An abstract machine for HOcore

Lionel Zoubritzky

August 11, 2017

Contents

1 Introduction

HOcore is a minimal higher-order process calculus inspired of HOπ. It is minimal in the sense that it remains Turing-complete [3] , although it does not have any name restriction. It also has the property that strong barbed congruence and IO bisimilarity coincide and are decidable [4].

We describe a simple abstract machine for HOcore and prove its soundness and completeness. To do this, we propose a translation between HOcore processes and machines an prove that any reduction of the process matches a transition of the machine and the converse.

We then refine the definition of the machine to create a notion of machine bisimilarity, and prove that it is equivalent modulo translation to IO bisimilarity for HOcore processes.

Overall, the machine is quite similar to the Krivine Abstract Machine, with environments used to simulate reductions, only distributed in a multiset of processes that may interact if there is a valid transmission available.

This work is organized as follow:

- Part 2 presents the semantics of HOcore
- Part 3 introduces notations for the abstract machine
- Part 4 presents the main results on well-formedness
- Part 5 introduces the translation between processes and machines
- Part 6 details various consequences of the previous definitions
- Part 7 presents the proof of the soundness of the machine
- Part 8 presents the proof of the completeness of the machine
- Part 9 introduces the notion of machine bisimilarity
- Part 10 presents the proof of the equivalence between bisimilarities

2 HOcore

Definition 1. A (HOcore) *process* is defined as

$$P ::= \ \bar{a}\left\langle P\right\rangle \,\big|\, a(x).P \,\big|\, P \parallel P \,\big|\, x \,\big|\, \star$$

where x ranges over the variables and a over the channels.

- $\bar{a}\left\langle P\right\rangle$ designates the emission of P on channel a. HOcore is higher-order in the sense that the message P is itself a process.

- $a(x).P$ designates the reception on channel a of a process that will replace x in process P.
- $P \parallel Q$ designates the parallel composition of P and Q. This operation is commutative, associative and admits $\star$ as neutral element.
- x is a variable. A process is closed if all its variables x have been bound within receptions of the form $a(x).P$.
- $\star$ designates the empty process.

Processes may also be represented using de Bruijn variables with the locally nameless convention [2]:

$$P ::= \bar{a}\left\langle P \right\rangle \bigm| a.P \bigm| P \parallel P \bigm| i \bigm| x \bigm| \star$$

where i is a natural integer which represents a bound variable and x ranges over the free variables.

From now on, we shall only use this latter representation.

Examples:

- The process $\bar{a}\left\langle \star \right\rangle \parallel a(x).x$ is noted in de Bruijn $\bar{a}\left\langle \star \right\rangle \parallel a.0$.
- The process $(b(x).(y \parallel a(y).x)) \parallel \bar{b}\left\langle c(x).\bar{a}\left\langle x \right\rangle \right\rangle$ is noted in de Bruijn $(b.(y \parallel a.1)) \parallel \bar{b}\left\langle c.\bar{a}\left\langle 0 \right\rangle \right\rangle$.
- The process $\Omega = \bar{a}\left\langle a(x).(x \parallel \bar{a}\left\langle x \right\rangle) \right\rangle \parallel (a(x).(x \parallel \bar{a}\left\langle x \right\rangle))$ is represented in de Bruijn by $\Omega = \bar{a}\left\langle a.\left(0 \parallel \bar{a}\left\langle 0 \right\rangle\right) \right\rangle \parallel \left(a.\left(0 \parallel \bar{a}\left\langle 0 \right\rangle\right)\right)$.

Since $\parallel$ is commutative, associative and admits $\star$ as neutral element, a process may also be represented as a multiset of objects which may be either an emission $\bar{a}\left\langle P \right\rangle$, a reception $a.P$ or a variable i or x.

HOcore only has one reduction rule, consisting of the transmission of a message on a channel. It may be defined using the following construct:

Definition 2. Given two processes Q and R and a depth $i \in \mathbb{N}$, the substitution $R\left\{Q/i\right\}$ is a process defined by induction on the structure of R by:

- $\bar{b}\left\langle P \right\rangle \left\{Q/i\right\} = \bar{b}\left\langle P\left\{Q/i\right\} \right\rangle$
- $(b.P)\left\{Q/i\right\} = b.\left(P\left\{Q/(i+1)\right\}\right)$.
- $(P_1 \parallel P_2)\left\{Q/i\right\} = P_1\left\{Q/i\right\} \parallel P_2\left\{Q/i\right\}$.

- $i\{Q/i\} = Q$ and for all bound variable $j \neq i$, $j\{Q/i\} = j$.
- $x\{Q/i\} = x$.
- $\star\{Q/i\} = \star$.

This substitution is simple because of the locally nameless setting we use. We will see in 3.1.1 a notion of well-formedness to ensure that all de Buijn indices are bound.

Definition 3. Let P be a process of the form $\bar{a}\langle s\rangle \parallel (a.r) \parallel P'$. Let $t = (\bar{a}\langle s\rangle, a.r)$ be the pair emission / reception (such a pair is called a *valid transmission* of P. P' is called the *difference* between the process P and the transmission t, which is noted $P' = P \setminus t$). Then, there is a possible reduction on channel a which results in $P'' = r\{s/0\} \parallel P'$. This is noted $P \xrightarrow{t} P''$, or simply $P \to P''$ if no precision is required.

Examples:

- $\bar{a}\langle\star\rangle \parallel a.0 \xrightarrow{(\bar{a}\langle\star\rangle, a.0)} \star$.
- $b.(y \parallel a.1) \parallel \bar{b}\langle c.\bar{a}\langle 0\rangle\rangle \to y \parallel (a.c.\bar{a}\langle 0\rangle)$
- $\Omega = \bar{a}\langle a.(0 \parallel \bar{a}\langle 0\rangle)\rangle \parallel (a.(0 \parallel \bar{a}\langle 0\rangle)) \to \Omega$.

3 Definition of the abstract machine

3.1 Definitions and notations

3.1.1 Freedom and well-formedness

We begin by defining a notion of well-formedness for processes, which will be necessary to build the abstract machine since it can only work with well-formed processes.

Definition 4. The *freedom* of a process P, noted $f(P)$ is defined by induction with

- $f(\bar{a}\langle P\rangle) = f(P)$.
- $f(a.P) = f(P) - 1$.
- $f(P \parallel Q) = \max(f(P), f(Q))$.
- $f(i) = i + 1$.
- $f(x) = f(\star) = 0$.

Note that if there is no bound variable in P, $f(P) \leq 0$.

Definition 5. A process P is said to be *well-formed* when $f(P) \leq 0$.
A process P is said to be *closed* if it is well-formed and contains no free variable.

Examples:

- $f(\star) = 0$ so it is both closed and well-formed.
- $f(a.(b.0 \parallel \bar{b}\langle x \rangle)) = -1$ so it is well-formed. Since it has a free variable x, it is not closed.
- $f(a.\bar{a}\langle \star \rangle) = -1$ so it is well-formed, and it is also closed.
- $f(a.b.0 \parallel a.1) = 1$ so it is not well-formed, thus not a closed process.
- $f(\Omega) = 0$ where $\Omega = \bar{a}\langle a.(0 \parallel \bar{a}\langle 0 \rangle)\rangle \parallel (a.(0 \parallel \bar{a}\langle 0 \rangle))$.

3.1.2 Level

The notion of freedom can be approached by another point of view which consists in comparing the bound variables with their level, that is the number of binder above them. We now define these notions properly:

Definition 6. Let P and Q be processes and l a natural integer. We define the predicate "Q is at level l in P", noted $\text{Level}(P, Q, l)$, by

- (**A**) $\text{Level}(Q, Q, 0)$.
- (**P**) $\text{Level}(P, Q, l) \Rightarrow \text{Level}(R \parallel P, Q, l)$ if $P \neq \star$ and $R \neq \star$.
- (**S**) $\text{Level}(P, Q, l) \Rightarrow \text{Level}(\bar{a}\langle P \rangle, Q, l)$.
- (**R**) $\text{Level}(P, Q, l) \Rightarrow \text{Level}(a.P, Q, l+1)$.

Note that if Q is not a subprocess of P, then for all l, $\text{Level}(P, Q, l)$ is false. Moreover, Q may be at level l and at level l' in P with $l \neq l'$, for instance in $P = Q \parallel a.Q$.

Lemma 1. Let P, Q and R be processes with $P \neq \star$ and $Q \neq \star$. Then:

1. $\text{Level}(P \parallel Q, i, l) \Leftrightarrow \text{Level}(P, i, l)$ or $\text{Level}(Q, i, l)$.
2. $\text{Level}(\bar{a}\langle R \rangle, i, l) \Leftrightarrow \text{Level}(R, i, l)$
3. $\text{Level}(a.R, i, l+1) \Leftrightarrow \text{Level}(R, i, l)$

Proof of 1. In all three cases, the converse implication is a direct consequence of axiom respectively (**P**), (**S**) and (**R**). We now show the direct implication.

1. Let i and l be so that $\text{Level}(P \parallel Q, i, l)$. The only rule that may lead to $\text{Level}(P \parallel Q, i, l)$ is (**P**) because i is not of the form $P' \parallel Q'$ (otherwise (**A**) could also have applied).

   So there is P_0 and Q_0 so that $P \parallel Q = P_0 \parallel Q_0$ and either $\text{Level}(P_0, i, l)$ or $\text{Level}(Q_0, i, l)$. Since both cases are symmetric, let's suppose that $\text{Level}(P_0, i, l)$. As long as P_j is of the form $P' \parallel Q'$, the same method applies, so we may build a decreasing sequence $P_0, ..., P_n$ and $Q_0, ..., Q_n$ such that $P_j \parallel Q_j = P_{j+1} \parallel Q_{j+1}$ and $\text{Level}(P_j, i, l)$, until P_n is no more of the form $P' \parallel Q'$.

   Since P_n is an atom, it belongs either to P or to Q. Let's suppose it belongs to P, i.e. $P = P_n \parallel P'$. Then $\text{Level}(P_n, i, l)$ by construction of the P_j, thus, by (**P**), $\text{Level}(P, i, l)$. If it belongs to Q, then $\text{Level}(Q, i, l)$.

   Hence the implication.

2. Let i and l be so that $\text{Level}(\bar{a}\langle R\rangle, i, l)$. The only rule that may lead to $\text{Level}(\bar{a}\langle R\rangle, i, l)$ is (**S**) because i is not of the form $\bar{a}\langle P\rangle$, thus $\text{Level}(R, i, l)$.

3. Let i and l be so that $\text{Level}(a.R, i, l+1)$. The only rule that may lead to $\text{Level}(a.R, i, l+1)$ is (**R**) because i is not of the form $a.P$, thus $\text{Level}(R, i, l)$.

Lemma 2. If P contains at least one bound variable, then its freedom is so that $f(P) \geq 1 + \max\{i - l \mid \text{Level}(P, i, l),\ i \text{ bound variable of } P\}$.

Proof of 2. By induction on the structure of P:

- P cannot be $\star$ nor x because it contains at least one bound variable.
- If $P = \bar{a}\langle Q\rangle$, then by definition $f(P) = f(Q)$ and by hypothesis Q contains at least one bound variable so the induction gives $f(Q) \geq 1 + \max\{i - l \mid \text{Level}(Q, i, l)\}$. But by 2. of the last lemma, $\text{Level}(Q, i, l) \Leftrightarrow \text{Level}(P, i, l)$ so $f(P) \geq 1 + \max\{i - l \mid \text{Level}(P, i, l)\}$.
- If $P = a.Q$ then by definition, $f(P) = f(Q) - 1$ and by hypothesis Q contains at least one bound variable, so by induction $f(Q) \geq 1 + \max\{i - l \mid \text{Level}(Q, i, l)\}$. By 3. of the last lemma, $\text{Level}(Q, i, l) \Leftrightarrow \text{Level}(P, i, l+1)$ so

$$\begin{aligned} f(P) &\geq 1 + \max\{i - l \mid \text{Level}(Q, i, l)\} - 1 \\ &= \max\{i - l \mid \text{Level}(P, i, l+1)\} \\ &= \max\{1 + i - (l+1) \mid \text{Level}(P, i, l+1)\} \\ &= 1 + \max\{i - l' \mid \text{Level}(P, i, l')\} \end{aligned}$$

- If $P = Q \parallel R$ with both $Q \neq \star$ and $R \neq \star$, then, by definition, $f(P) = \max(f(Q), f(R))$. If Q contains at least one bound variable, then by induction, $f(Q) \geq 1 + \max\{i - l \mid \text{Level}(Q, i, l)\}$. The same is valid for R. Moreover, because of 1. of the last lemma, $\text{Level}(Q, i, l)$ or $\text{Level}(R, i, l) \Leftrightarrow \text{Level}(P, i, l)$. Thus, since all the bound variables of P are either in Q or R, $f(P) \geq 1 + \max\{i - l \mid \text{Level}(P, i, l)\}$.
- If $P = i$, then $f(P) = i + 1$. Moreover, by (**A**), $\text{Level}(P, i, 0)$, and i is the only bound variable of P, so $1 + \max\{i - l \mid \text{Level}(P, i, l)\} = 1 + i$.

Theorem 3. Let P be a process. If it is well-formed, then for all bound variable i within P, $\text{Level}(P, i, l) \Rightarrow i < l$.

Proof of 3. If there is no bound variable in P, the proposition holds. Otherwise, suppose there is a bound variable i of P and $\text{Level}(P, i, l)$ so that $i \geq l$. Then $1 + \max\{i - l \mid \text{Level}(P, i, l)\} \geq 1 + 0 = 1$ so P is ill-formed.

The reciproque is also true, but we do not need it.

3.1.3 Environments

The abstract machine for HOcore works using processes annotated with environments to simulate the reductions. We now define such environments.

Definition 7. An *environment* e is recursively defined as a list of annotated processes, an *annotated process* being a pair (process, environment):

$$e ::= \;[\,] \;\big|\; (P, e) :: e$$

For any environment e, we will note $\text{len}(e)$ the length of e.

Definition 8. An environment is said to be *well-formed* if all its elements are well-formed and, recursively, an annotated process (P, e) is said to be well-formed if e is well-formed and $f(P) \leq \text{len}(e)$.

Finally, we define a few notions of size to be used in inductions:

Definition 9.

- The *size of a process* P is defined recursively by
  - $\text{size}(\star) = 0$
  - $\text{size}(i) = \text{size}(x) = 1$

  - $\text{size}(P \parallel Q) = \text{size}(P) + \text{size}(Q)$
  - $\text{size}(\bar{a}\langle P\rangle) = 1 + \text{size}(P)$
  - $\text{size}(a.P) = 1 + \text{size}(P)$

- The *size of an environment* e is defined recursively by:
  - $\text{size}([\,]) = 0$
  - $\text{size}((P, e) :: l) = 1 + \text{size}(P) + \text{size}(e) + \text{size}(l)$

  Beware that the size of an environment is not its length.

- The *size of an annotated process* (P, e) is defined as $\text{size}((P, e)) = \text{size}(e) + \text{size}(P)$.

3.1.4 State of the machine

We may now define the abstract machine. Much like in the definition of a process, the following definition by induction may be replaced by a definition with a multiset.

Definition 10. A *state* M of the machine is defined by induction with

$$M ::= \ (\bar{a}\langle P\rangle, e) \mid (a.P, e) \mid M + M \mid x \mid \star$$

- The operation $+$ is commutative, associative and admits $\star$ as neutral element. It is the equivalent of the parallel composition for processes.
- $\star$ designates the empty state, similarly to the empty process.

Definition 11. The well-formedness of a state M is defined by induction on the structure of M:

- If $M = (P, e)$, it is well-formed if the annotated process (P, e) is well-formed, that is if e is well-formed and $f(P) \leq \text{len}(e)$.
- If $M = M_1 + M_2$, it is well-formed if both M_1 and M_2 are well-formed
- $\star$ and x are well-formed.

Examples:

- $\star$ is well-formed.
- $(\bar{a}\langle\star\rangle, e_1) + (a.0, e_2)$ is a well-formed state as long as e_1 and e_2 are well-formed environments.
- $(\bar{a}\langle 0\rangle, [\,])$ is an ill-formed annotated process, thus an ill-formed state.

3.2 Transitions between states

The only possible transition of the machine state consists in sending an emission to a reception in the same channel, similarly to the reduction of a process. The main difference comes from the treatment of the reception, where there is no substitution in the machine.

3.2.1 Abstraction and transmission

We first need to define a few operations:

Definition 12. To *abstract* a process P in an environment e consists in creating a new state noted $[\![(P, e)]\!]_{\mathcal{M}}$ and defined by induction on the structure of P:

- $[\![(\bar{a}\langle Q\rangle, e)]\!]_{\mathcal{M}} = (\bar{a}\langle Q\rangle, e)$.
- $[\![(a.Q, e)]\!]_{\mathcal{M}} \quad = (a.Q, e)$.
- $[\![(Q \parallel R, e)]\!]_{\mathcal{M}} = [\![(Q, e)]\!]_{\mathcal{M}} + [\![(R, e)]\!]_{\mathcal{M}}$.
- $[\![(i, e)]\!]_{\mathcal{M}} = [\![e[i]]\!]_{\mathcal{M}}$ if variable i is a valid index of e. Otherwise, the abstraction fails.
- $[\![(x, e)]\!]_{\mathcal{M}} = x$.
- $[\![(\star, e)]\!]_{\mathcal{M}} = \star$.

Definition 13. To *select a transmission* in a state M consists in choosing a channel a so that $M = (\bar{a}\langle s\rangle, e_1) + (a.r, e_2) + M'$.

If there is no channel a that satisfies the property that there is both an emission and a reception on channel a, then M is said to be *terminal*, and there is no transmission.

Otherwise, the pair $t = ((\bar{a}\langle s\rangle, e_1), (a.r, e_2))$ of annotated emission / annotated reception is said to be a *transmission*. Such a transmission is said to be valid for M if $(\bar{a}\langle s\rangle, e_1)$ is an emission and $(a.r, e_2)$ is a reception that both exist in M.

Definition 14. If $t = (S, R)$ is a valid transmission for state M, let's note $M = S + R + M'$. The *difference* between M and transmission t is defined as $M \setminus t = M'$.

3.2.2 Transition

We may now define a *transition* on a state M. The machine simulates the reduction by selecting a transmission if there is a valid one, then putting the emitted process in the environment of the reception so that it becomes a well-formed process, ready to be abstracted into a well-formed state. The preservation of well-formedness will be proven shortly after.

The abstraction we use here does not look inside of emissions and receptions, which is the main difference between the transition of the machine and the actual reduction of the process. However, the abstraction of a variable requires an environment: this is necessary to allow the machine and the process it represents to have equivalent reductions.

To preserve the original semantics of HOcore, the machine operates one transition at a time, so parallelism is only simulated in the non-deterministic choice of the transmission, but it could easily be adapted to perform multiple parallel transitions at a time. This non-determinism is another important difference with the Krivine Abstract Machine, and it comes from both the inherent non-determinism of HOcore and the choice made here not to implement a specific reduction strategy.

Definition 15.

If M is terminal, there is no possible transition.

Otherwise, let $t = (S, R)$ be a valid transmission of M. Let's note $R = (a.r, e)$ and $S = (\bar{a}\langle s\rangle, e')$.

The result of this transition is $M' = (M \setminus t) + [\![(r, (s, e') :: e)]\!]_{\mathcal{M}}$.

We will note $M \xrightarrow{t} M'$ or, more simply, $M \to M'$, similarly to the reduction of a process.

4 Preservation of well-formedness

We now prove that well-formedness is preserved through transition.

This result is necessary to prove that the machine is complete, which means that any sequence of reductions starting from a well-formed process matches an equivalent sequence of translations of the machine that represents the process.

4.1 Well-formed abstraction theorem

Theorem 4. Let (P, e) be a well-formed annotated process. Then $[\![(P, e)]\!]_{\mathcal{M}}$ is well-formed and the abstraction does not fail.

Proof of 4. Let's procede by complete induction on $\text{size}((P, e))$. We note $n = \text{len}(e)$.

- If $\text{size}((P, e)) = 0$ then $P = \star$, so $[\![(P, e)]\!]_{\mathcal{M}} = \star$ which is well-formed, thus the proposition holds.
- Otherwise,
  - If $P = x$ then $[\![(x, e)]\!]_{\mathcal{M}} = x$ is well-formed.
  - If $P = Q \parallel R$ with $\text{size}(Q) > 0$ and $\text{size}(R) > 0$, then we have both $\text{size}((Q, e)) < \text{size}((P, e))$, $\text{size}((R, e)) < \text{size}((P, e))$ and $[\![(Q \parallel R, e)]\!]_{\mathcal{M}} = [\![(Q, e)]\!]_{\mathcal{M}} + [\![(R, e)]\!]_{\mathcal{M}}$, so the proposition is true by induction.

- If $P = \bar{a}\langle Q \rangle$ or $a.Q$, then $[\![(P,e)]\!]_{\mathcal{M}} = (P,e)$. Since (P,e) is a well-formed annotated process, $[\![(P,e)]\!]_{\mathcal{M}}$ is well-formed.
- If $P = i$ is a variable, then $f(P) = i+1$, but (P,e) is a well-formed annotated process, so $f(P) \leq n$ i.e. $i < n$. Thus i is a valid index of e and this step of the abstraction does not fail.

  Let $(P', e') = e[i]$. Therefore, $[\![(P,e)]\!]_{\mathcal{M}} = [\![(P',e')]\!]_{\mathcal{M}}$.

  However, $\text{size}((P',e')) = \text{size}(e') + \text{size}(P')$ and $\text{size}(e) \geq 1 + \text{size}(P') + \text{size}(e')$, thus $\text{size}(e) > \text{size}((P',e'))$, so $\text{size}((P',e')) < \text{size}((P,e))$ and the hypothesis is valid by induction.

4.2 Transition theorem

We will now prove the main theorem regarding the preservation of well-formedness:

Theorem 5. If state M is well-formed and $M \to^* M'$, then M' is well-formed.

Proof of 5. By induction on the number n so that $M \to^n M'$:

- If $n = 0$, then $M' = M$ is well-formed.
- Suppose $M \xrightarrow{t} M'$ with $t = (S,R)$, $R = (a.r, e)$ and $S = (\bar{a}\langle s \rangle, e')$.

  Then $M' = (M \setminus t) + [\![(r, (s,e') :: e)]\!]_{\mathcal{M}}$. The annotated process $(a.r, e)$ is well-formed so $f(r) \leq \text{len}(e) + 1 = \text{len}((s,e') :: e)$. Moreover, since M is well-formed, e is also well-formed, and so is (s,e'), so environment $(s,e') :: e$ is well-formed. Thus, the annotated process $(r, (s,e') :: e)$ is well-formed.

  Thus, because of theorem 4, $[\![(r, (s,e') :: e)]\!]_{\mathcal{M}}$ is well-formed. So is $M \setminus t$ because all the annotated processes that appear in $M \setminus t$ also appear in M and M is well-formed.

  Thus, we may conclude that M' is well-formed.

5 Translation

We now define a translation of HOcore terms into states. This will allow to define and prove both soudness and completeness properties.

5.1 Forward translation

The forward translation converts a process into a state.

Definition 16. Given a well-formed HOcore process P, the *translation* of P into a state, noted $[\![P]\!]_{\mathcal{M}}$, is $[\![(P, [\,])]\!]_{\mathcal{M}}$.

Intuitevly, this means that all emissions and receptions Q that were put in parallel in P were mapped to $(Q, [\,])$, and are now separated by $+$ instead of $\|$. The free variables were kept as such. Thus, the resulting state is syntactically quite close to the original process.

5.2 Partial translation

The next definition is the core of the backward translation. It is parametrized by a natural integer, the depth d which accounts for the level of the currently translated subprocess, and comes from the use of de Bruijn variables.

Definition 17. The *partial translation* of a process P in an environment e, with depth d, noted $[\![(P, e)]\!]^d_{\mathcal{P}}$ is a process defined by induction on the structure of P:

- $[\![(\star, e)]\!]^d_{\mathcal{P}} = \star$
- $[\![(x, e)]\!]^d_{\mathcal{P}} = x$
- $[\![(Q \parallel R, e)]\!]^d_{\mathcal{P}} = \ [\![(Q, e)]\!]^d_{\mathcal{P}} \ \Big\| \ [\![(R, e)]\!]^d_{\mathcal{P}}$
- $[\![(\bar{a} \langle Q \rangle, e)]\!]^d_{\mathcal{P}} = \bar{a} \left\langle [\![(Q, e)]\!]^d_{\mathcal{P}} \right\rangle$
- $[\![(a.Q, e)]\!]^d_{\mathcal{P}} = a. \left([\![(Q, e)]\!]^{d+1}_{\mathcal{P}} \right)$
- $[\![(i, e)]\!]^d_{\mathcal{P}} = \begin{cases} i & \text{if } i < d \\ [\![e\,[i-d]]\!]^0_{\mathcal{P}} & \text{else, if } i-d \text{ is a valid index of } e \end{cases}$

  The translation fails otherwise.

In essence, the partial translation recursively translates all the processes and subprocesses into a new one, where all the bound variables that were mapped to a process in a precedent transition are now actually replaced by this process, found in the environment. The depth is used to keep track of which bound variable corresponds to which index of the environment.

The various properties of the partial translation will be discussed further.

5.3 Backward translation

We may now define the backward translation of a state toward a process.

Definition 18. Let M be a state. We define by induction on the structure of M its *backward translation*, noted $[\![M]\!]_{\mathcal{P}}$ with

- $[\![(P, e)]\!]_{\mathcal{P}} = [\![(P, e)]\!]^0_{\mathcal{P}}$.

- $[\![M_1 + M_2]\!]_{\mathcal{P}} = [\![M_1]\!]_{\mathcal{P}} \parallel [\![M_2]\!]_{\mathcal{P}}$.
- $[\![\star]\!]_{\mathcal{P}} = \star$.
- $[\![x]\!]_{\mathcal{P}} = x$.

If one of the partial translations fails, the translation fails.

Given the nature of the partial translation, the meaning of the backward translation is that all the successive transitions whose traces were kept in the environments are now executed all at once. To do this, the environments are translated into substitutions.

5.4 Equivalence of states

Definition 19. We will therefore say that two states M and M' are equivalent and we note $M \equiv M'$ when $[\![M]\!]_{\mathcal{P}} = [\![M']\!]_{\mathcal{P}}$. The *standard representant* of the equivalence class to which belongs M is noted $\widehat{M}$ and is defined as $\widehat{M} = [\![[\![M]\!]_{\mathcal{P}}]\!]_{\mathcal{M}}$.

Examples:

- Let $M_1 = (\bar{a}\left<\star\right>, [\,])$ and $M_2 = (\bar{a}\left<0\right>, [\star])$. Then $M_1 \equiv M_2$ and we have $[\![M_1]\!]_{\mathcal{P}} = [\![M_2]\!]_{\mathcal{P}} = \bar{a}\left<\star\right>$ and $\widehat{M_1} = \widehat{M_2} = M_1$.
- Let $M_1 = \big(b.\,(0 \parallel 1)\,, \big[\big(\bar{a}\left<0\right> \parallel 0 \parallel c.0, \big[\star\ ;\ (\bar{d}\left<\star\right>, [\,])\big]\big)\big]\big)$ and
  $M_2 = (b.\,(0 \parallel c.0 \parallel 1)\,, [((\bar{a}\left<\star\right>, [\,]))])$.
  Then $M_1 \equiv M_2$ and $[\![M_1]\!]_{\mathcal{P}} = [\![M_2]\!]_{\mathcal{P}} = b.(0 \parallel \bar{a}\left<\star\right> \parallel c.0)$.

5.5 Well-formed forward translation

The last theorem of the section ensures that the forward translation of a well-formed term leads to a well-formed state. This is important to ensure the soundness of the machine.

Theorem 6. If P is well-formed, then $[\![P]\!]_{\mathcal{M}}$ is well-formed.

Proof of 6. By induction on the structure of P

- If $P = \star$, then $[\![P]\!]_{\mathcal{M}} = \star$ is well-formed
- If $P = x$, then $[\![P]\!]_{\mathcal{M}} = x$ is well-formed
- If $P = Q \parallel R$ then both Q and R are well-formed, so, by induction, both $[\![Q]\!]_{\mathcal{M}}$ and $[\![R]\!]_{\mathcal{M}}$ are well-formed. Thus $[\![P]\!]_{\mathcal{M}} = [\![Q]\!]_{\mathcal{M}} + [\![R]\!]_{\mathcal{M}}$ is well-formed.
- If $P = \bar{a}\left<Q\right>$ or $P = a.Q$ then $f(P) \leq 0 = \text{len}([\,])$, thus $[\![P]\!]_{\mathcal{M}} = (P, [\,])$ is well-formed.
- P cannot be a bound variable since it is well-formed.

6 Partial translation properties

We establish here a few properties of the partial translation to further explain its meaning and to ensure that the translation is consistent and may be operated without failure under the right conditions.

6.1 Dual properties of the level

We begin with the following properties of the level which will be useful in the next proof.

Theorem 7. The dual following properties of the level hold:

- ($\mathbf{P'}$) $\text{Level}(P, Q \parallel R, l) \Rightarrow \text{Level}(P, Q, l)$.
- ($\mathbf{S'}$) $\text{Level}(P, \bar{a}\left\langle Q \right\rangle, l) \Rightarrow \text{Level}(P, Q, l)$.
- ($\mathbf{R'}$) $\text{Level}(P, a.Q, l) \Rightarrow \text{Level}(P, Q, l+1)$.

Proof of 7. By induction on the rule that led to the predicate. We give as example the proof for ($\mathbf{P'}$), the other ones are similar.
Let $Q' = Q \parallel R$. By induction on the rule that led to $\text{Level}(P, Q', l)$:

- ($\mathbf{A}$): $\text{Level}(P, Q', l)$, $l = 0$ and $P = Q'$. Then, by ($\mathbf{A}$), $\text{Level}(Q, Q, 0)$, thus by ($\mathbf{P}$), $\text{Level}(Q', Q, 0)$ that is $\text{Level}(P, Q, l)$.
- ($\mathbf{P}$): $\text{Level}(P', Q', l)$ with $P = P' \parallel R'$. Then, by induction, $\text{Level}(P', Q, l)$ thus by ($\mathbf{P}$), $\text{Level}(P, Q, l)$.
- ($\mathbf{S}$): $\text{Level}(P', Q', l)$ with $P = \bar{a}\left\langle P' \right\rangle$. Then, by induction, $\text{Level}(P', Q, l)$ thus by ($\mathbf{S}$), $\text{Level}(P, Q, l)$.
- ($\mathbf{R}$): $\text{Level}(P', Q', l-1)$ with $P = a.P'$. Then, by induction, $\text{Level}(P', Q, l-1)$ thus by ($\mathbf{R}$), $\text{Level}(P, Q, l)$.

6.2 Partial translation in an empty environment

We may now show the following result, which gives an intuition on the meaning of the partial translation.

Lemma 8. Let P be a well-formed process, let Q be any subterm that may appear within P and let l be an integer so that $\text{Level}(P, Q, l)$. Then $[\![(Q, [\,])]\!]^{l}_{\mathcal{P}} = Q$ and the partial translation does not fail.

Proof of 8. By induction on the structure of Q:

- If $Q = \star$ then $[\![(Q, [\,])]\!]^l_{\mathcal{P}} = \star$.
- If $Q = x$ then $[\![(Q, [\,])]\!]^l_{\mathcal{P}} = x$.
- If $Q = Q_1 \parallel Q_2$ then by $(\mathbf{P}')$, Level(P, Q_1, l) and Level(P, Q_2, l). Thus $[\![(Q_1 \parallel Q_2, [\,])]\!]^l_{\mathcal{P}} = [\![(Q_1, [\,])]\!]^l_{\mathcal{P}} \parallel [\![(Q_2, [\,])]\!]^l_{\mathcal{P}} = Q_1 \parallel Q_2 = Q$ by induction
- If $Q = \bar{a}\,\langle R\rangle$ then by $(\mathbf{S}')$, Level(P, R, l). Thus, $[\![(Q, [\,])]\!]^l_{\mathcal{P}} = \bar{a}\left\langle [\![(R, [\,])]\!]^l_{\mathcal{P}} \right\rangle = \bar{a}\,\langle R\rangle$ by induction.
- If $Q = a.R$ then by $(\mathbf{R}')$, Level$(P, R, l{+}1)$. Thus, $[\![(Q, [\,])]\!]^l_{\mathcal{P}} = a.\left([\![(R, [\,])]\!]^{l+1}_{\mathcal{P}}\right) = a.R$ by induction.
- If $Q = i$ is a variable, then, since P is well-formed, i is bound in P, so by theorem 3, $i < l$. Thus, it is kept untouched, the translation does not fail and $[\![(Q, [\,])]\!]^l_{\mathcal{P}} = i = Q$.

In particular, at level 0, if P is well-formed then $[\![(P, [\,])]\!]^0_{\mathcal{P}} = P$.

6.3 Consistency of the translation

The following theorem ensures that the standard representant $\widehat{M}$ of an equivalent class of $\equiv$ is well-defined.

Theorem 9. For all process P that is well-formed, $[\![[\![P]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = P$ and the partial translations do not fail.

Proof of 9. Let $M = [\![P]\!]_{\mathcal{M}}$. We show the theorem by induction on the structure of P:

- If $P = \bar{a}\,\langle Q\rangle$ or $a.Q$, then $M = (P, [\,])$ so $[\![M]\!]_{\mathcal{P}} = [\![(P, [\,])]\!]^0_{\mathcal{P}} = P$ because of the preceding lemma, and the partial translation does not fail.
- If $P = Q \parallel R$, then $M = M_1 + M_2$ where $M_1 = [\![Q]\!]_{\mathcal{M}}$ and $M_2 = [\![R]\!]_{\mathcal{M}}$, thus $[\![M]\!]_{\mathcal{P}} = [\![M_1]\!]_{\mathcal{P}} \parallel [\![M_2]\!]_{\mathcal{P}}$. By induction, $[\![M_1]\!]_{\mathcal{P}} = Q$ and $[\![M_2]\!]_{\mathcal{P}} = R$, hence the result.
- If $P = \star$ then $M = \star$ so $[\![M]\!]_{\mathcal{P}} = \star$ and the translation does not fail.
- If $P = x$ then $M = x$ so $[\![M]\!]_{\mathcal{P}} = x$ and the translation does not fail.
- P cannot be a bound variable because it is well-formed.

Because of this last theorem, $M \equiv \widehat{M}$ and $\widehat{\widehat{M}} = \widehat{M}$.

6.4 Stability of $\equiv$ by $+$

We prove a small corollary of the definition of the backward translation.

Lemma 10. For all well-formed states M_1, M_2 and M':
$M_1 \equiv M_2 \Leftrightarrow M_1 + M' \equiv M_2 + M'$.

Proof of 10. By definition of the backward translation, for all well-formed N_1 and N_2, we have $[\![N_1 + N_2]\!]_{\mathcal{P}} = [\![N_1]\!]_{\mathcal{P}} \parallel [\![N_2]\!]_{\mathcal{P}}$. Thus, if $M_1 \equiv M_2$, $[\![M_1 + M']\!]_{\mathcal{P}} = [\![M_1]\!]_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}} = [\![M_2]\!]_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}} = [\![M_2 + M']\!]_{\mathcal{P}}$.

Reciprocally, if $[\![M_1]\!]_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}} = [\![M_2]\!]_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}}$, then by identification of the structures, $[\![M_1]\!]_{\mathcal{P}} = [\![M_2]\!]_{\mathcal{P}}$.

6.5 Partial translation lemma

We now generalize the last results on failure of the partial translation to the most general case. Note that this result is optimal in the sense that if $f(P) > \text{len}(e) + d$, then the partial translation $[\![(P, e)]\!]^d_{\mathcal{P}}$ is bound to pfail.

Lemma 11. Let P, e and d be so that $f(P) \leq \text{len}(e) + d$. Then the partial translation $[\![(P, e)]\!]^d_{\mathcal{P}}$ does not fail.

Proof of 11. By induction on the structure of P, for all e and d that satisfy the condition:

- if $P = \star$ then $[\![(P, e)]\!]^d_{\mathcal{P}} = \star$.
- if $P = x$ then $[\![(P, e)]\!]^d_{\mathcal{P}} = x$.
- if $P = Q \parallel R$, then $f(Q) \leq f(P)$ and $f(R) \leq f(P)$ so, by induction, both partial translations $[\![(Q, e)]\!]^d_{\mathcal{P}}$ and $[\![(R, e)]\!]^d_{\mathcal{P}}$ do not fail. Thus, the partial translation $[\![(P, e)]\!]^d_{\mathcal{P}} = [\![(Q, e)]\!]^d_{\mathcal{P}} \Big\| [\![(R, e)]\!]^d_{\mathcal{P}}$ does not fail.
- if $P = \bar{a}\,\langle Q \rangle$, then $f(Q) = f(P) \leq \text{len}(e) + d$ so, by induction, $[\![(Q, e)]\!]^d_{\mathcal{P}}$ does not fail. Thus, $[\![(P, e)]\!]^d_{\mathcal{P}} = \bar{a}\left\langle [\![(Q, e)]\!]^d_{\mathcal{P}} \right\rangle$ does not fail.
- if $P = a.Q$, then $f(Q) = f(P) + 1 \leq \text{len}(e) + (d + 1)$ so, by induction, $[\![(Q, e)]\!]^{d+1}_{\mathcal{P}}$ does not fail. Thus, $[\![(P, e)]\!]^d_{\mathcal{P}} = a.\left([\![(Q, e)]\!]^{d+1}_{\mathcal{P}}\right)$ does not fail.
- if $P = i$ is a variable, then $f(P) = i + 1$, so $i - d < \text{len}(e)$. Thus, either $i < d$, in which case it remains untouched, either $i - d \in \{0, \text{len}(e) - 1\}$ in which case $i - d$ is a valid index of e. In both cases, the partial translation does not fail.

6.6 Well-formed translation property

We may now prove a generalized form of theorem 9.

Theorem 12. If M is well-formed, then the backward translation $[\![M]\!]_{\mathcal{P}}$ does not fail.

Proof of 12. This is quite straightforward, by induction on the structure of M:

- If $M = (P, e)$. Then, since M is well-formed, (P, e) is well-formed, which means that $f(P) \leq \text{len}(e)$. Thus, thanks to the lemma 11 we know that $[\![(P, e)]\!]^0_{\mathcal{P}}$ does not fail.
- If $M = M_1 + M_2$, both M_1 and M_2 are well-formed and by induction, the translations $[\![M_1]\!]_{\mathcal{P}}$ and $[\![M_2]\!]_{\mathcal{P}}$ do not fail, so $[\![M]\!]_{\mathcal{P}}$ does not fail either.
- If $M = \star$, $[\![M]\!]_{\mathcal{P}} = \star$ and the translation does not fail.
- If $M = x$, $[\![M]\!]_{\mathcal{P}} = x$ and the translation does not fail.

Considering this theorem, the transition theorem and the well-formed forward translation theorem, we have proven so far that starting from a well-formed process P, its forward translation $M = [\![P]\!]_{\mathcal{M}}$ exists and is well-formed, thus for any M' resulting from transitions of M i.e. $M \rightarrow^* M'$, M' is well-formed so its backward translation $[\![M']\!]_{\mathcal{P}}$ does not fail.

All that remains to show to prove the soundness of the machine is that there is a chain of reductions so that $P \rightarrow^* [\![M']\!]_{\mathcal{P}}$.

6.7 Elementary transmissions

We refine here the notion of transmission to prove another propery of stability by $\equiv$.

Definition 20. Given a transmission $t = (S, R)$,

- The *reduced transmission* of t, noted $[\![t]\!]_{\mathcal{P}}$, is defined as $[\![t]\!]_{\mathcal{P}} = ([\![S]\!]^0_{\mathcal{P}}, [\![R]\!]^0_{\mathcal{P}})$.
- The *standard transmission* of t, noted $\widehat{t}$, is defined as $\widehat{t} = \left(\left([\![S]\!]^0_{\mathcal{P}}, []\right)\ \left([\![R]\!]^0_{\mathcal{P}}, []\right)\right)$.

Theorem 13.

1. If t is a valid transmission for state M, then $[\![t]\!]_{\mathcal{P}}$ is a valid transmission of $[\![M]\!]_{\mathcal{P}}$ and $\widehat{t}$ is a valid transmission of $\widehat{M}$.
2. Let M be a state. If t' is a valid transmission of $[\![M]\!]_{\mathcal{P}}$ (respectively $\widehat{M}$) on channel a, then there is a valid transmission t of M on channel a so that $t' = [\![t]\!]_{\mathcal{P}}$ (respectively $t' = \widehat{t}$).
3. If t is a valid transmission for state M, then $[\![M \setminus t]\!]_{\mathcal{P}} = [\![M]\!]_{\mathcal{P}} \setminus [\![t]\!]_{\mathcal{P}}$ and $\widehat{M \setminus t} = \widehat{M} \setminus \widehat{t}$.

Proof of 13.

1. Let $t = (S, R)$ with $S = (\bar{a}\langle s\rangle, e_1)$ and $R = (a.r, e_2)$. Let M' be so that $M = S + R + M'$. S and R are both of the form (P, e) so $[\![S]\!]_{\mathcal{P}} = [\![S]\!]^0_{\mathcal{P}}$ and $[\![R]\!]_{\mathcal{P}} = [\![R]\!]^0_{\mathcal{P}}$. Thus $[\![M]\!]_{\mathcal{P}} = [\![S]\!]^0_{\mathcal{P}} \parallel [\![R]\!]^0_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}}$. Since we have $[\![t]\!]_{\mathcal{P}} = ([\![S]\!]^0_{\mathcal{P}}, [\![R]\!]^0_{\mathcal{P}})$, $[\![t]\!]_{\mathcal{P}}$ is a valid transmission for $[\![M]\!]_{\mathcal{P}}$.

Similarly, since $[\![S]\!]^0_{\mathcal{P}}$ is of the form $\bar{a}\langle P\rangle$ and $[\![R]\!]^0_{\mathcal{P}}$ is of the form $a.P$, we have $\left[\!\left[[\![S]\!]^0_{\mathcal{P}}\right]\!\right]_{\mathcal{M}} = \left([\![S]\!]^0_{\mathcal{P}}, []\right)$ and likewise for R. Thus $\widehat{M} = \left([\![S]\!]^0_{\mathcal{P}}, []\right) + \left([\![R]\!]^0_{\mathcal{P}}, []\right) + \widehat{M'}$ and $\widehat{t} = \left(\left([\![S]\!]^0_{\mathcal{P}}, []\right), \left([\![R]\!]^0_{\mathcal{P}}, []\right)\right)$ hence the result.

2. We consider the first case: $[\![M]\!]_{\mathcal{P}}$ and $[\![t]\!]_{\mathcal{P}}$. Let $t' = (\bar{a}\langle s'\rangle, a.r')$ and $[\![M]\!]_{\mathcal{P}} = \bar{a}\langle s'\rangle \parallel a.r' \parallel M''$. The definition of $[\![M]\!]_{\mathcal{P}}$ imposes the following structure for M: $M = (\bar{a}\langle s\rangle, e_1) + (a.r, e_2) + M'$, with $[\![(\bar{a}\langle s\rangle, e_1)]\!]^0_{\mathcal{P}} = \bar{a}\langle s'\rangle$, $[\![(a.r, e_2)]\!]^0_{\mathcal{P}} = a.r'$ and $[\![M']\!]_{\mathcal{P}} = M''$. Let $t = ((\bar{a}\langle s\rangle, e_1), (a.r, e_2))$. Then $[\![t]\!]_{\mathcal{P}} = t'$ and t is a valid transmission for M.

The second case (with $\widehat{M}$ and $\widehat{t}$) is very similar.

3. Let $t = (S, R)$ and $M = S + R + M'$. Then $[\![M \setminus t]\!]_{\mathcal{P}} = [\![M']\!]_{\mathcal{P}}$ and $[\![M]\!]_{\mathcal{P}} \setminus [\![t]\!]_{\mathcal{P}} = (\ [\![S]\!]_{\mathcal{P}} \parallel [\![R]\!]_{\mathcal{P}} \parallel [\![M']\!]_{\mathcal{P}}\ ) \setminus (\ [\![S]\!]_{\mathcal{P}} \parallel [\![R]\!]_{\mathcal{P}}\ ) = [\![M']\!]_{\mathcal{P}}$.

The second case is very similar too.

6.8 Partial translation theorem

The next theorem aims at linking the main notions seen until now: the abstraction of an annotated process, the partial translation and the backward translation.

Theorem 14. Let (P, e) be an annotated process. Then
$[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(P, e)]\!]^0_{\mathcal{P}}$.

Proof of 14. By induction on size $((P, e))$:

- If size $((P, e)) = 0$ then $P = \star$, thus $[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(P, e)]\!]^0_{\mathcal{P}} = \star$.
- Otherwise,
  - If $P = x$ then $[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![P]\!]^0_{\mathcal{P}} = x$.
  - If $P = Q \parallel R$ with neither $Q = \star$ nor $R = \star$, then by definition, $[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![[\![(Q, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} \parallel [\![[\![(R, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}}$ and by induction, $[\![[\![(Q, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(Q, e)]\!]^0_{\mathcal{P}}$ and $[\![[\![(R, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(R, e)]\!]^0_{\mathcal{P}}$. Thus, $[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(Q, e)]\!]^0_{\mathcal{P}} \parallel [\![(R, e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^0_{\mathcal{P}}$.
  - If $P = \bar{a}\langle Q\rangle$ or $P = a.Q$, then $[\![(P, e)]\!]_{\mathcal{M}} = (P, e)$. Therefore, by definition, $[\![[\![(P, e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![(P, e)]\!]^0_{\mathcal{P}}$.

- If $P = i$ is a variable, then
  - Either $i < \text{len}(e)$, in which case i is a valid index of e. Thus, $[\![(P,e)]\!]_{\mathcal{M}} = [\![e\,[i]]\!]_{\mathcal{M}}$ so $[\![[\![(P,e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}} = [\![[\![e\,[i]]\!]_{\mathcal{M}}]\!]_{\mathcal{P}}$ and $[\![(P,e)]\!]^0_{\mathcal{P}} = [\![e\,[i]]\!]^0_{\mathcal{P}}$. But $\text{size}\,(e\,[i]) < \text{size}\,((P,e))$ so, by induction, the proposition holds.
  - Either $i \geq \text{len}(e)$ in which case both $[\![(P,e)]\!]_{\mathcal{M}}$ and $[\![(P,e)]\!]^0_{\mathcal{P}}$ fail.

7 Soundness

We are now going to prove that the machine is sound, in the sense that any transition performed by the machine matches a possible transition of the HOcore process it is translated into.

To do this, we begin by showing that any transition performed by the machine matches a possible transition of its standard representant (main lemma). Afterwards, we may conclude using the close relation there is between a standard representant and its backward translation.

7.1 Simplification lemma

The following lemma taken with $c = d = 0$ means that if s is a well-formed annotated process, that is suited to be translated into a process, then the obtained process may not be re-translated into anything other than itself.

This is quite intuitive since the obtained process should be well-formed, thus independant of the environment it may be annotated with.

Lemma 15. Let $S = (P, e)$ be an annotated process and depth c be so that $f(P) \leq \text{len}(e) + c$ and e is well-formed. Let f be an environment. Then, for all $d \geq c$, $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = [\![S]\!]^c_{\mathcal{P}}$.

Proof of 15. By induction on $\text{size}\,((P,e))$:

- If $\text{size}\,((P,e)) = 0$ then $P = \star$, thus $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = [\![S]\!]^c_{\mathcal{P}} = \star$.
- Otherwise,
  - If $P = x$ then $[\![S]\!]^c_{\mathcal{P}} = x$ thus $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = x$.
  - If $P = Q \parallel R$ then $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = [\![([\![(Q,e)]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} \parallel [\![([\![(R,e)]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}}$ thus, by induction, $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = [\![(Q,e)]\!]^c_{\mathcal{P}} \parallel [\![(R,e)]\!]^c_{\mathcal{P}} = [\![S]\!]^c_{\mathcal{P}}$.
  - If $P = \bar{a}\,\langle Q\rangle$ then $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = \bar{a}\left\langle [\![([\![(Q,e)]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} \right\rangle$. By induction, $[\![([\![(Q,e)]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = [\![(Q,e)]\!]^c_{\mathcal{P}}$.

– If $P = a.Q$ then $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = a.\left(\left[\!\!\left[([\![(Q,e)]\!]^{c+1}_{\mathcal{P}}, e')\right]\!\!\right]^{d+1}_{\mathcal{P}}\right)$. By induction, $\left[\!\!\left[([\![(Q,e)]\!]^{c+1}_{\mathcal{P}}, e')\right]\!\!\right]^{d+1}_{\mathcal{P}} = [\![(Q,e)]\!]^{c+1}_{\mathcal{P}}$.

– If $P = i$ is a variable, then

  * Either $i < c$, so $[\![S]\!]^c_{\mathcal{P}} = i$ and $i < c \le d$, thus $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = i$.
  * Either $i \ge c$ in which case, since $f(P) = i + 1 \le \text{len}(e) + c$, $i - c$ is a valid index of e so the translations do not fail. Thus $[\![S]\!]^c_{\mathcal{P}} = [\![e\,[i-c]]\!]^0_{\mathcal{P}}$ and $[\![([\![S]\!]^c_{\mathcal{P}}, e')]\!]^d_{\mathcal{P}} = \left[\!\!\left[([\![e\,[i-c]]\!]^0_{\mathcal{P}}, e')\right]\!\!\right]^d_{\mathcal{P}}$. By induction, with $s' = e\,[i-c]$ and $c' = 0 \le d$, we have $\left[\!\!\left[([\![e\,[i-c]]\!]^0_{\mathcal{P}}, e')\right]\!\!\right]^d_{\mathcal{P}} = [\![e\,[i-c]]\!]^0_{\mathcal{P}}$ hence the result.

7.2 One-step theorem

The following theorem links the substitution, used in the definition of the reduction of an HOcore process, with the partial translation in an environment made of a singleton.

Theorem 16. Let i be a natural integer, s be a well-formed process, $\widetilde{s} = [(s, [\,])]$ and r be a process so that $f(r) \le i + 1$. Then, $r\{s/i\} = [\![(r, \widetilde{s})]\!]^i_{\mathcal{P}}$.

Proof of 16. By induction on the structure of r:

- If $r = \star$, then $r\{s/i\} = [\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = \star$.
- If $r = x$, then $r\{s/i\} = [\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = x$.
- If $r = P \parallel Q$, then $r\{s/i\} = P\{s/i\} \parallel Q\{s/i\}$, we have both $f(P) \le f(r)$ and $f(Q) \le f(r)$ and $[\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = [\![(P, \widetilde{s})]\!]^i_{\mathcal{P}} \parallel [\![(Q, \widetilde{s})]\!]^i_{\mathcal{P}}$. Thus, by induction, $P\{s/i\} = [\![(P, \widetilde{s})]\!]^i_{\mathcal{P}}$ and $Q\{s/i\} = [\![(Q, \widetilde{s})]\!]^i_{\mathcal{P}}$ hence the result.
- If $r = \bar{a}\,\langle P\rangle$ then $r\{s/i\} = \bar{a}\,\langle P\{s/i\}\rangle$, $[\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = \bar{a}\left\langle [\![(P, \widetilde{s})]\!]^i_{\mathcal{P}}\right\rangle$ and $f(P) = f(r)$ thus the induction allows to conclude.
- If $r = a.P$ then $r\{s/i\} = a.\,(r\{s/(i+1)\})$ and $[\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = a.\left([\![(P, \widetilde{s})]\!]^{i+1}_{\mathcal{P}}\right)$, and $f(P) = f(r) + 1 \le (i+1) + 1$ thus the induction allows to conclude.
- If r is a variable, then
  – If $r = i$ then $r\{s/i\} = s$ and $[\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = [\![(s, [\,])]\!]^0_{\mathcal{P}} = s$ since it is the partial translation of a well-formed annotated process in an empty environment.
  – If $r < i$ then $r\{s/i\} = [\![(r, \widetilde{s})]\!]^i_{\mathcal{P}} = r$.
  – $r > i$ is an impossible case because $f(r) \le i + 1$.

7.3 Transmission theorem

The transmission theorem establishes an equality between two terms whose translations did not happen at the same time.

Using the upcoming notations, P designates a process which is translated only once after multiple transitions, whereas in P', R was first partially translated, and is now translated again after one last transition.

The equality shows that the order in which the partial translations and the transitions were operated does not change the equivalence class of the resulting state.

Theorem 17. Let S be a well-formed annotated process, d a natural integer and $R = (r, e)$, an annotated process so that $f(r) \leq \text{len}(e) + d$ and e is well-formed. Let $\widetilde{S} = \left[\left(\llbracket S \rrbracket_{\mathcal{P}}^{0}, [\,]\right)\right]$.

Let $P = \llbracket (r, S :: e) \rrbracket_{\mathcal{P}}^{d}$ and $P' = \left\llbracket \left(\llbracket R \rrbracket_{\mathcal{P}}^{d+1}, \widetilde{S}\right) \right\rrbracket_{\mathcal{P}}^{d}$.

Then $P = P'$.

Proof of 17. By induction on the structure of r:

- If $r = \star$ or x, $P = P' = r$.
- If $r = Q \parallel R$, then we have $P = \overbrace{\llbracket (Q, S :: e) \rrbracket_{\mathcal{P}}^{d}}^{P_1} \parallel \overbrace{\llbracket (R, S :: e) \rrbracket_{\mathcal{P}}^{d}}^{P_2}$ and $P' = \overbrace{\left\llbracket \left(\llbracket (Q, e) \rrbracket_{\mathcal{P}}^{d+1}, \widetilde{S}\right) \right\rrbracket_{\mathcal{P}}^{d}}^{P_1'} \parallel \overbrace{\left\llbracket \left(\llbracket (R, e) \rrbracket_{\mathcal{P}}^{d+1}, \widetilde{S}\right) \right\rrbracket_{\mathcal{P}}^{d}}^{P_2'}$.
  By induction, $P_1 = P_1'$ and $P_2 = P_2'$, thus $P = P'$.
- If $r = \bar{a}\,\langle Q \rangle$, then $P = \bar{a}\left\langle \overbrace{\llbracket (Q, S :: e) \rrbracket_{\mathcal{P}}^{d}}^{X_1} \right\rangle$. Moreover, $\llbracket R \rrbracket_{\mathcal{P}}^{d+1} = \bar{a}\left\langle \llbracket (Q, e) \rrbracket_{\mathcal{P}}^{d+1} \right\rangle$, thus $P' = \bar{a}\left\langle \underbrace{\left\llbracket \left(\llbracket (Q, e) \rrbracket_{\mathcal{P}}^{d+1}, \widetilde{S}\right) \right\rrbracket_{\mathcal{P}}^{d}}_{X_2} \right\rangle$. By induction, $X_1 = X_2$ so $P = P'$.
- If $r = a.Q$, then $P = a.\left(\overbrace{\llbracket (Q, S :: e) \rrbracket_{\mathcal{P}}^{d+1}}^{X_1} \right)$. Moreover, $\llbracket R \rrbracket_{\mathcal{P}}^{d+1} = a.\left(\llbracket (Q, e) \rrbracket_{\mathcal{P}}^{d+2}\right)$ thus $P' = a.\left(\underbrace{\left\llbracket \left(\llbracket (Q, e) \rrbracket_{\mathcal{P}}^{d+2}, \widetilde{S}\right) \right\rrbracket_{\mathcal{P}}^{d+1}}_{X_2} \right)$. By induction, $X_1 = X_2$ so $P = P'$.

- If $r = i$ is a variable, then
  - Either $i < d+1$, in which case $[\![R]\!]_{\mathcal{P}}^{d+1} = i$, thus
    - If $i < d$ then $P = P' = i$.
    - If $i = d$ then $P = [\![(d, S :: e)]\!]_{\mathcal{P}}^{d} = [\![S]\!]_{\mathcal{P}}^{0}$ and $P' = \left[\!\!\left[\left([\![R]\!]_{\mathcal{P}}^{d+1}, \widetilde{S}\right)\right]\!\!\right]_{\mathcal{P}}^{d} = \left[\!\!\left[\left(d, \left[([\![S]\!]_{\mathcal{P}}^{0}, [\,])\right]\right)\right]\!\!\right]_{\mathcal{P}}^{d} = \left[\!\!\left[\left([\![S]\!]_{\mathcal{P}}^{0}, [\,]\right)\right]\!\!\right]_{\mathcal{P}}^{0} = [\![S]\!]_{\mathcal{P}}^{0}$ because of lemma 15.
  - Either $i \geq d+1$, in which case $[\![R]\!]_{\mathcal{P}}^{d+1} = [\![e\,[i-d-1]]\!]_{\mathcal{P}}^{0}$, same as $P = [\![(r, S :: e)]\!]_{\mathcal{P}}^{d} = [\![e\,[i-d-1]]\!]_{\mathcal{P}}^{0}$.
    But $\left[\!\!\left[\left([\![R]\!]_{\mathcal{P}}^{d+1}, \widetilde{S}\right)\right]\!\!\right]_{\mathcal{P}}^{d} = \left[\!\!\left[\left([\![e\,[i-d-1]]\!]_{\mathcal{P}}^{0}, \widetilde{S}\right)\right]\!\!\right]_{\mathcal{P}}^{d} = [\![e\,[r-d-1]]\!]_{\mathcal{P}}^{0}$ since e is made of well-formed annotated processes and because of lemma 15. so $P = P'$.

7.4 Main lemma

The main lemma is the last step before the proof of soundness. It reduces the problem of dealing with any machine state M to a more specific state, $\widehat{M}$ whose properties are much easier to study since all its environments are empty.

Theorem 18. If M is a well-formed machine state and $M \xrightarrow{t} M_1$, then $\widehat{M} \xrightarrow{\widehat{t}} M_2$ where $M_2 \equiv M_1$.

Proof of 18. Let $(S, R) = t$. We have already proven that $\widehat{t}$ was a valid transmission of $\widehat{M}$ (see Elementary transmissions).

Let's now write $M_1 = (M \setminus t) + M_1'$, like in the definition of a transition. Similarly, let's write $M_2 = (\widehat{M} \setminus \widehat{t}) + M_2'$. We have also seen that $\widehat{M \setminus t} = \widehat{M} \setminus \widehat{t}$, so $(M \setminus t) \equiv (\widehat{M} \setminus \widehat{t})$.

To prove that $M_1 \equiv M_2$, we now only have to prove that $M_1' \equiv M_2'$, because of the stability by $+$. Let $(a.r, e) = R$ and $(\bar{a}\,\langle s \rangle, e') = S$. We can note $M_1' = [\![(r, (s, e') :: e)]\!]_{\mathcal{M}}$.

Moreover, since $\widehat{t} = \left(\left(\left(\bar{a}\left\langle [\![(s, e')]\!]_{\mathcal{P}}^{0} \right\rangle\right), [\,]\right), \left(a.\left([\![(r, e)]\!]_{\mathcal{P}}^{1}\right), [\,]\right)\right)$, we can rewrite $M_2' = \left[\!\!\left[\left([\![(r, e)]\!]_{\mathcal{P}}^{1}, \widetilde{S}\right)\right]\!\!\right]_{\mathcal{M}}$ with $\widetilde{S} = \left[\left([\![(s, e')]\!]_{\mathcal{P}}^{0}, [\,]\right)\right]$.

Both forms are well-defined thanks to theorem 4.

Moreover, $[\![M_1']\!]_{\mathcal{P}} = [\![(r, (s, e') :: e)]\!]_{\mathcal{P}}^{0}$ and $[\![M_2']\!]_{\mathcal{P}} = \left[\!\!\left[\left([\![(r, e)]\!]_{\mathcal{P}}^{1}, \widetilde{S}\right)\right]\!\!\right]_{\mathcal{P}}^{0}$ because of theorem 14. Thus, theorem 17 applies, with depth 0.

Hence $[\![M_1']\!]_{\mathcal{P}} = [\![M_2']\!]_{\mathcal{P}}$, that is $M_1' \equiv M_2'$, so $M_1 \equiv M_2$.

7.5 Soundness property

We may now prove that the machine is sound.

Theorem 19. Let M be a well-formed state. If $M \to^* M'$ then $[\![M]\!]_{\mathcal{P}} \to^* [\![M']\!]_{\mathcal{P}}$

Proof of 19. By induction on n so that $M \to^n M'$:

- If $n = 0$, $M = M'$ so $[\![M]\!]_{\mathcal{P}} \to^0 [\![M']\!]_{\mathcal{P}}$.
- If $M \to^{n+1} M'$, let N be so that $M \to^n N \xrightarrow{t} M'$. By induction, $[\![M]\!]_{\mathcal{P}} \to^* [\![N]\!]_{\mathcal{P}}$. Let's show that $[\![N]\!]_{\mathcal{P}} \to [\![M']\!]_{\mathcal{P}}$. First, note that since M is well-formed, so are M' and N because of theorem 5, and consequently, because of property 12, both $[\![N]\!]_{\mathcal{P}}$ and $[\![M']\!]_{\mathcal{P}}$ exist.

  The main lemma implies that $\widehat{N} \xrightarrow{\widehat{t}} N'$ and $N' \equiv M'$.

  Let $(S, R) = \widehat{t}$, that may be decomposed into $(\bar{a}\,\langle s' \rangle\,, [\,]) = S$ and $(a.r', [\,]) = R$.

  Let's note $N' = (\widehat{N} \setminus \widehat{t}) + N''$. By definition, $N'' = [\![(r', [(s', [\,])])]\!]_{\mathcal{M}}$. Because of theorem 14, $[\![N'']\!]_{\mathcal{P}} = [\![(r', [(s', [\,])])]\!]^0_{\mathcal{P}}$.

  Thus, because of theorem 16, $[\![N'']\!]_{\mathcal{P}} = r'\{s'/0\}$, hence
  $[\![N']\!]_{\mathcal{P}} = \left[\!\!\left[\widehat{N} \setminus \widehat{t}\right]\!\!\right]_{\mathcal{P}} + [\![N'']\!]_{\mathcal{P}} = \left(\left[\!\!\left[\widehat{N}\right]\!\!\right]_{\mathcal{P}} \setminus [\![\widehat{t}]\!]_{\mathcal{P}}\right) + r'\{s'/0\}$.

  Let $\widetilde{t} = [\![\widehat{t}]\!]_{\mathcal{P}} = (\bar{a}\,\langle s' \rangle\,, a.r')$. Then, by definition of the reduction, $\left[\!\!\left[\widehat{N}\right]\!\!\right]_{\mathcal{P}} \xrightarrow{\widetilde{t}} [\![N']\!]_{\mathcal{P}}$, thus $[\![N]\!]_{\mathcal{P}} \xrightarrow{\widetilde{t}} [\![M']\!]_{\mathcal{P}}$.

8 Completeness

We now prove that the machine is complete, in the sense that any transition performable by the terms matches a possible transition of the machine. To put it more formally, we prove the following theorem:

Theorem 20. If P is a well-formed term and $P \xrightarrow{t'} P'$, then for all well-formed M so that $[\![M]\!]_{\mathcal{P}} = P$, there is a valid transmission t of M so that $[\![t]\!]_{\mathcal{P}} = t'$ and $M \xrightarrow{t} M'$ with $[\![M']\!]_{\mathcal{P}} = P'$.

Proof of 20. The existence of a valid transmission t of M so that $[\![t]\!]_{\mathcal{P}} = t'$ comes from a property of the elementary transmissions. Considering the unique M' so that $M \xrightarrow{t} M'$, the soundness of the machine ensures that there is a P'' so that $P \xrightarrow{[\![t]\!]_{\mathcal{P}}} P''$ and $[\![M']\!]_{\mathcal{P}} = P''$. But $P \xrightarrow{[\![t]\!]_{\mathcal{P}}} P'$ already, so $P' = P''$.

9 Machine bisimilarity

In this section, we define a notion of bisimilarity between abstract machines.

9.1 Definition for HOcore

We begin by recalling the definition of IO bisimilarity for HOcore. To do this, we need to change the Labelled Transition System we used up to now. Let's now consider that the old LTS, whose labels were of the form $P \xrightarrow{t} Q$ with t a valid transmission of P, does not exist.

Definition 21. The LTS of HOcore consists in two possible transitions: input transitions of the form $P \xrightarrow{a} Q$ or output transitions of the form $P \xrightarrow{\bar{a}\langle R\rangle} Q$. They are defined by three rules:

1. $a.P \xrightarrow{a} P$
2. $\bar{a}\langle P\rangle \xrightarrow{\bar{a}\langle P\rangle} \star$
3. for α being either a or $\bar{a}\langle R\rangle$, $\dfrac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q}$.

Definition 22. A symmetric relation $\mathcal{R}$ on HOcore processes is

- a *higher-order output bisimulation* if $P\mathcal{R}Q$ and $P \xrightarrow{\bar{a}\langle P''\rangle} P'$ imply that there are Q' and Q'' so that $Q \xrightarrow{\bar{a}\langle Q''\rangle} Q'$ with $P'\mathcal{R}Q'$ and $P''\mathcal{R}Q''$.
- an *open bisimulation* if whenever $P\mathcal{R}Q$:
  - $P \xrightarrow{a} P'$ implies that there is Q' so that $Q \xrightarrow{a} Q'$ and $P'\mathcal{R}Q'$.
  - $P = v \parallel P'$ implies that there is Q' so that $Q = v \parallel Q'$ and $P'\mathcal{R}Q'$, where v is either a bound variable i or a free variable x.

Definition 23. *IO bisimilarity*, written $\sim^{\circ}_{\mathtt{IO}}$, is defined as the largest relation on HOcore processes that is a higher-order output bisimulation and is open.

It has been proven in [4] that IO bisimilarity coincides with strong barbed congruence and that both are decidable.

9.2 Definition for the abstract machine

We now define a notion of bisimilarity for the abstract machine.

Such a notion is defined and used in [1], where it uses the locally nameless convention with a fresh variables factory in order to deal with bound variables. However, this approach is impossible in our case because it requires that there be a unique environment for the whole machine, whereas here there are as many environments as there are atoms of the form $(\bar{a}\langle P\rangle, e)$ or $(a.P, e)$ in the machine.

Indeed, with $M = (a.P, e) + N$ for instance, we would have $[\![M]\!]_{\mathcal{P}} \xrightarrow{a} [\![(P, e)]\!]^1_{\mathcal{P}} \parallel [\![N]\!]_{\mathcal{P}}$ whereas, if n represents a fresh variables factory, $(M, n) \xrightarrow{a} ([\![(P, n :: e)]\!]_{\mathcal{M}} + N, n+1)$. But $([\![(P, n :: e)]\!]_{\mathcal{M}} + N, n+1)$ cannot correspond to $[\![(P, e)]\!]^1_{\mathcal{P}} \parallel [\![N]\!]_{\mathcal{P}}$ because n only appears in one part of the machine, where it ought to appear in both to establish a correspondence.

Instead of this method, we use another way consisting in the transformation of the de Bruijn variables into free variables that keep track of the former value of the variable. To define this properly, we need to extend the syntax.

9.2.1 Extension of syntax

We extend the definition of abstract machine to be able to create a valid notion of machine bisimilarity.

First, we extend the syntax of environments:

$$e = (P, e) :: e \;\big|\; \odot :: e \;\big|\; []$$

The new symbol $\odot$ represents a hole in the environment. We also extend the associated size with $\text{size}(\odot :: e) = \text{size}(e)$.

We may then adapt the notion of abstraction to change the case of the variable:

- If i is a valid index of e and $e[i] \neq \odot$, then $[\![(i, e)]\!]_{\mathcal{M}} = [\![e[i]]\!]_{\mathcal{M}}$
- If i is a valid index of e and $e[i] = \odot$, or if i is not a valid index of e, then $[\![(i, e)]\!]_{\mathcal{M}} = \underline{i}$. For all bound variable i, $\underline{i}$ is a free variable. Moreover, $\underline{i}$ is considered different than all the other free variables that may appear in the machine.

The partial translation is adapted similarly, with $[\![(i, e)]\!]^d_{\mathcal{P}} = \underline{i}$ when $i < d$ or when $i - d$ is an invalid index of e or when $e[i - d] = \odot$.

Note that the abstractions and the partial translations may no more fail. Consequently, neither forward nor backward translations may fail either.

9.2.2 New LTS

Similarly to the definition of IO bisimilarity, we change the old LTS, whose labels were of the form $M \xrightarrow{t} M'$ with t a valid transmission of M, into a new one.

Definition 24. The LTS of the machine consists in the following different possible transitions:

- $(a.r, e) + M \xrightarrow{a} [\![(r, \odot :: e)]\!]_{\mathcal{M}} + M$.
- $(\bar{a}\langle s\rangle, e) + M \xrightarrow{(\overline{a})} [\![(s, e)]\!]_{\mathcal{M}}$.
- $(\bar{a}\langle s\rangle, e) + M \xrightarrow{\overline{a}+} M$.
- $x + M \xrightarrow{x} M$.
- $\star \xrightarrow{\star}$.

There are five possible types of flags:

- $\star$ is the *terminal flag*.
- $(\overline{a})$, $\overline{a}+$, a and x are called the *non-terminal flags*. a ranges over the channel names and x over the free variables.

9.2.3 Machine bisimulation

We use the notion of machine bisimulation developped in [1], conditioned by the LTS for the machines we just saw.

Let f range over the non-terminal flags of the machine.

Definition 25. A symmetric relation $\mathcal{R}$ is a *machine bisimulation* if $M_1 \mathcal{R} M_2$ implies:

- If $M_1 \xrightarrow{f} M_1'$ then there is M_2' so that $M_2 \xrightarrow{f} M_2'$ and $M_1' \mathcal{R} M_2'$.
- If $M_1 \xrightarrow{\star}$ then $M_2 \xrightarrow{\star}$.

Machine bisimilarity, noted $\approx_m$ is the largest machine bisimulation.

9.3 Intermediate lemma

We now prove a theorem that links the extensions of the syntax and the partial translation.

Definition 26. The environment $\overbrace{\odot :: \odot :: ... :: \odot}^{n \text{ times}} :: e$ is noted $\odot^n :: e$.

Theorem 21. Let P be a process. Then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}}$.

Proof of 21. By induction on the structure of P:

- If $P = \star$ then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}} = \star$.
- If $P = x$ then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}} = x$.
- If $P = Q \parallel R$ then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(Q, \odot^n :: e)]\!]^0_{\mathcal{P}} \parallel [\![(R, \odot^n :: e)]\!]^0_{\mathcal{P}}$ thus, by induction $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(Q, e)]\!]^n_{\mathcal{P}} \parallel [\![(R, e)]\!]^n_{\mathcal{P}}$ so finally $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}}$.
- If $P = \bar{a}\langle Q\rangle$ then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = \bar{a}\left\langle [\![(Q, \odot^n :: e)]\!]^0_{\mathcal{P}}\right\rangle$ and by induction we have $[\![(Q, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(Q, e)]\!]^n_{\mathcal{P}}$ so $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}}$.
- If $P = a.Q$ then $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = a.\left([\![(Q, \odot^n :: e)]\!]^1_{\mathcal{P}}\right)$ and by induction, $[\![(Q, \odot^n :: e)]\!]^1_{\mathcal{P}} = [\![(Q, \odot^{n+1} :: e)]\!]^0_{\mathcal{P}} = [\![(Q, e)]\!]^{n+1}_{\mathcal{P}}$ so $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = a.\left([\![(Q, e)]\!]^{n+1}_{\mathcal{P}}\right) = [\![(P, e)]\!]^n_{\mathcal{P}}$.
- If $P = i$ then
  - Either $i \geq \text{len}(e) + n$ in which case $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}} = \underline{i}$.
  - Either $i < n$ in which case $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}} = \underline{i}$.
  - Either $n \leq i < n+\text{len}(e)$ in which case $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![(P, e)]\!]^n_{\mathcal{P}} = [\![e\,[i-n]]\!]^0_{\mathcal{P}}$.

As a consequence of this theorem, we have the following result which will serve as an intermediate lemma in the proof of the strong equivalence of bisimilarities:

Lemma 22. $[\![(P, \widehat{\odot^n :: e})]\!]_{\mathcal{M}} = [\![[\![(P, e)]\!]^n_{\mathcal{P}}]\!]_{\mathcal{M}}$

Proof of 22. Because of the theorem 14 we have $[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}} = [\![[\![(P, \odot^n :: e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}}$ so

$$\begin{aligned}
[\![(P, \widehat{\odot^n :: e})]\!]_{\mathcal{M}} &= \left[\!\left[[\![[\![(P, \odot^n :: e)]\!]_{\mathcal{M}}]\!]_{\mathcal{P}}\right]\!\right]_{\mathcal{M}} \\
&= \left[\!\left[[\![(P, \odot^n :: e)]\!]^0_{\mathcal{P}}\right]\!\right]_{\mathcal{M}} \\
&= [\![[\![(P, e)]\!]^n_{\mathcal{P}}]\!]_{\mathcal{M}} \text{ because of the preceding lemma.}
\end{aligned}$$

10 Equivalence of bisimilarities

In this last section, we prove the equivalence between machine bisimilarity and IO bisimilarity for processes, modulo translation.

10.1 Weak equivalence

We now show the following theorem:

Theorem 23. Let P and Q be well-formed processes. Then

$$P \sim^{\circ}_{\mathtt{IO}} Q \Leftrightarrow [\![P]\!]_{\mathcal{M}} \approx_m [\![Q]\!]_{\mathcal{M}}$$

Proof of 23.

- Let $\approx' = \{([\![P]\!]_{\mathcal{M}}, [\![Q]\!]_{\mathcal{M}}) \mid P \sim^{\circ}_{\mathtt{IO}} Q,\ P$ and Q well-formed$\}$. We show that $\approx'$ is a machine bisimulation. To do this, let P and Q be so that $[\![P]\!]_{\mathcal{M}} \approx' [\![Q]\!]_{\mathcal{M}}$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{\star}$, then $P = \star$. Thus, since $P \sim^{\circ}_{\mathtt{IO}} Q$, $Q = \star$ so $[\![Q]\!]_{\mathcal{M}} = \star \xrightarrow{\star}$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{a} M$, then $[\![P]\!]_{\mathcal{M}} = (a.r_P, [\,]) + M'$. Thus $P = (a.r_P) \parallel P'$, $M' = [\![P']\!]_{\mathcal{M}}$ and $M = [\![(r_P, [\odot])]\!]_{\mathcal{M}} + [\![P']\!]_{\mathcal{M}}$.
    Since $P \sim^{\circ}_{\mathtt{IO}} Q$, $Q = (a.r_Q) \parallel Q'$ and $r_P \parallel P' \sim^{\circ}_{\mathtt{IO}} r_Q \parallel Q'$. Thus $[\![Q]\!]_{\mathcal{M}} = (a.r_Q, [\,]) + [\![Q']\!]_{\mathcal{M}} \xrightarrow{a} N = [\![(r_Q, [\odot])]\!]_{\mathcal{M}} + [\![Q']\!]_{\mathcal{M}}$.
    But for all process R, $[\![R]\!]_{\mathcal{M}} = [\![(R, [\odot])]\!]_{\mathcal{M}}$ because the abstraction of $(i, [\,])$ and $(i, [\odot])$ are the same. Thus $M = [\![r_P \parallel P']\!]_{\mathcal{M}}$ and $N = [\![r_Q \parallel Q']\!]_{\mathcal{M}}$.
    By definition of $\approx'$, $[\![r_P \parallel P']\!]_{\mathcal{M}} \approx' [\![r_Q \parallel Q']\!]_{\mathcal{M}}$ so $M \approx' N$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{(\bar{a})} M$, then $[\![P]\!]_{\mathcal{M}} = (\bar{a}\langle s_P \rangle, [\,]) + M'$.
    Thus $P = \bar{a}\langle s_P \rangle \parallel P'$ hence $M = [\![s_P]\!]_{\mathcal{M}}$ and $Q = \bar{a}\langle s_Q \rangle \parallel Q'$ with both $s_P \sim^{\circ}_{\mathtt{IO}} s_Q$ and $P' \sim^{\circ}_{\mathtt{IO}} Q'$. So $[\![Q]\!]_{\mathcal{M}} \xrightarrow{(\bar{a})} N = [\![s_Q]\!]_{\mathcal{M}}$. By definition of $\approx'$, $[\![s_P]\!]_{\mathcal{M}} \approx' [\![s_Q]\!]_{\mathcal{M}}$. Thus $M \approx' N$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{\bar{a}+} M$, then $[\![P]\!]_{\mathcal{M}} = (\bar{a}\langle s_P \rangle, [\,]) + M'$.
    Thus $P = \bar{a}\langle s_P \rangle \parallel P'$ with $M' = [\![P']\!]_{\mathcal{M}}$, hence $M = M' = [\![P']\!]_{\mathcal{M}}$ and $Q = \bar{a}\langle s_Q \rangle \parallel Q'$ with both $s_P \sim^{\circ}_{\mathtt{IO}} s_Q$ and $P' \sim^{\circ}_{\mathtt{IO}} Q'$. So $[\![Q]\!]_{\mathcal{M}} \xrightarrow{\bar{a}+} N = [\![Q']\!]_{\mathcal{M}}$.
    By definition of $\approx'$, $[\![P']\!]_{\mathcal{M}} \approx' [\![Q']\!]_{\mathcal{M}}$. Thus $M \approx' N$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{x} M$, then $[\![P]\!]_{\mathcal{M}} = x + M'$, so $P = v \parallel P'$, with $M' = [\![P']\!]_{\mathcal{M}}$ and either $x = \underline{i}$ and $v = i$, or $v = x$.
    Thus $Q = v \parallel Q'$, with $P' \sim^{\circ}_{\mathtt{IO}} Q'$. Hence, $[\![Q]\!]_{\mathcal{M}} = x + [\![Q']\!]_{\mathcal{M}} \xrightarrow{x} [\![Q']\!]_{\mathcal{M}}$ and by definition of $\approx'$, $[\![P']\!]_{\mathcal{M}} \approx' [\![Q']\!]_{\mathcal{M}}$.
  - If $[\![P]\!]_{\mathcal{M}} \xrightarrow{\tilde{i}} M$, the precedent proof applies by replacing x with i.
- Let $\sim' = \{(P, Q) \mid [\![P]\!]_{\mathcal{M}} \approx_m [\![Q]\!]_{\mathcal{M}},\ P$ and Q well-formed$\}$. We show that $\sim'$ is an IO bisimulation. To do this, let P and Q be so that $P \sim' Q$.

– If $P \xrightarrow{a} P'$, then $P = (a.r_P) \parallel P''$ with $P' = r_P \parallel P''$.
Thus $[\![P]\!]_{\mathcal{M}} = (a.r_P, [\,]) + [\![P'']\!]_{\mathcal{M}} \xrightarrow{a} M_P = [\![(r_P, [\odot])]\!]_{\mathcal{M}} + [\![P'']\!]_{\mathcal{M}} = [\![r_P \parallel P'']\!]_{\mathcal{M}}$.
Since $[\![P]\!]_{\mathcal{M}} \approx_m [\![Q]\!]_{\mathcal{M}}$, $[\![Q]\!]_{\mathcal{M}} \xrightarrow{a} M_Q$ with $M_P \approx_m M_Q$. Thus $[\![Q]\!]_{\mathcal{M}} = (a.r_Q, [\,]) + N$, hence $Q = (a.r_Q) \parallel Q''$ with $N = [\![Q'']\!]_{\mathcal{M}}$. Therefore, $Q \xrightarrow{a} Q' = r_Q \parallel Q''$ and $M_Q = [\![(r_Q, [\odot])]\!]_{\mathcal{M}} + [\![Q'']\!]_{\mathcal{M}} = [\![r_Q \parallel Q'']\!]_{\mathcal{M}}$.
Since $M_P \approx_m M_Q$, $[\![r_P \parallel P'']\!]_{\mathcal{M}} \approx_m [\![r_Q \parallel Q'']\!]_{\mathcal{M}}$, thus, by definition of $\sim'$, $r_P \parallel P' \sim' r_Q \parallel Q'$, that is $P' \sim' Q'$.

– If $P \xrightarrow{\bar{a}\langle s_P \rangle} P'$ then $P = \bar{a}\langle s_P \rangle \parallel P'$. Thus, $[\![P]\!]_{\mathcal{M}} = (\bar{a}\langle s_P \rangle, [\,]) + [\![P']\!]_{\mathcal{M}}$ which admits two transitions: $[\![P]\!]_{\mathcal{M}} \xrightarrow{(\overline{a})} [\![s_P]\!]_{\mathcal{M}}$ and $[\![P]\!]_{\mathcal{M}} \xrightarrow{\overline{a}+} [\![P']\!]_{\mathcal{M}}$.
Since $[\![P]\!]_{\mathcal{M}} \approx_m [\![Q]\!]_{\mathcal{M}}$, $[\![Q]\!]_{\mathcal{M}}$ also admits two transitions flagged with $(\overline{a})$ and $\overline{a}+$, so $[\![Q]\!]_{\mathcal{M}} = (\bar{a}\langle s_Q \rangle, [\,]) + N$. Thus $Q = \bar{a}\langle s_Q \rangle \parallel Q'$ with $N = [\![Q']\!]_{\mathcal{M}}$.
Moreover, $[\![Q]\!]_{\mathcal{M}} \xrightarrow{(\overline{a})} [\![s_Q]\!]_{\mathcal{M}}$ and $[\![Q]\!]_{\mathcal{M}} \xrightarrow{\overline{a}+} [\![Q']\!]_{\mathcal{M}}$ so, since $[\![P]\!]_{\mathcal{M}} \approx_m [\![Q]\!]_{\mathcal{M}}$, we have both $[\![s_P]\!]_{\mathcal{M}} \approx_m [\![s_Q]\!]_{\mathcal{M}}$ and $[\![P']\!]_{\mathcal{M}} \approx_m [\![Q']\!]_{\mathcal{M}}$.
Thus $Q \xrightarrow{\bar{a}\langle s_Q \rangle} Q'$ and by definition of $\sim'$, $s_P \sim' s_Q$ and $P' \sim' Q'$.

– If $P = v \parallel P'$, then $[\![P]\!]_{\mathcal{M}} = x + [\![P']\!]_{\mathcal{M}}$ with, if $v = i$ then $x = \underline{i}$ else $x = v$. Thus $[\![P]\!]_{\mathcal{M}} \xrightarrow{x} [\![P']\!]_{\mathcal{M}}$.
Since $[\![P]\!]_{\mathcal{M}} \approx' [\![Q]\!]_{\mathcal{M}}$, $[\![Q]\!]_{\mathcal{M}} \xrightarrow{x} N$ thus $[\![Q]\!]_{\mathcal{M}} = x + N$, so $Q = v + Q'$ with $N = [\![Q']\!]_{\mathcal{M}}$.
Moreover, $[\![P']\!]_{\mathcal{M}} \approx_m [\![Q']\!]_{\mathcal{M}}$ so, by definition of $\sim'$, $P' \sim Q'$.

10.2 Size of the abstract machine

In order to prove a generalization of the last theorem, we need to define a notion of size for the abstract machine.

Definition 27. For a given process P, we note $\#P$ the number of bound variables within P. The *measure* of an annotated process (P, e), noted $|(P, e)|$ is recursively defined by $|(P, e)| = 1 + \mathrm{size}(P) + \#P \times |e|$ where the *measure* of an environment e, noted $|e|$ is defined as $|[\,]| = 0$, $|\odot :: l| = |l|$ and $|(P, e) :: l| = |(P, e)| + |l|$

The *size* of an abstract machine M, noted $\mathrm{size}\,(M)$ is defined by induction with:

- $\mathrm{size}\,(M + N) = \mathrm{size}\,(M) + \mathrm{size}\,(N)$.
- $\mathrm{size}\,((\bar{a}\langle P \rangle, e)) = |(P, e)|$.
- $\mathrm{size}\,((a.P, e)) = |(P, \odot :: e)|$.
- $\mathrm{size}\,(x) = 1$.

- $\text{size}(\star) = 0$.

Lemma 24. $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) < |(P,e)|$.

Proof of 24. By induction on $\text{size}((P,e))$:

- If $\text{size}((P,e)) = 0$ then $P = \star$ so $\llbracket (P,e) \rrbracket_{\mathcal{M}} = \star$ whose size is 0, whereas $|(P,e)| = 1 + \text{size}(P) + \#P \times |e| \geq 1$.
- Otherwise:
  - If $P = \bar{a}\langle Q \rangle$, then $\llbracket (P,e) \rrbracket_{\mathcal{M}} = (\bar{a}\langle Q \rangle, e)$. Thus, $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = |(Q,e)| = 1 + \text{size}(Q) + \#Q \times |e| = \text{size}(P) + \#P \times |e| = |(P,e)| - 1$.
  - If $P = a.Q$, then $\llbracket (P,e) \rrbracket_{\mathcal{M}} = (a.Q, e)$. Thus $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = |(Q, \odot :: e)| = 1 + \text{size}(Q) + \#Q \times |\odot :: e| = \text{size}(P) + \#P \times |e| = |(P,e)| - 1$.
  - If $P = Q \parallel R$, then $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = \text{size}\left(\llbracket (Q,e) \rrbracket_{\mathcal{M}}\right) + \text{size}\left(\llbracket (R,e) \rrbracket_{\mathcal{M}}\right)$. Thus, by induction, $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) \leq |(Q,e)| - 1 + |(R,e)| - 1 = \text{size}(Q) + \text{size}(R) + (\#Q + \#R)\,|e| = |(P,e)| - 1$.
  - If $P = x$ then $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = \text{size}(x) = 1$ and $|(P,e)| \geq 1 + \text{size}(P) = 2$.
  - If $P = i$, then
    - Either i is not a valid index of e, thus $\llbracket (P,e) \rrbracket_{\mathcal{M}} = \underline{i}$ so $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = 1$ and $|(P,e)| \geq 1 + \text{size}(P) = 2$.
    - Either i is a valid index of e, so $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = \text{size}\left(\llbracket e[i] \rrbracket_{\mathcal{M}}\right)$ whereas $|(P,e)| = 1 + \text{size}(P) + \#P \times |e| = 2 + |e|$. Since i is a valid index of e, $e[i]$ is
      - either one element of the form (Q,f), so $|e| \geq |(Q,f)|$. By induction, $\text{size}\left(\llbracket (Q,f) \rrbracket_{\mathcal{M}}\right) < |(Q,f)|$. But then $|(P,e)| \geq 2 + |(Q,f)| \geq |(Q,f)|$ so finally $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = \text{size}\left(\llbracket (Q,f) \rrbracket_{\mathcal{M}}\right) < |(P,e)|$.
      - either $\odot$, in which case $\text{size}\left(\llbracket (P,e) \rrbracket_{\mathcal{M}}\right) = \text{size}(\underline{i}) = 1$ and $|(P,e)| \geq 1 + \text{size}(P) = 2$.

This notions now allows to build induction using it, because of the following property:

Theorem 25. Let f be a non-terminal flag and M and M' be states so that $M \xrightarrow{f} M'$. Then $\text{size}(M') < \text{size}(M)$.

Proof of 25. By case on f:

- If $f = a$, then $M = (a.r, e) + N \xrightarrow{a} M' = [\![(r, \odot :: e)]\!]_{\mathcal{M}} + N$ but $\text{size}(M') = \text{size}([\![(r, \odot :: e)]\!]_{\mathcal{M}}) + \text{size}(N) < |(r, \odot :: e)| + \text{size}(N) = \text{size}((a.r, e)) + \text{size}(N) = \text{size}(M)$.

- If $f = (\overline{a})$, then $M = (\bar{a}\langle s\rangle, e) + N \xrightarrow{(\overline{a})} M' = [\![(s, e)]\!]_{\mathcal{M}}$ but $\text{size}(M') = \text{size}([\![(s, e)]\!]_{\mathcal{M}}) < |(s, e)| \leq |(s, e)| + \text{size}(N) = \text{size}((\bar{a}\langle s\rangle, e)) + \text{size}(N) = \text{size}(M)$.

- If $f = \bar{a}+$, then $M = (\bar{a}\langle s\rangle, e) + N \xrightarrow{\bar{a}+} M' = N$ but $\text{size}(M') = \text{size}(N) < \text{size}((\bar{a}\langle s\rangle, e)) + \text{size}(N) = \text{size}(M)$.

- If $f = x$ then $M = x + N \xrightarrow{x} M' = N$ but $\text{size}(M') = \text{size}(N) < \text{size}(x) + \text{size}(N) = \text{size}(M)$.

10.3 Stability of $\approx_m$ by $+$

We may now establish the following property, which is quite intuitive:

Theorem 26. Let M_1, M_2, N_1 and N_2 be states so that $M_1 \approx_m N_1$ and $M_2 \approx_m N_2$. Then $M_1 + M_2 \approx_m N_1 + N_2$.

Proof of 26. By induction on $\text{size}(M_1 + M_2)$.

- If $\text{size}(M_1 + M_2) = 0$ then $M_1 = M_2 = N_1 = N_2 = \star$.

- Else, suppose $M_1 + M_2 \xrightarrow{f} M$. By case on f:

  - If $f = a$, then $M_1 + M_2 = (a.r, e) + M' + M_i$. Since both cases are symmetric, we suppose that $M_i = M_2$. Thus $M_1 = (a.r, e) + M'$ so $N_1 = (a.r', e') + N'$ with $[\![(r, \odot :: e)]\!]_{\mathcal{M}} + M' \approx_m [\![(r', \odot :: e')]\!]_{\mathcal{M}} + N'$.

    Thus $M_1 + M_2 \xrightarrow{a} M = [\![(r, \odot :: e)]\!]_{\mathcal{M}} + M' + M_2$. By induction, $M \approx_m [\![(r', \odot :: e')]\!]_{\mathcal{M}} + N' + N_2$, but $N_1 + N_2 \xrightarrow{a} N = [\![(r', \odot :: e')]\!]_{\mathcal{M}} + N' + N_2$, so $M \approx_m N$.
  - If $f = (\overline{a})$, then $M_1 + M_2 = (\bar{a}\langle s\rangle, e) + M' + M_i$. Again, let's take $M_i = M_2$. Thus, $M_1 = (\bar{a}\langle s\rangle, e) + M'$ so $N_1 = (\bar{a}\langle s'\rangle, e') + N'$ with $[\![(s, e)]\!]_{\mathcal{M}} \approx_m [\![(s', e')]\!]_{\mathcal{M}}$.

    Thus $M_1 + M_2 \xrightarrow{(\overline{a})} M = [\![(s, e)]\!]_{\mathcal{M}} + M_2$. By induction, $M \approx_m [\![(s', e')]\!]_{\mathcal{M}} + N_2$, but $N_1 + N_2 \xrightarrow{(\overline{a})} [\![(s', e')]\!]_{\mathcal{M}} + N_2$, so $M \approx_m N$.
  - If $f = \bar{a}+$, then $M_1 + M_2 = (\bar{a}\langle s\rangle, e) + M' + M_i$. Again, let's take $M_i = M_2$. Thus, $M_1 = (\bar{a}\langle s\rangle, e) + M'$ so $N_1 = (\bar{a}\langle s'\rangle, e') + N'$ with $M' \approx_m N'$.

    Thus $M_1 + M_2 \xrightarrow{\bar{a}+} M = M' + M_2$. By induction, $M \approx_m N' + N_2$, but $N_1 + N_2 \xrightarrow{\bar{a}+} N' + N_2$, so $M \approx_m N$.

  - If $f = x$, then $M_1 + M_2 = x + M' + M_i$. Again, let's take $M_i = M_2$. Thus, $M_1 = x + M'$ so $N_1 = x + N'$ with $M' \approx_m N'$.
    Thus $M_1 + M_2 \xrightarrow{x} M = M' + M_2$. By induction, $M \approx_m N' + N_2$, but $N_1 + N_2 \xrightarrow{x} N' + N_2$, so $M \approx_m N$.

10.4 Strong equivalence

The weak equivalence is actually a restriction of a more general result, which completes it with all the remaining machine states, in which environments may not be empty (or filled with $\odot$):

Theorem 27. Let M_1 and M_2 be well-formed states. Then

$$M_1 \approx_m M_2 \Leftrightarrow [\![M_1]\!]_{\mathcal{P}} \sim^{\circ}_{\mathtt{IO}} [\![M_2]\!]_{\mathcal{P}}$$

We prove it using the following lemma:

Lemma 28. $M \approx_m \widehat{M}$.

Proof of 28. By induction on $\operatorname{size}(M)$:

- If $\operatorname{size}(M) = 0$, then $M = \star$, so $\widehat{M} = \star$, thus $M \approx_m \widehat{M}$.
- Else, let's first prove that $\widehat{M}$ simulates M:
  - If $M \xrightarrow{a} M'$, then $M = (a.r, e) + M'' \xrightarrow{a} [\![(r, \odot :: e)]\!]_{\mathcal{M}} + M''$, so $\widehat{M} = \left(a.\left([\![(r,e)]\!]^1_{\mathcal{P}}\right), [\,]\right) + \widehat{M''} \xrightarrow{a} \left[\!\!\left[\left([\![(r,e)]\!]^1_{\mathcal{P}}, [\odot]\right)\right]\!\!\right]_{\mathcal{M}} + \widehat{M''} = \left[\!\!\left[[\![(r,e)]\!]^1_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}} + \widehat{M''}$.
    First, note that $[\![\widehat{(r, \odot :: e)}]\!]_{\mathcal{M}} = \left[\!\!\left[[\![(r,e)]\!]^1_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}}$ because of lemma 22.
    Thus, by induction, $\left[\!\!\left[[\![(r,e)]\!]^1_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}} \approx_m [\![(r, \odot :: e)]\!]_{\mathcal{M}}$. By induction, we also have $M'' \approx_m \widehat{M''}$.
    Hence, by stability of $\approx_m$ by $+$, $M' \approx_m \left[\!\!\left[[\![(r,e)]\!]^1_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}} + \widehat{M''}$.
  - If $M \xrightarrow{(\bar{a})} M'$, then $M = (\bar{a}\langle s\rangle, e) + M'' \xrightarrow{(\bar{a})} [\![(s,e)]\!]_{\mathcal{M}}$, so $\widehat{M} = \left(\bar{a}\left\langle [\![(s,e)]\!]^0_{\mathcal{P}}\right\rangle, [\,]\right) + \widehat{M''} \xrightarrow{(\bar{a})} \left[\!\!\left[\left([\![(s,e)]\!]^0_{\mathcal{P}}, [\,]\right)\right]\!\!\right]_{\mathcal{M}} = \left[\!\!\left[[\![(s,e)]\!]^0_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}}$.
    But $[\![\widehat{(s,e)}]\!]_{\mathcal{M}} = \left[\!\!\left[[\![(s,e)]\!]^0_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}}$ because of lemma 22, so, by induction, $[\![(s,e)]\!]_{\mathcal{M}} \approx_m \left[\!\!\left[[\![(s,e)]\!]^0_{\mathcal{P}}\right]\!\!\right]_{\mathcal{M}}$.

  - If $M \xrightarrow{\bar{a}+} M'$, then $M = (\bar{a}\langle s\rangle, e) + M'' \xrightarrow{\bar{a}+} M''$, so $\widehat{M} = \left(\bar{a}\left\langle [\![(s,e)]\!]^0_{\mathcal{P}}\right\rangle, [\,]\right) + \widehat{M''} \xrightarrow{\bar{a}+} \widehat{M''}$.
    Thus, by induction, $M'' \approx_m \widehat{M''}$.
  - If $M \xrightarrow{x} M'$, then $M = x + M'' \xrightarrow{x} M''$, so $\widehat{M} = x + \widehat{M''} \xrightarrow{x} \widehat{M''}$.
    By induction, $M'' \approx_m \widehat{M''}$.

- The proof that M simulates $\widehat{M}$ uses the exact same arguments, since the general structure of M can be deduced from that of $\widehat{M}$.

We may finally conclude the proof of the strong equivalence.

Proof of 27.

$$\begin{aligned} M_1 \approx_m M_2 &\Leftrightarrow \widehat{M_1} \approx_m \widehat{M_2} \text{ as we just saw} \\ &\Leftrightarrow [\![[\![M_1]\!]_{\mathcal{P}}]\!]_{\mathcal{M}} \approx_m [\![[\![M_2]\!]_{\mathcal{P}}]\!]_{\mathcal{M}} \text{ by definition of } \widehat{M} \\ &\Leftrightarrow [\![M_1]\!]_{\mathcal{P}} \sim^{\circ}_{\mathtt{IO}} [\![M_2]\!]_{\mathcal{P}} \end{aligned}$$

because of the weak equivalence of bisimilarities.

References

[1] Malgorzata Biernacka, Dariusz Biernacki, Sergueï Lenglet, Piotr Polesiuk, Damien Pous, and Alan Schmitt. Fully Abstract Encodings of λ-Calculus in HOcore through Abstract Machines. In *LICS 2017*, Proceedings of LICS 2017, Reykjavik, Iceland, June 2017. To appear.

[2] Arthur Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, 49(3):363–408, October 2012.

[3] Martín Escarrá, Petar Maksimović, and Alan Schmitt. HOCore in Coq. In *Vingt-sixièmes Journées Francophones des Langages Applicatifs (JFLA 2015)*, Le Val d'Ajol, France, January 2015.

[4] Ivan Lanese, Jorge A. Peréz, Davide Sangiorgi, and Alan Schmitt. On the Expressiveness and Decidability of Higher-Order Process Calculi. In *23rd Annual IEEE Symposium on Logic in Computer Science (LICS 2008)*, Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS 2008), pages 145–155, Pittsburgh, Pennsylvania, United States, June 2008.