Une traduction de PLOTOS en MEIJE

Guillaume Doumenc, Eric Madelaine

INRIA

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 938

Programme 1

UNE TRADUCTION DE PLOTOS EN MEIJE

Guillaume DOUMENC
Eric MADELAINE

Décembre 1988

Une traduction de PLOTOS en MEIJE

A translation from PLOTOS to MEIJE

Guillaume Doumenc [1]
Eric Madelaine

I.N.R.I.A. Sophia Antipolis
Route des Lucioles
06 560 Valbonne FRANCE
madelain@mirsa.inria.fr

Résumé. Nous définissons la notion de validité sémantique d'une traduction simple d'algèbres de processus et montrons comment nous pouvons valider une traduction de PLOTOS, le noyau de synchronisation pure de LOTOS, en MEIJE. Cette validation sera faite automatiquement au moyen de l'outil ECRINS, qui est un laboratoire de preuves pour les algèbres de processus.
Mots clés : Traduction, algèbre de processus, équivalences par bisimulation, LOTOS, validité de transformations.

Abstract. We define semantical validation of a simple process algebra translation, and then validate a translation from PLOTOS, the pure synchronisation kernel of LOTOS, into MEIJE, using the ECRINS proof laboratory for process calculi.
Keywords : Translation, process calculi, bisimulation equivalences, LOTOS, valid transformations.

[1] G. Doumenc a contribué au présent rapport dans le cadre d'une thèse de doctorat de 3^{eme} cycle en cours de préparation au Centre Etudes et Recherches de IBM à la Gaude.

1 Introduction

Le but de cet article est de donner un exemple de traduction automatiquement validée entre calculs de processus, et plus précisement de PLOTOS, le noyau de synchronisation pur de LOTOS ([5], [3]), en MEIJE ([1], [2]). Nous validons cette traduction par l'outil ECRINS ([7], [6]). ECRINS est un environnement qui permet de définir des algèbres de processus et de faire des preuves de bisimulations [8] sur des termes de ces algèbres; on supposera par la suite, que le lecteur connait l'outil ECRINS. Nous nous limiterons aux calculs de processus dont la sémantique opérationnelle est définie sous forme de systèmes de transitions ([9]) et nous noterons $\sim$ l'équivalence par bisimulation.

2 Traducteur : PLOTOS $\mapsto$ MEIJE

Les définitions que nous allons être amenées à poser, proviennent d'un travail en cours sur la notion de validité sémantique d'une traduction entre deux algèbres de processus via un critère d'observation ([4], [11]). Elles sont adaptées ici au cas particulier où les algèbres d'actions sont identiques. Nous n'insisterons pas ici sur la généralité de ces définitions, le but étant de traiter un seul exemple significatif.

2.1 MEIJE et PLOTOS

Rappelons brièvement les algèbres d'actions de MEIJE et de PLOTOS :

- MEIJE : L'algèbre des actions est le monoide commutatif engendré par un ensemble d'atomes et de signaux inversibles.
- PLOTOS : Une action est représentée par un atome dont le nom est celui de la porte sur laquelle à lieu la communication.

L'algèbre des actions de PLOTOS est donc un sous-ensemble strict de celle de MEIJE (où il n'y a ni signaux, ni produit d'actions).

2.2 Définition d'un traducteur

Soient deux calculs de processus $\mathcal{F}$, $\mathcal{G}$ et leurs domaines sémantiques respectifs $[\mathcal{F}]_1$, $[\mathcal{G}]_2$. Nous appelerons *traducteurs d'algèbres de processus*, tout couple $(\mathcal{T}, \pi)$, avec $\mathcal{T}$ un homomorphisme de $\mathcal{F}$ vers $\mathcal{G}$, et π une bijection reliant les deux espaces sémantiques et respectant certaines conditions de compositionalité par rapport à la sémantique opérationnelle.

Dans le présent rapport, nous nous limiterons au cas où π est l'identité, ou plus exactement une forme d'inclusion d'une structure de règles s'appuyant sur l'algèbre des actions de PLOTOS dans celle des actions de MEIJE dont elle est

un sous-ensemble strict. Toutefois pour les besoins de la traduction, on pourra naturellement manipuler de manière interne des expressions utilisant des actions plus complexes de l'algèbre cible, sous réserve que les comportements apparents restent bien dans la restriction aux actions PLOTOS.

La validité d'une traduction $(\mathcal{T}, \pi)$, s'enonce simplement :

$$\forall t \text{ (terme clos) } \in \mathcal{F},\ \pi([\![\mathcal{T}(t)]\!]_2) \sim [\![t]\!]_1$$

$\mathcal{F}$ est alors appelée algèbre *source* et $\mathcal{G}$ algèbre *cible*.

Dans le cas où les domaines sémantiques des deux algèbres sont identiques, la projection π est réduite à l'identité; la notion de validité d'une traduction est réduite à l'équivalence de la sémantique de tout terme et de sa traduction. Ainsi, pour une sémantique opérationnelle *à la Plotkin* définie sur l'algèbre PLOTOS et sur l'algèbre MEIJE, et en considérant le même ensemble d'actions (l'ensemble des atomes, noms de porte), la validité d'une traduction se traduit par :

$$(1)\ \forall t \text{ (terme clos) } \in \text{ PLOTOS },\ [\![\mathcal{T}(t)]\!] \sim [\![t]\!]$$

La structure d'algèbre permet de définir inductivement tout traducteur par la donnée de la traduction de chaque opérateur. La bisimulation par un critère fort étant une congruence sur l'ensemble des opérateurs, tout homomorphisme d'algèbres tel que la traduction de chaque opérateur de l'algèbre source est en bisimulation forte avec ce même opérateur, sera un traducteur.

C'est-à-dire, en notant :

- Θ l'ensemble des opérateurs de $\mathcal{F}$ et $\theta_I \in \Theta$ d'arité I,
- $\vec{p_i}$ le vecteur de variables de processus de l'opérateur θ_I,
- $\underset{\text{FH}}{\cong}$, définie par R. de Simone [10], l'extension de la bisimulation forte aux termes non clos. Cette équivalence, que nous appellerons FH-bisimulation (*Formal Hypothesis*), utilise un ensemble de règles de comportement pour représenter l'ensemble des comportements de toutes les instantiations possibles des variables de processus,
- $\mathcal{T}(\vec{p_i})$ la traduction des variables de processus. Pour les besoins de la FH-bisimulation, celles-ci doivent se comporter comme des variables de processus de l'algèbre $\mathcal{F}$, c'est pourquoi nous les noterons $\vec{p_i}$ par abus de notation.

alors la traduction sera supposée valide si :

$$(2)\ \forall \theta_I \in \Theta, \forall p_{i \in I},\ [\![\theta_I(\vec{p_i})]\!] \underset{\text{FH}}{\cong} [\![\mathcal{T}(\theta_I)(\vec{p_i})]\!]$$

Pour la traduction étudiée, la relation de FH-bisimulation, pourra être prouvée en mode CEA d'ECRINS (*composed enumeration array*), car il n'existe pas de produit d'actions en PLOTOS. En ECRINS, l'utilisation de ce mode remplace les prédicats formels des règles par leur interprétation dans un modèle fini ne comportant pas de produits d'actions.

2.3 Méthode de traduction

Avant de présenter la structure générale de la traduction, nous rappelons un opérateur de l'outil ECRINS :

$$\textbf{local signals } a_1, \cdots, a_n \textbf{ in } process$$

Cet opérateur joue un rôle très important dans la traduction. En effet, nous avons besoin de définir des actions dont la visibilité est strictement locale au schéma de traduction. Cela implique deux contraintes pour l'évaluateur :

1. il faut renommer les actions locales, en fonction de l'environnement afin d'éviter tout conflit de noms de variables. Cela revient à dire que l'on se donne le moyen de générer de nouveaux noms de variables, différents de ceux préalablement définis (similaire à la notion d'α-conversion),

2. en particulier, les variables de processus ne peuvent réaliser une action locale. En effet, pour prouver la bisimulation nous devons faire des hypothèses sur les actions que peuvent réaliser les variables des processus; nous imposerons que les actions locales ne fassent pas partie de la base de l'expression, c'est-à-dire des actions réalisables par les variables libres de processus.

Trois opérateurs MEIJE vont nous être particulièrement utiles pour cette traduction : le pilotage associé à la restriction et le renommage.

1. Le pilotage couplé à la restriction, nous permettera de bloquer un processus, ou de controler son exécution. Pour cela, il suffit de piloter le processus p par le signal s que l'on restreint, signal dont on n'émettra l'inverse que lorsque l'on voudra permettre au processus p de s'exécuter.

2. Le renommage nous permettera de savoir si une action se réalise ou non. Pour cela, il suffit de renommer une action par un produit d'action là contenant ainsi qu'un signal de trace. Ainsi, si l'on véut savoir si une action a est réalisée par le processus p, on renomme a en $a.s$, où s est un nouveau signal, et on surveille le signal s pour *observer* l'action a.

La traduction de chaque opérateur de *PLOTOS* se fera alors suivant le schéma :

- On pilote ou observe une action particulière de chaque variable de processus intervenant dans la sémantique de cet opérateur,

- On associe en parallèle une horloge qui synchronise les comportements des processus, soit :

  - en émettant l'inverse des signaux de pilotage pour permettre à un processus de s'exécuter normalement,

- en bloquant un processus en n'émettant pas l'inverse d'un signal de pilotage.
- en attendant des signaux pour se transformer en une autre horloge et ainsi modifier le comportement de l'expression resultante.

2.4 Définition du calcul M+P

Dans la version actuelle d'ECRINS, il n'est pas possible de considérer deux calculs en même temps, c'est pourquoi on est obligé de définir un nouveau calcul $M + P$ comprenant l'union des opérateurs de MEIJE et de PLOTOS. Ainsi les expressions et leurs traductions appartiendront au même calcul, ce qui nous permettra de tester leur bisimulation. Pour des raisons de conflits lexicaux et en l'absence de possibilité de surcharge, la syntaxe du calcul $M+P$ peut paraître originale; toutefois les opérateurs restent syntaxiquement les mêmes, aux seules notables exceptions :

- le parallèle de MEIJE (symbole utilisé pour le rendez-vous en PLOTOS) qui est décrit par le symbole ///,
- le rendez-vous PLOTOS qui utilise l'opérateur rendez-vous (dont la syntaxe est prédéfinie en ECRINS) noté //{ ensemble de portes }//.

La définition du calcul $M + P$ est, dans la syntaxe du formalisme ECRINS :

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%                        MEIJE + PLOTOS CALCULUS                               %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


empty i                          % the internal action
constant atom d                  % the terminaison action


%-----------------   MEIJE and derived operators


operator Zero::  Process
syntax
   natural 0
end


operator Prefixing::  Action  Process --> Process
syntax
   :  right 3
semantics
   prefixing
```

```
        -------------
        a:p -- a --> p
end

operator Parallel:: Process Process --> Process
syntax
   /// left 6
semantics
   parallel_left
          p -- a -->p'
        --------------------
        p///q -- a --> p'///q

   parallel_right
           q -- b -->q'
        --------------------
        p///q -- b --> p///q'

   parallel_all
      p -- a -->p' & q -- b -->q'
      -------------------------
       p///q -- a.b --> p'///q'
end

operator Restriction:: Process Label --> Process
syntax
   \ left 4
semantics
   restriction
      p -- a --> p' & not ( s divide a)
      ---------------------------------
           p\s -- a --> p'\s
end

operator Ticking :: Action Process --> Process
syntax
   * right 3
semantics
    pilot
          p -- a -->p'
       -----------------
       s*p -- s.a -->s*p'
end
```

```
operator Trigger ::  Action  Process --> Process
syntax
    => right 3
semantics
    trigger
          p -- a -->p'
       -----------------
       s=>p -- s.a -->p'
end

operator P-renaming ::  builtin
semantics
   renaming
        p -- a -->p'
       ------------------------
p [R] -- a<R> --> p'[R]
end

operator Sum::  Process  Process --> Process
syntax
   + left 4
semantics
   sum_left
         p -- a --> p'
       -----------------
       p + q -- a --> p'

    sum_right
         q -- b --> q'
       -----------------
       p + q -- b --> q'
end


%----------------------  PLOTOS


operator Stop::  Process
syntax
   ident stop
fin

operator Exit::  Process
```

```
syntax
   ident exit
semantics
   exit_rule
       ------------------
       exit -- d --> stop
fin

operator Hiding:: Process Label --> Process
syntax
   \\ left 8
semantics
   hiding_i
       p -- a --> p' & ((a equal s) and (not(a equal i)))
       ---------------------------------------------------
                      p\\s -- i --> p'\\s

   hiding
             p -- a --> p' & not (a equal s)
             -------------------------------
                    p\\s -- a --> p'\\s
fin

operator rdv:: builtin
semantics
   rdv_gauche
      p -- a -->p' & (not(a equal g))
      -------------------------------
      p// {g} //q -- a --> p'// {g} //q

   rdv_droite
      q -- b -->q' & (not(b equal g))
      -------------------------------
      p// {g} //q -- b --> p// {g} //q'

   rdv_tous
 p -- a -->p' & q -- b -->q' & (a equal g)and(b equal g)
 ------------------------------------------------------
             p// {g} //q -- a --> p'// {g} //q'
fin

operator Disable:: Process Process --> Process
syntax
   [> left 4
```

```
semantics
   disabling_1
      p -- a --> p'  &  not (a equal d)
      ---------------------------------
            p[>q -- a --> p'[>q

   disabling_2
        p -- a --> p'  &  (a equal d)
        ----------------------------
             p[>q -- d --> p'

   disabling_3
              q -- a --> q'
              --------------
              p[>q -- a --> q'
fin

operator Enable::  Process  Process --> Process
syntax
   >> left 2
semantics
   enabling_1
      p -- a --> p'  &  not (a equal d)
      ---------------------------------
            p>>q -- a --> p'>>q

   enabling_2
      p -- a --> p'  &  (a equal d)
      -----------------------------
           p>>q -- i --> q
fin

operator Sequence::  Process  Process --> Process
syntax
   :> left 2
semantics
   sequence_1
       p -- a --> p'  &  not (a equal d)
       ---------------------------------
             p:>q -- a --> p':>q

   sequence_2
    p -- a --> p'  &  q -- b --> q'  &  (a equal d)
    -----------------------------------------------
```

```
                p:>q -- b --> q'
fin

operator Choice ::  Process Process --> Process
syntax
   [] right 3
semantics
   choice_left
         p -- a --> p'
        ------------------
        p [] q -- a --> p'

    choice_right
         q -- b --> q'
        ------------------
        p [] q -- b --> q'
end
```

2.5 Définition du traducteur

Par simplification notationnelle, nous appellerons action locale, toute action locale et restreinte. La traduction des opérateurs PLOTOS en MEIJE est alors :

1. Stop : $\mathcal{T}(\textbf{stop}) = 0$,

   la traduction est immédiate car les deux opérateurs n'ont pas de règles sémantiques,

2. Exit : $\mathcal{T}(\textbf{exit}) = d : 0$,

   l'atome d représente le signal de terminaison PLOTOS,

3. Hiding : $\mathcal{T}(\textbf{hide } g \textbf{ in } p) = p[i/g]$,

   pour cacher une action, il suffit de la renommer en une action invisible

4. Rendez-vous : $\mathcal{T}(p|\ [\{g\}]\ |q) =$ local signals $\{a,\ b\}$ in
   $\{(a * p[b.g/g]//a * q[b/g]//$
   let rec $\{h = a^- : h + a^{-2}.b^{-2} : h\}$ in $h)\backslash a\backslash b\backslash c$,

   le principe est de piloter les deux processus par un signal local a que l'horloge n'émet qu'une fois par cycle, ce qui garantie l'exclusion mutuelle des processus. Pour l'action commune g, on la renomme avec un signal de trace b, l'horloge n'emettant b^- que par couples ce qui oblige les deux processus à travailler ensemble sur g.

5. Enabling : $\mathcal{T}(p¿¿q) =$ local signals $\{a,\ b,\ c\}$ in
   $\{(a * p[c/d]//b \Rightarrow q//$ let rec $\{h = a^- : h + a^-.c^- : b^- : 0\}$ in $h)$
   $\backslash a\backslash b\backslash c\}$,

   le principe est de garder le processus q par un signal b qui ne sera émit par l'horloge que si le processus p termine, ce que l'on observe par le signal c. Enfin, on bloque le processus p par le signal a qui ne sera plus émit par l'horloge apres l'action d,

6. Disabling : $\mathcal{T}(p|{>}q) =$ local signals $\{a,\ b,\ c\}$ in
   $\{(a * p[c.d/d]//b \Rightarrow q//$ let rec $\{h = a^- : h + a^-.c^- : h\}$ and
   $\{l = b^- : 0 + a^- : l + a^-.c^- : h\}$ in $l)\backslash a\backslash b\backslash c$,

   il faut distinguer deux comportements : celui où p termine, et celui où q peut faire une action. Ces deux comportements sont observés réciproquement par les signaux b et c :

   (a) p termine, l'horloge se trouve alors dans un état où elle permet à p de faire n'importe quelle action mais bloque q en m'émettant jamais le signal b.

   (b) q peut faire une action, elle se fera en même temps qu'une émission de b, l'horloge se bloque alors bloquant p qui est piloté par le sigal a.

7. Sequence : $\mathcal{T}(p;¿q$ = local signals $\{a,\ b,\ c\}$ in
   $(a * \mathcal{T}(p)[c/d]//b \Rightarrow \mathcal{T}(q)//$ let rec $h = a^- : h + a^-.b^-.c^- : 0$ in $h)\backslash a\backslash b\backslash c$,
   le principe est de bloquer le processus q par le signal b, signal qui ne sera émit que si p termine. Dans ce cas l'horloge s'arrête bloquant ainsi le processus p en n'émettant plus de signal a.

8. Choice : $\mathcal{T}(p[]q) = p + q$
   la traduction est immédiate car les deux opérateurs ont les mêmes règles de sémantiques,

2.6 Test de bisimulation, validation

```
**********************************************************
*                        ECRINS                          *
*             Version 1.8  (October 1988)                *
*                                                        *
*             Auditing on file:  trad.aud                 *
*          Generated on:  fri oct  28 88 11:40:09         *
**********************************************************

Current Toplevel Flags:

    debug-base = False          debug-tactic = 0
    debug-compile = 1           debug-top = False
    debug-equiv = 1             debug-verbose = False
    debug-eval = 0              echo = False
    debug-failure = False       load-stop = True
    debug-gc = False            print-brackets = False
    debug-locals = False        print-env = 0
    debug-simpl = True          timer = True

**********************************************************

Auditing on trad.aud

time = 0.82s
@ load "trad";
@
@
@ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@ %
@ %   Translation of P(ure)LOTOS operators in MEIJE
@ %
@ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
@
@
@ % load the calculus Meije and Plotos
@ calculus "m+p";

Operators:
  Zero Prefixing Parallel Restriction Ticking Trigger
  P-renaming Sum Stop Exit Hiding rdv Disable Enable
  Sequence Choice
No Simplifiers.

time = 19.78s
@
@ % we put the environment
@ set timer = true;
timer :  Bool

time = 0.52s
@ set debug-locals=false;
debug-locals :  Bool

time = 0.50s
@
@ % load the predicate-mode "cea" mode
@ predicate-mode "cea";
You are in cea mode.

time = 0.48s
@ global-constraint signal none;
Global Constraint:
  <none | any>

time = 0.50s
@
@ %
@ % We will prove the bisimulation between terms with the
@ % process variable sorts containing no signals, because
@ % an PLOTOS action is mapped into an MEIJE atom.
@ %
@
@ % load the translation of the operators
@ load "stop";
@
@ % Stop
```

```
@
@
@ parse
@ t_stop = 0;
t_stop :  Process of m+p

time = 0.40s
@ show eval t_stop;
Specif of:  0
   No behaviour
  :  Specif of m+p

time = 0.48s
@ parse r = {stop == t_stop};
r :  Relation of m+p

time = 0.64s
@ show prove r;
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  stop
   No behaviour

Evaluation of right-hand-side gives :
Specif of:  t_stop
   No behaviour

-------------------------------------------------
     Proof Succeeds
CEA |-- Isbisim {"stop == t_stop"} :  Theorem of m+p

time = 0.92s {+ display= 0.24s}
@
@ file stop.ec loaded.

time = 0.54s
@ load "exit";
@
@ % Exit
@
@
@ parse
@ t_exit = d :  0;
```

```
t_exit :  Process of m+p

time = 0.44s
@ show eval t_exit;
Specif of:  d:0
   {#<Cea res |d>} --res--> 0
  :  Specif of m+p

time = 0.48s {+ display= 0.28s}
@ parse r = {t_exit == exit, stop == 0 };
r :  Relation of m+p

time = 0.96s
@ show prove r;
-------------------------------------------------
 Processing Equation:  "t_exit == exit"
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_exit
   {#<Cea res |d>} --res--> 0

Evaluation of right-hand-side gives :
Specif of:  exit
   {#<Cea res |d>} --res--> stop

-------------------------------------------------
     Proof Succeeds
-------------------------------------------------
 Processing Equation:  "stop == 0"
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  stop
   No behaviour

Evaluation of right-hand-side gives :
Specif of:  0
   No behaviour

-------------------------------------------------
     Proof Succeeds
CEA |-- Isbisim {"t_exit == exit", "stop == 0"}
 :  Theorem of m+p
```

```
time = 1.84s {+ display= 0.34s}
@
@ file exit.ec loaded.

time = 0.56s
@ load "hiding";
@
@
@ % Hiding
@
@
@ parse
@ t_hiding = p[i / a];
t_hiding :  Process of *

time = 0.48s
@ show eval t_hiding;
Specif of:  p[i/a]
   {p}{#<Cea u_p |i a ext_1
             res |i i ext_1>}
        --res-->
        x_p[i/a]
  :  Specif of *

time = 0.60s {+ display= 0.92s}
@ parse r = {t_hiding == p \\ a};
r :  Relation of m+p

time = 0.88s
@ show prove r;
--------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_hiding
   {p}{#<Cea u_p |i a ext_1
             res |i i ext_1>}
        --res-->
        x_p[i/a]

Evaluation of right-hand-side gives :
Specif of:  p\\a
   {p}{#<Cea u_p |a
             res |i>}
```

```
        --res-->
        x_p\\a
   {p}{#<Cea u_p |i ext_1
            res |i ext_1>}
        --res-->
        x_p\\a

-------------------------------------------------
     Proof Succeeds
CEA |-- Isbisim {"t_hiding == (p\\a)"}
 :  Theorem of m+p

time = 3.14s {+ display= 0.30s}
@
@ file hiding.ec loaded.

time = 0.58s
@ load "rdv";
@
@ % Rendez-vous
@
@
@ parse
@ rendez-vous = local signals {a, b} in
m+p> (a!  * p[ b!.g / g ] /// a!  * q[ b!  / g ] ///
m+p> let rec h = a?:h + a??.b??:h in h)
m+p> \a!  \b!;
rendez-vous :  Process of m+p

time = 1.62s
@ show set s = eval rendez-vous;
s =
  Specif of:
    local signals {a , b} in
      (((a!*p[b!.g/g] ///a!*q[b!/g] )///
        let rec {h = a?:h+a??.b??:h} in h)\a!)
        \b!
     {p}{#<Cea u_p |i ext_1 ext_2
              res |i ext_1 ext_2>}
          --res-->
          local signals {a , b} in
            (((a!*x_p[b!.g/g] ///a!*q[b!/g] )///
                let rec {h = a?:h+a??.b??:h} in h)
              \a!)
```

```
            \b!
      {q}{#<Cea u_q |i ext_1 ext_2
                res |i ext_1 ext_2>}
           --res-->
           local signals {a , b} in
             (((a!*p[b!.g/g] ///a!*x_q[b!/g] )///
                 let rec {h = a?:h+a??.b??:h} in h)
               \a!)
              \b!
      {q, p}{#<Cea u_p |g
                   u_q |g
                   res |g>}
             --res-->
             local signals {a , b} in
               (((a!*x_p[b!.g/g] ///a!*x_q[b!/g] )///
                   let rec {h = a?:h+a??.b??:h} in h)
                 \a!)
                \b!
  :  Specif of m+p

time = 10.44s {+ display= 6.56s}
@ parse r = { rendez-vous == p // {g} // q};
r :  Relation of *

time = 0.98s
@ show prove r;
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  rendez-vous
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        local signals {a , b} in
          (((a!*x_p[b!.g/g] ///a!*q[b!/g] )///
              let rec {h = a?:h+a??.b??:h} in h)
            \a!)
           \b!
   {q}{#<Cea u_q |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        local signals {a , b} in
          (((a!*p[b!.g/g] ///a!*x_q[b!/g] )///
              let rec {h = a?:h+a??.b??:h} in h)
```

```
            \a!)
           \b!
   {q, p}{#<Cea u_p |g
               u_q |g
               res |g>}
          --res-->
          local signals {a , b} in
            (((a!*x_p[b!.g/g] ///a!*x_q[b!/g] )///
                let rec {h = a?:h+a??.b??:h} in h)
               \a!)
              \b!

Evaluation of right-hand-side gives :
Specif of:  p//{g}//q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p//{g}//q
   {q}{#<Cea u_q |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        p//{g}//x_q
   {q, p}{#<Cea u_p |g
               u_q |g
               res |g>}
          --res-->
          x_p//{g}//x_q

------------------------------------------------
     Proof Succeeds
CEA |-- Isbisim {"rendez-vous == (p//{g}//q)"}
 :  Theorem of m+p

time = 29.24s {+ display= 0.36s}
gc= 1
@ file rdv.ec loaded.

time = 0.56s
@ load "enabling";
@
@ % Enabling
@
@
@ parse
```

```
@ t_enabling = local signals {a, b, c} in
m+p> (a!  * p[ c!  / d ] /// b!  => q ///
m+p> let rec h = a?:h + a?.c?:b?:0 in  h)
m+p> \a!  \b!  \c!;
t_enabling :  Process of m+p

time = 1.60s
@ show set s = eval t_enabling;
s =
  Specif of:
    local signals {a , b , c} in
      ((((a!*p[c!/d] ///b!=>q)///
         let rec {h = a?:h+a?.c?:(b?:0)} in h)\a!)\b!)
        \c!
     {p}{#<Cea u_p |i ext_1 ext_2
              res |i ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!/d] ///b!=>q)///
                 let rec {h = a?:h+a?.c?:(b?:0)} in h)
                \a!)
               \b!)
              \c!
     {p}{#<Cea u_p |d
              res |i>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!/d] ///b!=>q)///b?:0)\a!)\b!)\c!
  :  Specif of m+p

time = 10.58s {+ display= 4.16s}
@ show set t_enabling2 = nextn(2, s);
t_enabling2 =
  local signals {a , b , c} in
     ((((a!*x_p[c!/d] ///b!=>q)///b?:0)\a!)\b!)\c!
  :  Process of m+p

time = 0.98s {+ display= 0.86s}
@ show set t_enabling3 = nextn(1, eval t_enabling2);
t_enabling3 =
  local signals {a , b , c} in
     ((((a!*x_p[c!/d] ///x_q)///0)\a!)\b!)\c!
  :  Process of m+p
```

```
time = 9.44s {+ display= 0.76s}
@ parse r = {t_enabling == p >> q,
m+p>            t_enabling2 == q,
m+p>            t_enabling3 == x_q};
r :  Relation of m+p

time = 0.86s
@ show prove r;
-------------------------------------------------
 Processing Equation:  "t_enabling == p>>q"
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_enabling
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        local signals {a , b , c} in
          ((((a!*x_p[c!/d] ///b!=>q)///
             let rec {h = a?:h+a?.c?:(b?:0)} in h)
              \a!)
             \b!)
            \c!
   {p}{#<Cea u_p |d
             res |i>}
        --res-->
        local signals {a , b , c} in
          ((((a!*x_p[c!/d] ///b!=>q)///b?:0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  p>>q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p>>q
   {p}{#<Cea u_p |d
             res |i>}
        --res-->
        q

-------------------------------------------------
     Proof Succeeds
-------------------------------------------------
 Processing Equation:  "t_enabling2 == q"
```

```
------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_enabling2
   {q}{#<Cea u_q |i d ext_1 ext_2
             res |i d ext_1 ext_2>}
         --res-->
         local signals {a , b , c} in
          ((((a!*x_p[c!/d] ///x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  q
   {q}{#<Cea u_q |i d ext_1 ext_2
             res |i d ext_1 ext_2>}
         --res-->
         x_q

------------------------------------------------
     Proof Succeeds
------------------------------------------------
 Processing Equation:  "t_enabling3 == x_q"
------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_enabling3
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                res |i d ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!/d] ///x_x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  x_q
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                res |i d ext_1 ext_2>}
          --res-->
          x_x_q

------------------------------------------------
     Proof Succeeds
CEA |-- Isbisim {"t_enabling == p>>q", "t_enabling2 == q"}
  :  Theorem of m+p

time = 30.64s {+ display= 0.44s}
```

```
@
@ file enabling.ec loaded.

time = 0.60s
@ load "disabling";
@
@ % Disabling
@
@
@ parse
@ t_disabling = local signals {a, b, c} in
m+p> (a!  * p[ c!.d / d ] /// b!  => q ///
m+p> let rec {l = b?:0 + a?:l + a?.c?:h and
m+p> h = a?:h + a?.c?:h} in l)
m+p> \a!  \b!  \c!;
t_disabling :  Process of m+p

time = 2.04s
@
@ show set s = eval t_disabling;
s =
  Specif of:
    local signals {a , b , c} in
      ((((a!*p[c!.d/d] ///b!=>q)///
           let rec {l = (b?:0+a?:l)+a?.c?:h
                 and h = a?:h+a?.c?:h} in l)
         \a!)
        \b!)
       \c!
     {p}{#<Cea u_p |i ext_1 ext_2
              res |i ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!.d/d] ///b!=>q)///
                 let rec {l = (b?:0+a?:l)+a?.c?:h
                       and h = a?:h+a?.c?:h} in
                   l)
                \a!)
               \b!)
              \c!
     {p}{#<Cea u_p |d
              res |d>}
          --res-->
          local signals {a , b , c} in
```

```
            ((((a!*x_p[c!.d/d] ///b!=>q)///
                 let rec {l = (b?:0+a?:l)+a?.c?:h
                       and h = a?:h+a?.c?:h} in
                   h)
                \a!)
               \b!)
              \c!
     {q}{#<Cea u_q |i d ext_1 ext_2
               res |i d ext_1 ext_2>}
          --res-->
         local signals {a , b , c} in
           ((((a!*p[c!.d/d] ///x_q)///0)\a!)\b!)\c!
  :  Specif of m+p

time = 22.28s {+ display= 7.36s}
gc= 1
@ show set t_disabling2 = nextn(2, s);
t_disabling2 =
  local signals {a , b , c} in
    ((((a!*x_p[c!.d/d] ///b!=>q)///
         let rec {l = (b?:0+a?:l)+a?.c?:h and
          h = a?:h+a?.c?:h} in h)
        \a!)
       \b!)
      \c!
  :  Process of m+p

time = 1.24s {+ display= 1.64s}
@ show set t_disabling3 = nextn(3, s);
t_disabling3 =
  local signals {a , b , c} in
       ((((a!*p[c!.d/d] ///x_q)///0)\a!)\b!)\c!
  :  Process of m+p

time = 0.92s {+ display= 0.76s}
@ parse r = {t_disabling == p [> q,
m+p>            t_disabling2 == x_p,
m+p>            t_disabling3 == x_q};
r :  Relation of m+p

time = 1.06s
@ show prove r;
-------------------------------------------------
 Processing Equation:  "t_disabling == p[>q"
```

```
-----------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_disabling
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        local signals {a , b , c} in
          ((((a!*x_p[c!.d/d] ///b!=>q)///
               let rec {l = (b?:0+a?:l)+a?.c?:h
                 and h = a?:h+a?.c?:h} in l)
            \a!)
           \b!)
          \c!
   {p}{#<Cea u_p |d
             res |d>}
        --res-->
        local signals {a , b , c} in
          ((((a!*x_p[c!.d/d] ///b!=>q)///
               let rec {l = (b?:0+a?:l)+a?.c?:h
                 and h = a?:h+a?.c?:h} in h)
            \a!)
           \b!)
          \c!
   {q}{#<Cea u_q |i d ext_1 ext_2
             res |i d ext_1 ext_2>}
        --res-->
        local signals {a , b , c} in
          ((((a!*p[c!.d/d] ///x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  p[>q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p[>q
   {p}{#<Cea u_p |d
             res |d>}
        --res-->
        x_p
   {q}{#<Cea u_q |i d ext_1 ext_2
             res |i d ext_1 ext_2>}
        --res-->
        x_q
```

```
------------------------------------------------
     Proof Succeeds
------------------------------------------------
 Processing Equation:  "t_disabling2 == x_p"
------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_disabling2
   {x_p}{#<Cea u_x_p |i ext_1 ext_2
                res |i ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_x_p[c!.d/d] ///b!=>q)///
                 let rec {l = (b?:0+a?:l)+a?.c?:h
                     and h = a?:h+a?.c?:h} in
                   h)
               \a!)
              \b!)
             \c!
   {x_p}{#<Cea u_x_p |d
                res |d>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_x_p[c!.d/d] ///b!=>q)///
                 let rec {l = (b?:0+a?:l)+a?.c?:h
                    and h = a?:h+a?.c?:h} in
                   h)
               \a!)
              \b!)
             \c!

Evaluation of right-hand-side gives :
Specif of:  x_p
   {x_p}{#<Cea u_x_p |i d ext_1 ext_2
                res |i d ext_1 ext_2>}
          --res-->
          x_x_p

------------------------------------------------
     Proof Succeeds
------------------------------------------------
 Processing Equation:  "t_disabling3 == x_q"
------------------------------------------------
```

```
Evaluation of left-hand-side gives :
Specif of:  t_disabling3
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                     res |i d ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*p[c!.d/d] ///x_x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  x_q
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                     res |i d ext_1 ext_2>}
          --res-->
          x_x_q

-------------------------------------------------
     Proof Succeeds
CEA |--
  Isbisim
    {"t_disabling == p[>q", "t_disabling2 == x_p",
    "t_disabling3 == x_q"} :  Theorem of m+p

time = 73.34s {+ display= 0.60s}
gc= 1
@ file disabling.ec loaded.

time = 0.56s
@ load "sequence";
@
@
@ % Sequence
@
@
@ parse
@ t_sequence = local signals {a, b, c} in
m+p> (a!  * p[ c!  / d ] /// b!  => q ///
m+p> let rec h = a?:h + a?.b?.c?:0 in  h)
m+p> \a!  \b!  \c!;
t_sequence :  Process of m+p

time = 1.68s
@ show set s = eval t_sequence;
s =
```

```
  Specif of:
    local signals {a , b , c} in
      ((((a!*p[c!/d] ///b!=>q)///
          let rec {h = a?:h+(a?.b?).c?:0} in h)\a!)\b!)
        \c!
     {p}{#<Cea u_p |i ext_1 ext_2
               res |i ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!/d] ///b!=>q)///
                 let rec {h = a?:h+(a?.b?).c?:0} in h)
               \a!)
              \b!)
             \c!
     {q, p}{#<Cea u_p |d d     d     d
                 u_q |i d ext_1 ext_2
                 res |i d ext_1 ext_2>}
            --res-->
            local signals {a , b , c} in
              ((((a!*x_p[c!/d] ///x_q)///0)\a!)\b!)\c!
  :  Specif of m+p

time = 11.02s {+ display= 4.74s}
@ show set t_sequence2 = nextn(2, s);
t_sequence2 =
  local signals {a , b , c} in
     ((((a!*x_p[c!/d] ///x_q)///0)\a!)\b!)\c!
  :  Process of m+p

time = 0.90s {+ display= 0.72s}
@ parse r = {t_sequence == p :> q, t_sequence2 == x_q};
r :  Relation of m+p

time = 0.90s
@ show prove r;
-------------------------------------------------
 Processing Equation:  "t_sequence == p:>q"
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_sequence
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
```

```
        local signals {a , b , c} in
          ((((a!*x_p[c!/d] ///b!=>q)///
               let rec {h = a?:h+(a?.b?).c?:0} in h)
              \a!)
             \b!)
            \c!
   {q, p}{#<Cea u_p |d d     d     d
               u_q |i d ext_1 ext_2
               res |i d ext_1 ext_2>}
          --res-->
          local signals {a , b , c} in
            ((((a!*x_p[c!/d] ///x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  p:>q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p:>q
   {q, p}{#<Cea u_p |d d     d     d
               u_q |i d ext_1 ext_2
               res |i d ext_1 ext_2>}
          --res-->
          x_q

-------------------------------------------------
     Proof Succeeds
-------------------------------------------------
 Processing Equation:  "t_sequence2 == x_q"
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  t_sequence2
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                res |i d ext_1 ext_2>}
         --res-->
         local signals {a , b , c} in
           ((((a!*x_p[c!/d] ///x_x_q)///0)\a!)\b!)\c!

Evaluation of right-hand-side gives :
Specif of:  x_q
   {x_q}{#<Cea u_x_q |i d ext_1 ext_2
                res |i d ext_1 ext_2>}
         --res-->
```

```
          x_x_q

------------------------------------------------- .
     Proof Succeeds
CEA |-- Isbisim {"t_sequence == p:>q", "t_sequence2 == x_q"}
  :  Theorem of m+p

time = 36.44s {+ display= 0.42s}
@
@ file sequence.ec loaded.

time = 0.60s
@ load "choice";
@
@
@ % Choice
@
@
@ parse r = {p [] q == p + q};
r :  Relation of m+p

time = 0.62s
@ show prove r;
-------------------------------------------------

Evaluation of left-hand-side gives :
Specif of:  p[]q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p
   {q}{#<Cea u_q |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_q

Evaluation of right-hand-side gives :
Specif of:  p+q
   {p}{#<Cea u_p |i ext_1 ext_2
             res |i ext_1 ext_2>}
        --res-->
        x_p
   {q}{#<Cea u_q |i ext_1 ext_2
             res |i ext_1 ext_2>}
```

```
        --res-->
        x_q
```

```
     Proof Succeeds
CEA |-- Isbisim {"p[]q == p+q"} :  Theorem of m+p

time = 3.62s {+ display= 0.30s}
@
@ file choice.ec loaded.

time = 0.58s
@
@ file trad.ec loaded.

time = 0.42s
@
@ close;
```

3 Conclusion

La validation de cette traduction, nous amène à poser deux remarques :

1. Deux conditions sont nécessaires dans la version actuelle d'ECRINSpour la validation de la traduction :

   - les bisimulations doivent pouvoir être prouvées automatiquement; la traduction de chaque opérateur doit être un terme dont la spécification ne doit pas être infinie, ni spacialement (en une étape d'évaluation), ni temporellement (en se retransformant en un terme différent à chaque évaluation).
   - l'égalité entre les divers prédicats doit être décidable; cela limite la puissance d'expression des prédicats,

   ces deux conditions sont suffisantes pour valider une traduction.

2. Si les algèbres d'actions ne sont pas semblables, c'est-à-dire qu'il n'y a pas de projection entre les actions de l'algèbre source et celles de l'algèbre cible, il faut introduire une relation entre celles-ci par le biais de la fonction π que nous appellerons *fonction d'abstraction.*

   Nous désirons réaliser celle-ci au moyen de la théorie des critères d'abstraction [2] qui semblent allier la simplicité et la puissance requise.

   Ces différentes définitions sont l'objet d'un travail de recherche de thèse [4].

Nous tenons à remercier R. de Simone pour sa collaboration á la réalisation de cet article.

References

[1] D. Austry et G. Boudol,
"Algèbre de processus et synchronisation"
Theoretical Computer Science 30 (91-131), 1984

[2] G. Boudol,
"Notes on Algabraic Calculi of Processes",
Logics and Models of Concurrent Systems,
NTAO ASI series F13, K.Apt ed., 1985

[3] C.C.I.T.T.
Specification and Description Langage
Recommendation Z.100

[4] G. Doumenc
Validité de traducteur d'algèbres de processus
Thèse à paraitre

[5] I.S.O.
Information processing - Open System Interconnection
The definition of the specification language LOTOS
Draft proposal, ISO/DP 8807 97/21 N 423 (1985)

[6] E. Madelaine,
ECRINS : Manuel de référence
En cours de réalisation

[7] E. Madelaine et R. de Simone,
Ecrins : un laboratoire de preuve pour les calculs de processus
Rapport INRIA. RR 672, 1987

[8] R. Milner,
A Calculus for Communicating System
Lectures Notes in Computer Science 92, 1980

[9] G. Plotkin,
A Structural Approach to Operationnal Semantics
Report Daimi FN-19, Comp. Sci. Dept., Aarhus Univ., 1981

[10] R. de Simone,
Higher-level synchronising devices in MEIJE-SCCS
Theoretical Computer Science 37 (245-267), 1985

[11] D. Vergamini,
Verification by Means of Observational Equivalence on Automata
Rapport INRIA RR501, 1986

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique