Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 171 (2007) 155–170
www.elsevier.com/locate/entcs

The Decidability of the Structural Congruence for Beta-binders

Corrado Priami[1], Alessandro Romanel[2]

The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
Trento, Italy

Abstract

Beta-binders is a recent process algebra developed for modeling and simulating biological systems. As usual for process calculi, the semantic definition heavily relies on a structural congruence. The treatment of the structural congruence is essential for implementation. The proof of the decidability of this congruence, reported in this paper, is a first step towards implementations.

Keywords: Process Calculi, Structural Congruence, Decidability.

1 Introduction

Systems Biology studies the behaviour and relationships of the elements composing a particular biological system. Recently, some authors [16] argue that concurrency theory and *process algebras* are usuful to specify and simulate the behaviour of living matter. As a consequence, a number of process calculi has been adapted or newly developed for applications in systems biology [14,15,2,12]. The operational semantics of such process algebras allows to describe the dynamical evolution of a system. This semantics is near to the implementation and is usually strongly related to the concept of *structural congruence*. This paper focuses on *Beta-binders*, a process algebra introduced for better representing biological interactions. We develop on the structural congruence of both qualitative and quantitative [3] version of the calculus. The proof of the decidability of the structural congruence for Beta-binders, reported in this paper, is in fact a first step towards the implementation of a familiy of efficient stochastic simulators for Beta-binders.

[1] Email: priami@msr-unitn.unitn.it
[2] Email: a.romanel@msr-unitn.unitn.it

doi:10.1016/j.entcs.2007.05.014

The remainder of the paper is structured as follows. In Sect. 2 a short introduction of Beta-binders is reported, along with the description of some particular normal forms and an overview of the decidability of the structural congruence for the π-calculus. In Sect. 3 and Sect. 4 the proof of the decidability of the structural congruence for Beta-binders is presented. In Sect. 5 a generalization of the proof is given.

2 Preliminaries

In this section we report a short introduction to Beta-binders and a short overview of the most important results regarding the decidability of the structural congruence for the π-calculus.

2.1 Beta-binders

Beta-binders [12,13] is a process algebra developed for better representing the interactions between biological entities. The main idea is to encapsulate π-calculus processes into *boxes* with interaction capabilities, also called *beta-processes*. Like the π-calculus also Beta-binders is based on the notion of *naming*. Thus, we assume the existence of a countably infinite set $\mathcal{N}$ of names (ranged over by lower-case letter). The processes wrapped into boxes, also called *pi-processes*, are given by the following context free grammar:

$$P ::= nil \mid \pi.P \mid P|Q \mid (\nu y)P \mid !P$$
$$\pi ::= \overline{x}\langle y\rangle \mid x(y) \mid \tau \mid expose(x,\Gamma) \mid hide(x) \mid unhide(x)$$

The syntax of the π-calculus is enriched by the last three options for π to manipulate the interactions *sites* of the boxes. Beta-processes are defined as pi-processes prefixed by specialised binders that represent interaction capabilities. An *elementary beta binder* has the form $\beta(x,\Gamma)$ (active) or $\beta^h(x,\Gamma)$ (hidden) where the name x is the subject of the beta binder and Γ represents the type of x. With $\widehat{\beta}$ we denote either β or β^h. A *well-formed* beta binder (ranged over by $\boldsymbol{B}$, $\boldsymbol{B}_1$, $\boldsymbol{B}'$, $\cdots$) is a non-empty string of elementary beta binder where subjects are all distinct. The function $sub(\boldsymbol{B})$ returns the set of all the beta binder subjects in $\boldsymbol{B}$. Moreover, $\boldsymbol{B}^*$ denote either a well-formed beta binder or the empty string. *Beta-processes* (ranged over by B, B_1, B', $\cdots$) are generated by the following context free grammar:

$$B ::= Nil \mid \boldsymbol{B}[P] \mid B \parallel B$$

The system is either the deadlock beta-process *Nil* or a parallel composition of boxes $\boldsymbol{B}[P]$. The structural congruence for Beta-binders is defined through a structural congruence over pi-processes and a structural congruence over beta-processes.

Definition 2.1 The structural congruence over pi-processes, denoted $\equiv$, is the

smallest relation which satisfies the laws in Fig. 1 (group a) and the structural congruence over beta-processes, denoted $\equiv$, is the smallest relation which satisfies the laws in Fig. 1 (group b).

group a - **pi-processes**		group b - **beta-processes**	
$a.1)$	$P_1 \equiv P_2$ if P_1 and P_2 are α-equivalent	$b.1)$	$\mathbf{B}[P_1] \equiv \mathbf{B}[P_2]$ if $P_1 \equiv P_2$
$a.2)$	$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$	$b.2)$	$B_1 \parallel (B_2 \parallel B_3) \equiv (B_1 \parallel B_2) \parallel B_3$
$a.3)$	$P_1 \mid P_2 \equiv P_2 \mid P_1$	$b.3)$	$B_1 \parallel B_2 \equiv B_2 \parallel B_1$
$a.4)$	$P \mid nil \equiv P$	$b.4)$	$B \parallel Nil \equiv B$
$a.5)$	$(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P$	$b.5)$	$\mathbf{B}_1\mathbf{B}_2[P] \equiv \mathbf{B}_2\mathbf{B}_1[P]$
$a.6)$	$(\nu z)P \equiv P$ if $x \notin fn(P)$	$b.6)$	$\mathbf{B}^*\widehat{\beta}(x:\Gamma)[P] \equiv \mathbf{B}^*\widehat{\beta}(y:\Gamma)[P\{y/x\}]$ with y fresh in P and $y \notin sub(\mathbf{B}^*)$
$a.7)$	$(\nu z)(P_1 \mid P_2) \equiv P_1 \mid (\nu z)P_2$ if $z \notin fn(P_1)$		
$a.8)$	$!P \equiv P \mid !P$		

Fig. 1. Structural laws for Beta-binders.

Notice that the same symbol is used to denote both congruences. The intended relation is disambiguated by the context of application.

In the stochastic extension of Beta-Binders [3] the syntax is enriched in order to allow a Gillespie Stochastic Simulation Algorithm (SSA) implementation [9]. The prefix $\pi.P$ is replaced by $(\pi, r).P$, where r is the single parameter defining an exponential distribution that drives the stochastic behaviour of the action corrisponding to the prefix π. Moreover, the classical replication $!P$ is replaced by the so called *guarded replication* $!\pi.P$. In order to manage this type of replication, the structural law $!P \equiv P \mid !P$ is replaced by the law $!(\pi, r).P \equiv (\pi, r).(P|!(\pi, r).P)$.

Notice that for the purpose of this paper we are not interested in the semantic of the language. We refer the reader to [12,13,3] for a more detailed description of both the qualitative and quantitative version of Beta-binders.

2.2 Normal forms

In [7] two normal forms for π-calculus processes, called *webform* and *super webform*, are introduced.

A process P is *fresh* if $x \notin fn(P)$ whenever (νx) is not in the scope of any guard or replication (called *outer restriction*) in P, and every restriction (νx) occurs at most once as outer restriction in P. For each process P there exists a fresh process P' such that $P' \equiv_\alpha P$. Let P be a fresh process. Let $\mathrm{os}(P)$, the *outer subterms* of P, be the set of occurrences of subterm $\pi.Q$ and $!Q$ of P that are not in the scope of any guard or replication. Let $\mathrm{or}(P)$, the *outer restrictions* of P, be the set of names x such that (νx) is not in the scope of any guard or replication in P and such that x occurs free in some outer subterm of P. Finally, let $\mathrm{og}(P)$, the *outer graph* of P, be the undirected bipartite graph with nodes $\mathrm{os}(P) \cup \mathrm{or}(P)$ and with an edge between $R \in \mathrm{os}(P)$ and $x \in \mathrm{or}(P)$ if $x \in fn(R)$.

A process $P = (\nu x_1)...(\nu x_k)(P_1 \mid ... \mid P_m)$ with $k \geq 0$ and $m \geq 1$ is a *web* if: (1) every process P_i is a replication $!Q$ or a guarded process $\pi.Q$; (2) $x_1, ..., x_k$ are all distinct (P is fresh); (3) for each x_j there exists a process P_i such that $x_j \in fn(P_i)$;

(4) $og(P)$ is connected. Every replication $!P$ and every guarded process $\pi.P$ is a web (with $k = 0$ and $m = 1$). No web is congruent to the inactive process nil. A web should be denoted with the set $\{x_1, ..., x_k, P_1, ..., P_m\}$ wich lists the names of the outer restrictions and the outer subterms. A *webform* of a fresh process P, denoted with $wf(P)$, is the composition of all the webs $(\nu x_1)...(\nu x_k)(P_1 \mid ... \mid P_m)$ such that $\{x_1, ..., x_k, P_1, ..., P_m\}$ is a connected component of $og(P)$ (in [7] an inductively computation of $wf(P)$ is reported). If $og(P)$ is the empty graph, then $wf(P) = nil$. The *super webform* of a fresh process P, denoted with $swf(P)$, is inductively defined in the following way: $swf(P) = wf(subwf(P))$ where, by definition, $subwf(P)$ is obtained from P by replacing every outer subterm $\pi.Q$ of P with $\pi.swf(Q)$ and every outer subterm $!Q$ with $!swf(Q)$. See [7] for a more detailed description.

2.3 The decidability of the structural congruence for the π-calculus

The most important results for the decidability of the structural congruence for the π-calculus are those presented by J. Engelfriet in [4] and by J. Engelfriet e T.E. Gelsema in [5,6,8,7]. They consider the syntax of the *small π-calculus* (presented in [11]) and the congruences over the set of processes generated by a subcollection of the structural laws presented in Fig. 2 (where, for our purpose, we add the congruence $\equiv^{\mathbf{min}}$). The standard structural congruence, defined in [4,5] and denoted with $\equiv^{\mathbf{std}}$, is determined by the laws (α), (1.1), (1.2), (1.3), (2.1), (2.2), (2.3) and (3.1). In [6], the middle congruence, denoted with $\equiv^{\mathbf{md}}$, was introduced to give a different view of the treatment of replication. The decidability of the middle congruence was shown in [8]. They reduce it to the decidability of extended structural congruence, denoted with $\equiv^{\mathbf{ext}}$, that was shown in [5]. In [7], instead, was shown the decidability of the replication free congruence, denoted with $\equiv^{\mathbf{!fr}}$, and the decidability of the standard congruence for the subclass of *replication restricted* processes. Formally, a process P is *replication restricted* if for every subterm $!R$ of P and every (νx) that covers $!R$ in P, if $x \in fn(R)$, then $x \in fn(S)$ for every component S of R where with component we mean a *web*. The decidability of the structural congruence for this subclass of processes is reduced to the problem of solving certain systems of linear equations with coefficents in $\mathbb{N}$.

3 Structural congruence over beta-processes

The structural laws for Beta-binders, presented in Fig. 1, are divided in two groups: the laws for pi-processes (*group a*) and the laws for beta-processes (*group b*). From law $b.1$ it turns out that the decidability of the structural congruence over pi-processes is a necessary condition for the decidability of the structural congruence over beta-processes.

The congruences that we consider in this paper are $\equiv_{bb}^{\mathbf{min}}$ and $\equiv_{bb}^{\mathbf{std}}$. Congruence $\equiv_{bb}^{\mathbf{min}}$ is generated by the structural laws of *group a* and the laws $b.1$, $b.5$ and $b.6$. Congruence $\equiv_{bb}^{\mathbf{std}}$ is generated by all the structural laws of *group a* and *group b*.

First, we prove the decidability of the congruence $\equiv_{bb}^{\mathbf{min}}$ making some assump-

	rule	$\equiv^{\mathbf{min}}$	$\equiv^{\nu\mathbf{fr}}$	$\equiv^{\mathbf{!fr}}$	$\equiv^{\mathbf{std}}$	$\equiv^{\mathbf{md}}$	$\equiv^{\mathbf{ext}}$
(α)	$P_1 \equiv P_2$ if P_1 and P_2 are α-equivalent	+	+	+	+	+	+
(1.1)	$P \mid nil \equiv P$		+	+	+	+	+
(1.2)	$P_1 \mid P_2 \equiv P_2 \mid P_1$	+	+	+	+	+	+
(1.3)	$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3)$	+	+	+	+	+	+
(2.1)	$(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P$	+	+	+	+	+	+
(2.2)	$(\nu z)P \equiv P$ if $x \notin fn(P)$			+	+	+	+
(2.3)	$(\nu z)(P_1 \mid P_2) \equiv P_1 \mid (\nu z)P_2$ if $z \notin fn(P_1)$			+	+	+	+
(3.1)	$!P \equiv P \mid !P$		+		+	(+)	+
(3.6)	$!(P \mid Q) \equiv !(\ P \mid Q) \mid P$					+	(+)
(3.2)	$!(P \mid Q) \equiv !P \mid !Q$						+
(3.3)	$!!P \equiv !P$						+
(3.4)	$!nil \equiv nil$						+
(2.4)	$(\nu x)\pi.P \equiv \pi.(\nu x)P$ if $x \notin n(\pi)$						+

Fig. 2. Structural laws for the π-calculus.

tions: (1) we restrict the *well-formedness* definition by assuming that a *well-formed* beta binder (ranged over by $\boldsymbol{B}$, $\boldsymbol{B}_1$, $\boldsymbol{B}'$, $\cdots$) is a non-empty string of elementary beta binder where subjects and types are all distinct; (2) we assume that the structural congruence over pi-processes is decidable, and therefore we assume that there exists a function $PiStdCong : \mathcal{P} \times \mathcal{P} \rightarrow \{true, false\}$ that accepts two pi-processes as parameters and returns *true* if the pi-processes are structural congruent, and returns *false* otherwise; (3) we assume that the types of the beta binders are defined over algebric structures with decidable equality relation, and therefore we assume that there exists a function $Equal : \Gamma \times \Delta \rightarrow \{true, false\}$ that accepts two types as parameters and returns *true* if the types are equal, and returns *false* otherwise. Then, we prove the decidability of the congruence $\equiv_{bb}^{\mathbf{std}}$ always under the previous assumptions. Finally, we will analyze in detail the decidability of the structural congruence over pi-processes.

We consider two beta-processes $\boldsymbol{B}[P]$ and $\boldsymbol{B}'[P']$. We notice that the laws of *group b* related to the congruence $\equiv_{bb}^{\mathbf{min}}$ only refers to the structure of the beta binders lists $\boldsymbol{B}$ and $\boldsymbol{B}'$. In fact, the two lists are considered congruent only if they are equal (law $b.1$), or if $\boldsymbol{B}$ is a permutation of $\boldsymbol{B}'$ that satisfies the laws $b.5$ and $b.6$.

For this reason the decidability of the congruence $\equiv_{bb}^{\mathbf{min}}$ can be described through a function $BBMinCong : \boldsymbol{B}[P] \times \boldsymbol{B}[P] \rightarrow \{true, false\}$ defined by induction on the structure of beta-processes in the following way:

$$BBMinCong(\epsilon[P], \epsilon[P']) = PiStdCong(P, P')$$

$$BBMinCong(\epsilon[P], \boldsymbol{B}'[P']) = BBMinCong(\boldsymbol{B}[P], \epsilon[P']) = false$$

$$BBMinCong(\widehat{\beta}(x : \Gamma)\boldsymbol{B}^*[P], \boldsymbol{B}'[P']) = \begin{cases} BBMinCong(\boldsymbol{B}^*[P\{z/x\}], \boldsymbol{B}_1^*\boldsymbol{B}_2^*[P'\{z/y\}]) & if\ (1) \\ BBMinCong(\boldsymbol{B}^*[P], \boldsymbol{B}_1^*\boldsymbol{B}_2^*[P']) & if\ (2) \\ false & o.w. \end{cases}$$

(1) $\boldsymbol{B'} = \boldsymbol{B}_1^* \widehat{\beta}(y:\Delta)\boldsymbol{B}_2^*$ *with* $Equal(\Gamma,\Delta) \wedge (x \neq y) \wedge\ z \notin fn(P) \cup fn(P') \cup sub(\boldsymbol{B}^*) \cup sub(\boldsymbol{B}_1^*\boldsymbol{B}_2^*)$
(2) $\boldsymbol{B'} = \boldsymbol{B}_1^* \widehat{\beta}(x:\Delta)\boldsymbol{B}_2^*$ *with* $Equal(\Gamma,\Delta)$

If the lists $\boldsymbol{B}$ and $\boldsymbol{B'}$ are not empty, then there are three different cases: (1) if a type corrispondence between the first beta binders $\widehat{\beta}(x:\Gamma)$ of $\boldsymbol{B}$ and one beta binder $\widehat{\beta}(y:\Delta)$ of $\boldsymbol{B'}$ such that $(x \neq y)$ exists, then the function $BBMinCong$ is recursively invoked on the beta-processes $\boldsymbol{B}_1[P\{z/x\}]$ and $\boldsymbol{B}_2[P'\{z/y\}]$, where $z \notin fn(P) \cup fn(P') \cup sub(\boldsymbol{B}_1) \cup sub(\boldsymbol{B}_2)$, $\boldsymbol{B}_1$ is obtained from $\boldsymbol{B}$ deleting the beta binder $\widehat{\beta}(x:\Gamma)$ and $\boldsymbol{B}_2$ is obtained from $\boldsymbol{B'}$ deleting the beta binder $\widehat{\beta}(y:\Delta)$; (2) if the first beta binder of the list $\boldsymbol{B}$ is equal to one beta binder of the list $\boldsymbol{B'}$, then the function $BBMinCong$ is recursively invoked on the beta-processes $\boldsymbol{B}_1[P]$ and $\boldsymbol{B}_2[P']$, where $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are respectively obtained from $\boldsymbol{B}$ and $\boldsymbol{B'}$ deleting the equal beta binders; (3) if no correspondence between the first beta binder of $\boldsymbol{B}$ and one beta binder of $\boldsymbol{B'}$ exists, then the function returns *false*.

If only one of the beta binders lists $\boldsymbol{B}$ and $\boldsymbol{B'}$ is empty, then the function returns *false*.

If both $\boldsymbol{B}$ and $\boldsymbol{B'}$ are empty, then the function $PiStdCong$ is invoked on the pi-processes P and P'. In this case the function $BBMinCong$ returns the result of $PiStdCong(P, P')$.

We notice that the decidability of the structural congruence over pi-processes is not only necessary condition but also sufficient condition for the decidability of the congruence $\equiv_{bb}^{\mathbf{min}}$.

Now we analyze the congruence $\equiv_{bb}^{\mathbf{std}}$. The law $b.2$ regards parallelization with the inactive beta-process Nil and the laws $b.3$ and $b.4$ are associtivity and commutativity rules. The decidability of the congruence $\equiv_{bb}^{\mathbf{std}}$ can be described through a function $BBStdCong : B \times B \rightarrow \{true, false\}$ defined by induction on the structure of beta-processes in the following way:

$$BBStdCong(Nil, B') = \begin{cases} false & if\ (1) \\ true & o.w. \end{cases}$$

$$BBStdCong(\boldsymbol{B}_1[P_1], B') = \begin{cases} BBStdCong(Nil, Remove(\boldsymbol{B''}[P''], B')) & if\ (2) \\ false & o.w. \end{cases}$$

$$BBStdCong(Nil \mid\mid B, B') = BBStdCong(B, B')$$

$$BBStdCong(\boldsymbol{B}_1[P_1] \mid\mid B, B') = \begin{cases} BBStdCong(B, Remove(\boldsymbol{B''}[P''], B')) & if\ (2) \\ false & o.w. \end{cases}$$

(1) $\exists\, j, n \in \mathbb{N}^+$ *with* $(B' = B_1 || \cdots || B_n) \wedge (j \leq n) \wedge (B_j = \boldsymbol{B''}[P''])$
(2) $\exists\, j, n \in \mathbb{N}^+$ *with* $(B' = B_1 || \cdots || B_n) \wedge (j \leq n) \wedge (B_j = \boldsymbol{B''}[P'']) \wedge BBMinCong(\boldsymbol{B}_1[P_1], \boldsymbol{B''}[P''])$

where if $B' = B_1 || \cdots || B_n$ e $n = 1$ then B' is a box or the inactive beta-process *Nil*. The function $Remove : \boldsymbol{B}[P] \times B \rightarrow B$ is defined in the following way:

$$Remove(\boldsymbol{B}[P], Nil) = Nil$$

$$Remove(\boldsymbol{B}[P], \boldsymbol{B}'[P']) = \begin{cases} Nil & if\ \boldsymbol{B}'[P'] = \boldsymbol{B}[P] \\ \boldsymbol{B}'[P'] & o.w. \end{cases}$$

$$Remove(\boldsymbol{B}[P], B_1||B') = \begin{cases} B' & if\ (1) \\ B_1\ ||\ Remove(\boldsymbol{B}[P], B') & o.w. \end{cases}$$

(1) $(B_1 = \boldsymbol{B}''[P'']) \wedge (\boldsymbol{B}''[P''] = \boldsymbol{B}[P])$

If B and B' are composed by a different number of boxes, then they are not congruent and the function returns *false*. If there exists a bijection between the boxes $\boldsymbol{B}_i[P_i]$ of B and the boxes $\boldsymbol{B}'_j[P'_j]$ of B' such that for each corrispondence it is $\boldsymbol{B}_i[P_i] \equiv_{bb}^{\mathbf{min}} \boldsymbol{B}'_j[P'_j]$, then the two beta-processes are congruent and the function returns *true*. Otherwise the function returns *false*.

Lemma 3.1 *The decidability of the structural congruence over pi-processes is a necessary and sufficient condition for the decidability of the structural congruence over beta-processes.*

4 Structural congruence over pi-processes

The results on which we base part of our work are those obtained from J. Engelfriet e T.E. Gelsema in [7] and reported in Sect. 2.3. In fact, the decidability of the structural congruence over beta-processes strongly depends on the structural congruence over pi-processes. Moreover, the pi-processes are *small pi-Calculus* processes with an extended set of actions, and the structural laws for the structural congruence over pi-processes are the same ones for the structural congruence over small pi-Calculus processes. Thereafter, the results presented in [7] for the standard congruence $\equiv^{\mathbf{std}}$ and the replication free congruence $\equiv^{\mathbf{!fr}}$ can also be used in this context because they do not depend on the specific types of actions contained in the processes.

Lemma 4.1 *The congruences $\equiv_{bb}^{\boldsymbol{std}}$ and $\equiv_{bb}^{\boldsymbol{min}}$ are decidable for the subclass of beta-processes with replication restricted pi-processes.*

Proof. Immediate from the definition of functions *BBStdCong* and *BBMinCong* and from the results presented in [7]. □

We notice that this result is valid for the qualitative version of Beta-binders. Now consider the stochastic extension of Beta-binders. The classical replication is replaced with the guarded replication and hence the syntax and the structural laws for pi-processes are modified substituting respectively $!P$ with $!\pi.P$ and $!P \equiv P \mid !P$ with $!\pi.P \equiv \pi.(P \mid !\pi.P)$ [3]. The Fig. 3 shows the congruences over guarded replication pi-processes that we will consider in the remainder of the paper.

[3] For semplicity in the remainder of the paper we omit the rate r in the prefixes beacause not important for our purpose.

	rule	$\equiv^{\mathbf{min}}$	$\equiv^{\mathbf{!fr}}$	$\equiv^{\mathbf{std}}$
(α)	$P_1 \equiv P_2$ if P_1 and P_2 are α-equivalent	+	+	+
(1.1)	$P \mid nil \equiv P$		+	+
(1.2)	$P_1 \mid P_2 \equiv P_2 \mid P_1$	+	+	+
(1.3)	$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$	+	+	+
(2.1)	$(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P$	+	+	+
(2.2)	$(\nu z)P \equiv P$ if $x \notin fn(P)$		+	+
(2.3)	$(\nu z)(P_1 \mid P_2) \equiv P_1 \mid (\nu z)P_2$ if $z \notin fn(P_1)$		+	+
(3.1)	$!\pi.P \equiv \pi.(P \mid !\pi.P)$			+

Fig. 3. Structural laws for the small π-calculus with guarded replication.

A process that only uses guarded replication is, by definition, replication restricted. Therefore, the standard structural congruence over guarded replication pi-processes is decidable. More precisely, this result is valid if we consider the replication structural law $!P \equiv P \mid !P$, whereas it must be prooved if we consider the replication structural law $!\pi.P \equiv \pi.(P \mid !\pi.P)$.

In this paper we want to face the problem of decidability of structural congruence for guarded replication pi-processes from another point of view. In particular, we will consider the structure of pi-processes that only use guarded replication. In [7], the main difficulty in showing the decidability of $\equiv^{\mathbf{std}}$ for replication restricted processes is the treatment of replication, which allows a process to grow indefinitely and without particular structure in its number of subterms. A process that uses guarded replication, instead, allows a process to grow indefinitely in its number of subterms mantaining structure.

Given a generic pi-process P, this characteristic allows us to define a function that recognizes and eliminates all the expanded replication in P.

This function, that we call *Implosion*, is defined by induction on the structure of processes:

$$Implosion(nil) = nil$$

$$Implosion(!\pi.P') = !Implosion(\pi.P')$$

$$Implosion((\nu x)P') = (\nu x)Implosion(P')$$

$$Implosion(P_0|P_1) = Implosion(P_0) \mid Implosion(P_1)$$

$$Implosion(\pi.P') = \begin{cases} !\pi.Q & if\ (1) \\ \pi.Implosion(P') & o.w. \end{cases}$$

(1) $\exists\, j, n \in \mathbb{N}^+$ *with* $(P' = P_1 \mid \cdots \mid P_n) \wedge (j \leq n) \wedge (P_j = !\pi.R) \wedge$
$(Q = Implosion(RemovePI(P_j, P'))) \wedge (Q \equiv^{\mathbf{!fr}} Implosion(R))$

where if $P' = P_1 \mid \cdots \mid P_n$ and $n = 1$ then P' is in the form nil, $\pi.R$, $!\pi.R$, or $(\nu x)R$. The function $RemovePI : \mathcal{P} \times \mathcal{P} \to \mathcal{P}$ is defined in the following way:

$$RemovePI(P, P') = \begin{cases} P_1 & if\ (P' = P_0 \mid P_1) \wedge (P_0 = P) \\ P_0 \mid RemovePI(P, P_1) & if\ (P' = P_0 \mid P_1) \wedge (P_0 \neq P) \\ nil & if\ (1) \\ P' & o.w. \end{cases}$$

(1) $((P' = nil) \vee (P' = \pi.R) \vee (P' = !\pi.R) \vee (P' = (\nu x)R)) \wedge (P = P')$

Since the processes have finite length the function *Implosion* ends.

Lemma 4.2 *Let P be a pi-process that only uses guarded replication. Then $Implosion(P) \equiv^{std} P$.*

Proof. Every substitution and modification that the function *Implosion* carries out on the structure of the process P comes from the recursive invocation of $Implosion(\pi.P')$. This substitutions and modifications are equivalent to the application of a sequence of structural laws α, 1.1, 1.2, 1.3, 2.1, 2.1, 2.3, 3.1. This laws are the structural laws of the congruence $\equiv^{\mathbf{std}}$. For this reason $Implosion(P) \equiv^{\mathbf{std}} P$.□

Now consider the subclass of guarded replication pi-processes that does not contain expanded replications. We call this subclass $\mathcal{P}_{rp}$.

Lemma 4.3 *Let P and Q be pi-processes belonging to $\mathcal{P}_{rp}$. Then $P \equiv^{std} Q$ iff $P \equiv^{!fr} Q$.*

Proof. ($\Rightarrow$) To show this implication we prove that in $P \equiv^{\mathbf{std}} Q$ the law 3.1 is never used. Assume that P is obtainable from Q by applying, for some subterm of Q, the law 3.1. This means that one of the two processes has a subterm in the form $\pi.(R \mid !\pi.R)$. But the subterm $\pi.(R \mid !\pi.R)$ expands the replication $!\pi.R$ and this contradicts our initial assumption that $P \in \mathcal{P}_{rp}$. Therefore, the law 3.1 is never used and the implication is true.
($\Leftarrow$) Since the structural laws of the congruence $\equiv^{\mathbf{!fr}}$ are a subset of the structural laws of the congruence $\equiv^{\mathbf{std}}$ then $P \equiv^{\mathbf{!fr}} Q$ implies $P \equiv^{\mathbf{std}} Q$. □

Lemma 4.4 *Let P and Q be guarded replication pi-processes. Then $P \equiv^{std} Q$ iff $Implosion(P) \equiv^{!fr} Implosion(Q)$.*

Proof. ($\Rightarrow$) Since $Implosion(P) \equiv^{\mathbf{std}} P \equiv^{\mathbf{std}} Q \equiv^{\mathbf{std}} Implosion(Q)$ (using Lemma 4.2) we obtain that $Implosion(P) \equiv^{\mathbf{std}} Implosion(Q)$. Since the pi-processes $Implosion(P)$ and $Implosion(Q)$ does not contain expanded replication, we have that $Implosion(P) \equiv^{\mathbf{std}} Implosion(Q)$ (using Lemma 4.3) implies $Implosion(P) \equiv^{\mathbf{!fr}} Implosion(Q)$.
($\Leftarrow$) The structural laws of congruence $\equiv^{\mathbf{!fr}}$ are a subset of the structural laws of the congruence $\equiv^{\mathbf{std}}$. For this reason $Implosion(P) \equiv^{\mathbf{!fr}} Implosion(Q)$ implies $Implosion(P) \equiv^{\mathbf{std}} Implosion(Q)$. For the Lemma 4.2 we have that $P \equiv^{\mathbf{std}}$

$Implosion(P) \equiv^{\mathbf{std}} Implosion(Q) \equiv^{\mathbf{std}} Q$ and therefore, for transitivity, we have that $P \equiv^{\mathbf{std}} Q$. □

We notice that the function *Implosion* is intrinsically based on the congruence relation $\equiv^{\mathbf{!fr}}$. So, we can assert that there exists a procedure that allows to verify the standard congruence over guarded replication pi-processes using only the laws of the replication free congruence. Therefore, this procedure is effectively decidable only if the replication free congruence is decidable. In [7] (Theorem 3.10) Engelfriet proves that

$$P \equiv^{\mathbf{!fr}} Q \iff swf(P) \equiv_{\alpha} swf(Q)$$

where, due to some initial conventions, with $\equiv_{\alpha}$ he means $\equiv^{\mathbf{min}}$. For showing that $\equiv^{\mathbf{!fr}}$ is really decidable, we prove that the problem $P \equiv^{\mathbf{min}} Q$ is equivalent to an isomorphism problem over labelled directed acyclic graphs (lDAGs), that we know to be a decidibile problem.

Let P be a pi-process. We define a procedure that permits to construct the lDAG, denoted with $GS(P)$, that we will use in the next proof.

Definition 4.5 Let P be a pi-process. The graph $GS(P)$ is built from the syntax tree of P applying the following transformations:

1) the multiple composition of binary parallels are replaced with a unique n-ary parallel (Fig. 4);
2) the restriction sequences are transformed as shown in Fig. 5;
3) the output nodes, that have label $\overline{x}\langle n\rangle$, are replaced with a sequence of two nodes where the first has label $\overline{x}$ and the second has label $\langle n\rangle$ (Fig. 6);
4) An edge is added from each node that contains a binding occurrence for a name to all the nodes that contains names binded to this occurrence (Fig. 7);
5) Every name that binds something is replaced with 0 and every binded name is replaced with 1.

Without loss of generality we assume that 0 and 1 do not belong to the set of names $\mathcal{N}$.

The GS graph can be built in polynomial time and is essential for the treatment of the α-conversion and the commutativity of restrictions. Let $P = (\nu x)(\nu y)(a(x).nil \mid y(z).\overline{b}\langle z\rangle.\overline{x}\langle m\rangle.nil)$. Fig. 8 shows the building procedure of the graph $GS(P)$. With $\cong$ with denote the classical isomorphism relation between lDAGs, where the ismorphism is a bijection of nodes that mantains labels and adiacency properties.

Lemma 4.6 *Let P and Q be pi-processes. Then $P \equiv^{\mathbf{min}} Q$ iff $GS(P) \cong GS(Q)$.*

Proof. Let R be a pi-process. Then the nodes of the graph $GS(R) = (V_R, E_R)$ are enumerated with a pre-order starting from the root of the cover tree of the

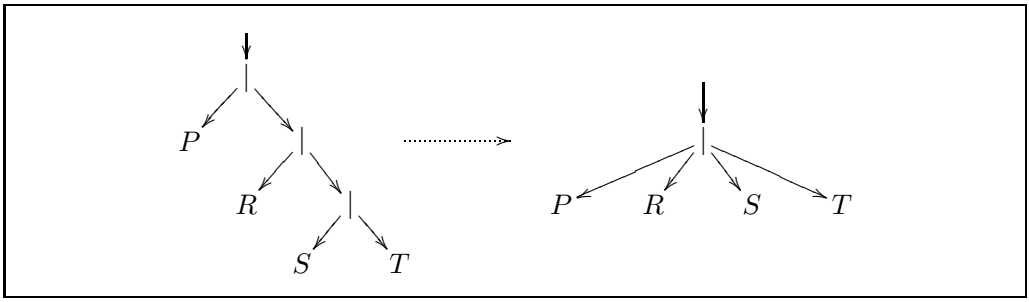

Fig. 4. Binary parallel composition transformation.

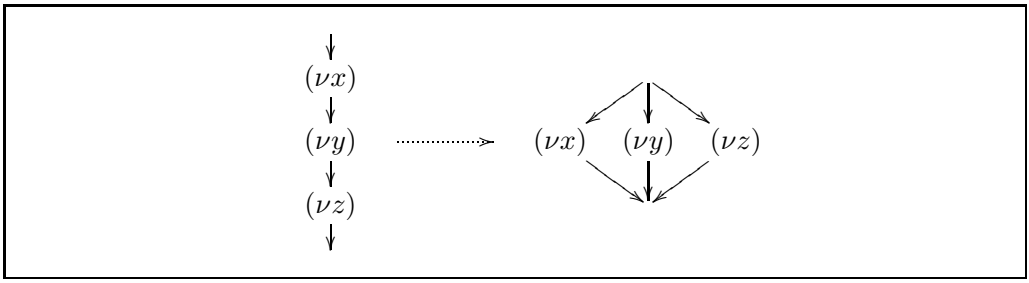

Fig. 5. Restriction sequences transformation.

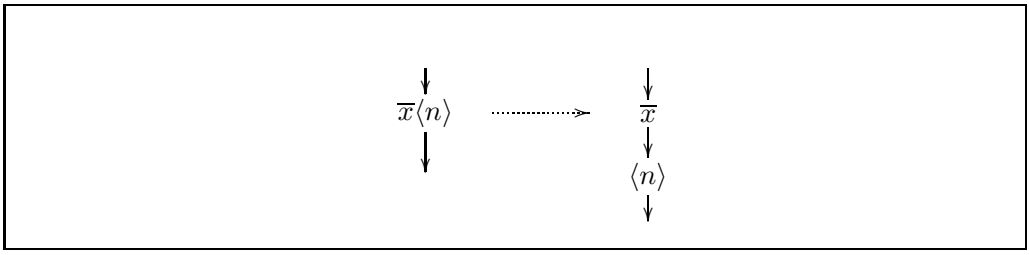

Fig. 6. Output node transformation.

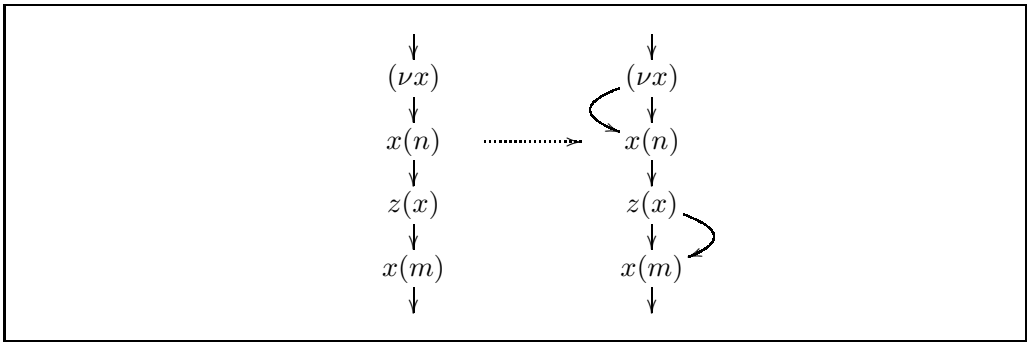

Fig. 7. Edge addition. Notice that there is not edge between the restriction (νx) and the node $x(m)$.

graph, without considering the added edges (Fig. 9). $(\Rightarrow)$ We assume by hypothesis that $P \equiv^{\mathbf{min}} Q$. This means that P is obtainable from Q (and viceversa) by applying, in Q, a sequence $r_0, ..., r_n$ of structural laws (we assume that r_i supplies the information about where to apply the law in Q). Notice that we obtain the process $Q_i \equiv^{\mathbf{min}} Q$ applying in Q the law r_i. The construction of an isomorphism

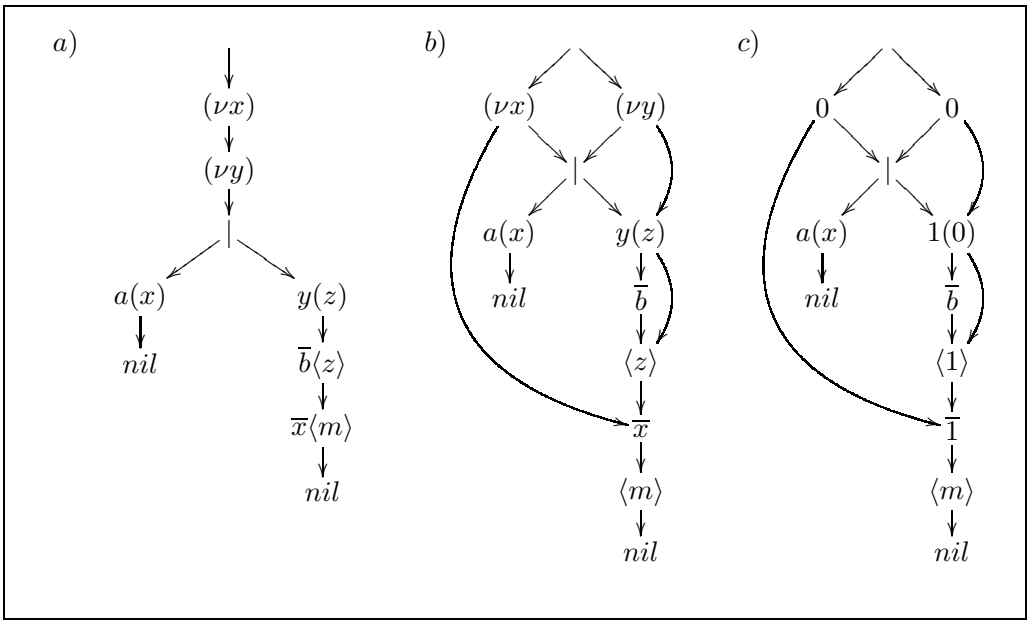

Fig. 8. Transformation of the syntax tree of the pi-process $P = (\nu x)(\nu y)(a(x).nil \mid y(z).\overline{b}\langle z\rangle.\overline{x}\langle m\rangle.nil)$ in $GS(P)$. In a) is shown the syntax tree of P. In b) is shown the application of the transformations 1,2,3 and 4. In c) the transformation is completed.

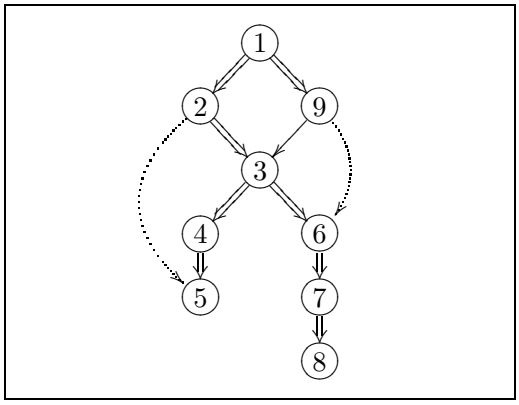

Fig. 9. Example of graph node enumeration. The double lined arrows show the cover tree of the graph. The dotted arrows represent the added edge that we do not consider.

ϕ_i between $GS(Q)$ and $GS(Q_i)$ depends on the structural law r_i applied. We have three cases: **(1)** Suppose that Q_i is obtained from Q by applying the law (2.1) on a subterm $(\nu x)(\nu y)Q'$ of Q. Therefore, the only difference between Q and Q_i is that in Q_i the subterm $(\nu x)(\nu y)Q'$ appears in the form $(\nu y)(\nu x)Q'$. Let n_1 and n_2 be the nodes in $GS(Q)$ that represent respectively the restrictions (νx) and (νy) of the subterm $(\nu x)(\nu y)Q'$. In the graph Q_i the representation is inverted. In fact, n_1 represents (νy) while n_2 represents (νx). Let ϕ_i be the mapping between the nodes of $GS(Q)$ and $GS(Q_i)$ such that for each node $n \in V_Q$ with $n \notin \{n_1, n_2\}$ is $\phi_i(n) = n$ and such that $\phi_i(n_1) = n_2$ and $\phi_i(n_2) = n_1$. ϕ_i is an isomorphism because, for the GS construction, the nodes $n \in V_Q$ and $\phi_i(n) \in V_{Q_i}$ have the same

labels and for each edge $(n, n') \in E_Q$ it is $(\phi_i(n), \phi_i(n')) \in E_{Q_i}$.
(2) Suppose that Q_i is obtained from Q by applying the law (1.2) on a subterm $Q'|Q''$ of Q. Thereafter, the only difference between Q and Q_i is that in Q_i the subterm $Q'|Q''$ appears in the form $Q''|Q'$. Let n_0 and n_1 be the nodes in $GS(Q)$ that represent respectively the root node of the subgraph $GS(Q')$ and the root node of the subgraph $GS(Q'')$. In $GS(Q_i)$ the representation is inverted. In fact, n_1 represents the root node of the subgraph $GS(Q'')$ while n_2 represents the root node of the subgraph $GS(Q')$. Let ϕ_i be the mapping between the nodes of $GS(Q)$ and $GS(Q_i)$ such that for each node $n \in V_Q$, with $n \notin \{GS(Q'), GS(Q'')\}$, it is $\phi_i(n) = n$ and such that for each node $n_1 + i$, with $i \geq 0$ and $n_1 + i \in GS(Q')$, and for each node $n_2 + j$, with $j \geq 0$ and $n_2 + j \in GS(Q'')$, it is $\phi_i(n_1 + i) = n_2 + i$ and $\phi_i(n_2 + j) = n_1 + j$. Also in this case ϕ_i is an isomorphism because, for the GS construction, the nodes $n \in V_Q$ and $\phi_i(n) \in V_{Q_i}$ have the same labels and for each edge $(n, n') \in E_Q$ it is $(\phi_i(n), \phi_i(n')) \in E_{Q_i}$.
(3) If Q_i is obtained from Q by applying α-conversion or the law (1.3) then the isomorphism ϕ_i is the identity id because, for the GS construction, the graphs $GS(Q)$ and $GS(Q_i)$ are equal.

Being the isomorphism relation closed under composition, then the composition $\phi_0 \circ \cdots \circ \phi_n$ is an isomorphism and precisely the isomorphism between $GS(Q)$ and $GS(P)$ we wanted.
($\Leftarrow$) Let P and Q pi-processes such that $GS(P) \cong GS(Q)$. We prove the implication by contradiction assuming that $P \not\equiv^{\mathbf{min}} Q$. The proof is by induction on the structure of the processes P and Q.

(Induction base) Let $P = nil$. Since $P \not\equiv^{\mathbf{min}} Q$ then $Q \neq nil$ and obviusly $GS(P) \ncong GS(Q)$. **(Case $P = x(y).R$)** if $Q \neq x(y).S$ then $GS(P) \ncong GS(Q)$ because in Q, by the graph GS construction, does not exists a node with the label and adiacency properties of the node that represent $x(y)$ in P. Otherwise, if $Q = x(y).S$ we have that $R \not\equiv^{\mathbf{min}} S$. By inductive hypothesis we obtain that $GS(R) \ncong GS(S)$ and since for each isomorphism the node that represent $x(y)$ in P should be mapped into the node that represent $x(y)$ in Q, it turns out that a total mapping does not exists and hence $GS(P) \ncong GS(Q)$. **(Case $P = \overline{x}\langle y\rangle.R$ and $P = !\pi.R$)** Similar to the previous case. **(Case $P = R_1 \mid \cdots \mid R_n$)** Let $P = R_1 \mid \cdots \mid R_n$ (we intend all the processes in a form like $(\cdots((R_1 \mid R_2) \mid R_3) \mid \cdots \mid R_n))$ such that R_i is not a parallel composition. If $Q \neq S_1 \mid \cdots \mid S_n$ (with S_i be not a parallel composition) then, by the graph GS construction, $GS(P) \ncong GS(Q)$. Otherwise, we have that $\exists R_i$ such that $\forall S_j$ it is $R_i \not\equiv^{\mathbf{min}} S_j$ and therefore, by inductive hypothesis, $\forall S_j$ it is $GS(R_i) \ncong GS(S_j)$. Since all the subgraphs R_i in P and S_j in Q are disjunct we obtain that $GS(P) \ncong GS(Q)$. **(Case $P = (\nu x_1)\cdots(\nu x_n)R$)** Let $P = (\nu x_1)\cdots(\nu x_n)R$ (with R not in the form $(\nu x)R'$). if $Q \neq (\nu y_1)\cdots(\nu y_n)S$ (with S not in the form $(\nu y)S'$) then, by the graph GS construction, $GS(P) \ncong GS(Q)$. Otherwise, we have that for each permutation of restrictions $(\nu y_1)\cdots(\nu y_n)$ and α-conversion it is $Q = (\nu x_1)\cdots(\nu x_n)T$ with $T \not\equiv^{\mathbf{min}} R$ and thus, by inductive hypothesis, $GS(R) \ncong GS(T)$. Since, by the graph GS construction, the nodes that represents $(\nu x_1)\cdots(\nu x_n)$ should be mapped into the nodes that represents

$(\nu y_1)\cdots(\nu y_n)$ we have that $GS(P) \ncong GS(Q)$.

This contradict the assumption that $GS(P) \cong GS(Q)$ and therefore the implication is valid. □

The lDAG isomorphism problem [10,1] is placed in the complexity class **GI**, which contains all the problems equivalent to the general graph isomorphism problem. The class **GI** is a particular complexity class. In fact, no polinomialy resolution algorithm for the problems in **GI** has been still found and it is not known if they are or not **NP-complete**. However, the congruence $\equiv^{\mathbf{min}}$ is decidable.

Theorem 4.7 *Let P and Q be guarded replication pi-processes. Then the evaluation of $P \equiv^{\mathbf{std}} Q$ is decidable.*

Proof. Using the Lemma 4.4, the Theorem 3.10 in [7] and the Lemma 4.6 we have that

$$\begin{gathered} P \equiv^{\mathbf{std}} Q \\ \Longleftrightarrow \\ Implosion(P) \equiv^{\mathbf{!fr}} Implosion(Q) \\ \Longleftrightarrow \\ swf(Implosion(P)) \equiv^{\mathbf{min}} swf(Implosion(Q)) \\ \Longleftrightarrow \\ GS(swf(Implosion(P))) \cong GS(swf(Implosion(Q))) \end{gathered}$$

and therefore, for transitivity, we can conclude that

$$P \equiv^{\mathbf{std}} Q \Longleftrightarrow GS(swf(Implosion(P))) \cong GS(swf(Implosion(Q)))$$

where $GS(swf(Implosion(P))) \cong GS(swf(Implosion(Q)))$ is a decidable problem. □

Corollary 4.8 *Let $\boldsymbol{B}[P]$ and $\boldsymbol{B}'[P']$ be boxes where P and P' are guarded replication pi-processes. Then the evaluation of $\boldsymbol{B}[P] \equiv_{bb}^{\boldsymbol{min}} \boldsymbol{B}'[P]$ is decidable.*

Proof. Immediate from the definition of the function $BBMinCong$ and the Theorem 4.7. □

Corollary 4.9 *Let B e B' be beta-processes composed by boxes with guarded replication pi-processes. Then the evaluation of $B \equiv_{bb}^{\boldsymbol{std}} B'$ is decidable.*

Proof. Immediate from the definition of the function $BBStdCong$ and the Corollay 4.8. □

5 Generalization

Although we think that the restricted beta binder *well-formedness* definition, presented in Sec.3, gives enough expressive power, in this section we briefly show that

the congruence $\equiv_{bb}^{\mathbf{min}}$ for the stochastic semantics of Beta-binders is decidable also considering the classical *well-formedness* definition, given in Sect.2.

Let $\boldsymbol{B}[P]$ and $\boldsymbol{B}'[P']$ be boxes where P and P' are guarded replication pi-processes. We assume the existence of an injective, decidable and polynomial function $[\![\ ]\!] : \widehat{\beta} \times \mathcal{T} \to \mathcal{S}$ where $\mathcal{T}$ is the set of beta binder types and $\mathcal{S}$ is a set of strings such that $0 \notin \mathcal{S}$ and $\mathcal{S} \cap \mathcal{L} = \emptyset$ (we assume $\mathcal{L}$ be the set of the possible labels generated by the GS construction) . For deciding $\boldsymbol{B}[P] \equiv_{bb}^{\mathbf{min}} \boldsymbol{B}'[P']$ we construct the lDAGs $GS(Q)$ and $GS(Q')$, where $Q = swf(Implosion(P))$ and $Q' = swf(Implosion(P'))$, we interpret the beta binders lists $\boldsymbol{B}$ and $\boldsymbol{B}'$ as a set of top level restrictions and we put them on the top of the constructed lDAGs, modifying the binded nodes as described in Def.4.5. The only difference is that a node that represents an elementary beta binder $\widehat{\beta}(x : \Gamma)$ is labelled with the result of the function $[\![\widehat{\beta}, \Gamma]\!]$ instead of 0. We call the obtained graphs $\overline{GS}(\boldsymbol{B}[Q])$ and $\overline{GS}(\boldsymbol{B}'[Q'])$. In Fig.10 an example is given.

The $\overline{GS}$ graphs can be built in polynomial time and since the graphs $\overline{GS}(\boldsymbol{B}[Q])$ and $\overline{GS}(\boldsymbol{B}'[Q'])$ differ from $GS(Q)$ and $GS(Q')$ only in the number and labels of nodes that represent restrictions, the Lemma 4.6 continues to hold and thus we have that:

Corollary 5.1 *Let $\boldsymbol{B}[P]$ and $\boldsymbol{B}'[P']$ be boxes where P and P' are guarded replication pi-processes. Then $\boldsymbol{B}[P] \equiv_{bb}^{\mathbf{min}} \boldsymbol{B}'[P']$ iff $\overline{GS}(\mathbf{B}[Q]) \cong \overline{GS}(\mathbf{B}'[Q'])$, where $Q = swf(Implosion(P))$ and $Q' = swf(Implosion(P'))$.*

The function $BBStdCong$ and the Corollaries 4.8 and 4.9 can be simply redefined considering the graph $\overline{GS}$ construction.

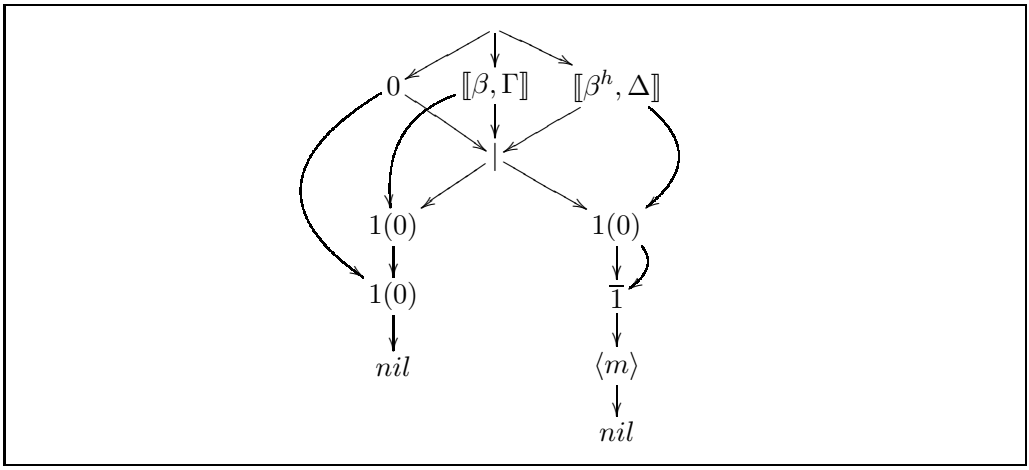

Fig. 10. lDAG $\overline{GS}$ for the box $\beta(x : \Gamma)\beta^h(y : \Delta)[(\nu z)(x(a).z(a).nil \mid y(b).\overline{b}\langle m \rangle.nil)]$.

6 Conclusions

We proved the decidability of the structural congruence used in [3] to define the stochastic semantics of Beta-binders. The proof is constructive so that we have suggestions for possible implementations of the calculus.

References

[1] K.S. Booth and C.J. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-04, Department of Computer Science, University of Waterloo, Ontario, Canada, 1977.

[2] L. Cardelli. Brane calculi. In *Proc. of Computational Methods in Systems Biology*, volume 3082 of *LNCS*, pages 257–278, 2005.

[3] P. Degano, D. Prandi, C. Priami, and P. Quaglia. Beta-binders for biological quantitative experiments. In *4rd Int. Workshop on Quantitative Aspects of Programming Languages (QAPL 06)*, 2006. to appear.

[4] J. Engelfriet. A multiset semantics for the pi-calculus with replication. *Theor. Comput. Sci.*, 153(1&2):65–94, 1996.

[5] J. Engelfriet and T.E. Gelsema. Multisets and structural congruence of the pi-calculus with replication. *Theor. Comput. Sci.*, 211(1-2):311–337, 1999.

[6] J. Engelfriet and T.E. Gelsema. Structural congruence in the pi-calculus with potential replication. Technical Report 00-02, Leiden Institute of Advanced Computer Science, Leiden University, The Nederlands, 2000.

[7] J. Engelfriet and T.E. Gelsema. The decidability of structural congruence for replication restricted pi-calculus processes. Technical Report 04-07, Leiden Institute of Advanced Computer Science, Leiden University, The Nederlands, 2004.

[8] J. Engelfriet and T.E. Gelsema. A new natural structural congruence in the pi-calculus with replication. *Acta Inf.*, 40(6-7):385–430, 2004.

[9] D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Phys. Chem.*, 22:403–434, 1976.

[10] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: its structural complexity.* Birkhäuser, 1993.

[11] R. Milner. The polyadic pi-calculus: a tutorial. Technical Report ECSLFCS-91-180, Computer Science Department, University of Edinburgh, UK, 1991.

[12] C. Priami and P. Quaglia. Beta binders for biological interactions. In *Proc. of Computational Methods in Systems Biology*, volume 3082 of *LNCS*, pages 20–33, 2005.

[13] C. Priami and P. Quaglia. Operational patterns in beta-binders. *T. Comp. Sys. Biology*, 1:50–65, 2005.

[14] C. Priami, A. Regev, E. Shapiro, and W. Silvermann. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.*, 80(1):25–31, 2001.

[15] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. Bioambients: an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1):141–167, 2004.

[16] A. Regev and E. Shapiro. Cells as computation. *Nature*, 2002.