $\mathcal{L}(P) = \{\text{ab, ac}\}$ and $\mathcal{L}(B) = \{\text{ab, ac}\}$. That is, both processes recognize the same language. However, intuitively, Q simulates P, but P does not simulate Q. Therefore, language equivalence is not good enough.



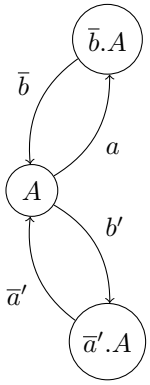
7 Example: One-place buffer

Let $P(x, x', y, y') = x.\overline{y}.P\langle x, x', y, y'\rangle + y'.x'.P\langle x, x', y, y'\rangle$

Let A be an instance of P:

$$\begin{aligned} A &= P\langle a, a', b, b'\rangle \\ &\equiv a.\overline{b}.A + b'.\overline{a}'.A \end{aligned}$$

We can draw the labeled transition system for A as follows.



Definition 3 (Simulation)**.** *A binary relation S is a* simulation *if, whenever $P\ S\ Q$, if $P \xrightarrow{\alpha} P'$ then $\exists Q'$ such that $Q \xrightarrow{\alpha} Q'$ and $P'\ S\ Q'$.*

This situation can be depicted graphically as follows:

$$\begin{array}{ccccc} & P & S & Q & \\ \alpha & \downarrow & & \downarrow & \alpha \\ P' & & \equiv S \equiv & & Q' \end{array}$$

Definition 4 (Bisimulation)**.** *A binary relation S is a* bisimulation *if both S and S^{-1} are simulations.*

Definition 5 (Bisimilarity)**.** Bisimilarity *is the union of all bisimulations.*

The proof of the following fact is left as an exercise.

Lemma 6. *Bisimilarity is an equivalence.*