Symbolic Weak Bisimulation for Value-passing Calculi *

Hee-Hwan Kwak, Jin-Young Choi[†], Insup Lee, Anna Philippou
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{heekwak,annap}@saul.cis.upenn.edu, lee@cis.upenn.edu
[†] Department of Computer Science and Engineering
Korea University, Seoul, Korea
choi@formal.korea.ac.kr

Abstract

Bisimulation equivalence has proved to be a useful notion for providing semantics to process calculi, and it has been an object of rigorous study within the concurrency theory community. Recently there have been significant efforts for extending bisimulation techniques to encompass process calculi that allow the communication of data, that is value-passing calculi. A main challenge in achieving this has been the fact that in value-passing processes, where the values are from an infinite domain, checking for process equivalence involves comparing an infinite number of behaviors. Central among work in this area is the approach in which a new notion of process semantics, namely *symbolic semantics*, is proposed as a means of expressing a class of value-passing processes as finite graphs, although their traditional underlying graphs are infinite. This paper presents new algorithms for computing symbolic weak bisimulation for value-passing processes. These algorithms are built on the basis of a new formulation of late weak bisimulation which we propose. Unlike the traditional definition, this formulation allows the use of a single symbolic weak transition relation for expressing both early and late bisimulations. In addition to being easy to implement, our algorithms highlight the distinction between early and late weak bisimulations. Furthermore, we propose a simple variation of symbolic transition graphs with assignment proposed by Lin, in which the order of assignments and actions in transitions is exchanged. We believe that the resulting graphs are easier to construct and more intuitive to understand.

*This research was supported in part by NSF CCR-9415346, NSF CCR-9619910, AFOSR F49620-95-1-0508, ARO DAAH04-95-1-0092, and ONR N00014-97-1-0505.

1 Introduction

Bisimulation equivalence [4] has proved to be a useful notion for providing semantics to process calculi, and it has been an object of rigorous study within the concurrency theory community. As a result, today the theory of bisimulation is well understood within the context of traditional process calculi such as CCS, and efficient bisimulation-checking algorithms exist that enable its automatic checking.

Recently there have been significant efforts for extending bisimulation techniques to encompass process calculi that allow the communication of data, that is value-passing calculi [4, 5, 3]. A main challenge in achieving this has been the fact that in value-passing processes, where the values are from an infinite domain, checking for process equivalence involves comparing an infinite number of behaviors. A substantial body of work has concentrated on addressing this challenge and several approaches have been proposed for establishing bisimulation for classes of value-passing processes. Central among them is the approach proposed by [1] where a new notion of process semantics, namely *symbolic semantics*, is proposed as a means of expressing a class of value-passing processes as finite graphs, although their traditional underlying graphs are infinite. Strong bisimulation equivalence was redefined at this level and algorithms for deciding it were presented. The class of processes that can be described by finite symbolic graphs was extended in [3], which generalized the notion of symbolic transition graphs by allowing assignments to take place in transitions, yielding *symbolic transition graphs with assignment*, STGA's in short, and the theory was appropriately extended. We refer to these papers for a detailed exposition. In this setting, bisimulation for symbolic transition graphs is defined in terms of relations parametrized on boolean expressions, of the form $\simeq^b$, where $p \simeq^b q$ if and only if, for each interpretation satisfying boolean b, p and q are bisimilar in the traditional notion. Furthermore, algorithms were presented for computing these equivalences. In particular, given two closed processes whose symbolic transition graphs are finite, the algorithm constructs a predicate equation system that corresponds to the most general condition for the two processes to be bisimilar.

An additional particularity of value-passing processes is the existence of more than one way for providing their semantics: as noted and investigated in the context of the π-calculus [5], a process-calculus where names, or channels are passed as objects of interaction among processes, there are two ways of interpreting input actions. These are often referred to as *early* and *late* semantics, which distinguish on the time when the receipt of a value takes place during an input transition. Each of these semantics, gives rise to the corresponding bisimulation equivalence, and it turns out that *late* bisimulation is strictly finer than *early* bisimulation. The work of [1, 3] deals with both of these equivalences. We should also mention that a similar approach for deciding strong early bisimulation was independently presented in [7] and we refer to these papers for further discussion of research in this field.

This line of work has been the main motivation of this paper, our aim being to extend reasoning about SGA's to handle weak behavioral relations. Definitions of such relations were provided in [6] in the context of the π-calculus and in [8] for value-passing CCS, the latter including symbolic variants. To the best of our knowledge, work involving these equicalences has mainly focused on their characterization [9].

The main contributions of this paper can be summarized as follows: First we develop and present algorithms for computing weak bisimulation for value-passing processes, corresponding to finite SGA's. Secondly, we study in some detail the distinction between early weak and late weak bisimulations. We observe that a close relationship can be established between the two and we present a new formulation of the late weak bisimulation definition that brings forward a new treatment of late bisimulation as a special case of early bisimulation. We use the new formulations as a basis for constructing our algorithms. An advantage of this approach is that the resulting algorithms follow easily from the definitions and the algorithm for late weak bisimulation is a simple extension of the early weak bisimulation one. As in the case of strong bisimulations, our algorithms produce logical formulas that capture the most general condition for two processes to be weakly bisimilar. Finally, in this paper we propose a simple variation of the SGA's proposed in [3], in which the order of assignments and actions in transitions is exchanged. We believe that the resulting graphs are easier to construct and more intuitive to understand. This is discussed in detail in Section 2. Similar graphs are also employed in [7], although no rules were defined for their construction in that paper.

In this paper we focus on value-passing CCS, which we refer to as CCS-VP, as a vehicle of presenting our results. However, the approach is applicable to other value-passing process calculi.

The rest of the paper is organized as follows: In the next section we present some background material on value-passing CCS and we define rules for constructing SGA's. In Section 3, we proceed to consider how operational semantics and ground bisimulations can be defined for SGA's. In Section 4, we define weak symbolic bisimulations and in Section 5 we present algorithms for computing them. We conclude with a summary and some final remarks in Section 6. An appendix contains the proof of a main result.

2 Symbolic Graph with Assignment for CCS-VP

The syntax of value-passing CCS is similar to that of CCS. The difference is that values may be passed along communication channels, and assigned to variables. We assume a set of variables X and let x, y range over X. A process P with free variables $\vec{x} = x_1, \ldots, x_n$ is denoted as $P(\vec{x})$. Furthermore, we use v for value constants ranging over some set V, e for value expressions ranging over some set $Expr$ (including for example arithmetic expressions), and b for boolean expressions ranging over some set $BExpr$, with $Expr \supset BExpr$. We denote a vector of value variables and value expressions as $\vec{x}$ and $\vec{e}$, respectively. Also, we assume a set of labels L and we use c to denote labels and F to represent subsets of L. Furthermore, we use C, X to stand for process constants. The following grammar describes the syntax of CCS-VP processes:

$$\begin{array}{lcl} P & ::= & \mathrm{NIL} \mid \alpha.P \mid P+P \mid P \| P \mid b \rightarrow P \mid P \backslash F \mid \mathsf{C} \\ \alpha & ::= & \tau \mid c?x \mid c!e \\ \mathsf{C} & ::= & \mathsf{X} \mid \mathsf{X}(\vec{e}) \end{array}$$

For each process constant X, we assume a declaration of the form $\mathsf{X}(\vec{x}) \stackrel{\mathrm{def}}{=} P$ where the free variables of P all occur in $\vec{x}$. We use $Act = \{c?x, c!e \mid c \in L\} \cup \{\tau\}$ to denote the domain of actions, and we let α, β range over Act. We use $l(\alpha)$ to represent the label of action α; e.g., $l(c!e) = c!$ and $l(c?x) = c?$. Lastly, given a process P, we write $fv(P)$ for the free variables occurring in P which are defined in the usual way, that is $fv(c!e.P) = fv(e) \cup fv(P)$ and $fv(c?x.P) = fv(P) - \{x\}$.

2.1 Symbolic Transition System

The Symbolic Graph with Assignment (SGA) we propose here has been used in [7]. However, no rules to construct the SGA were presented in that paper. In this Section, we introduce rules to construct the SGA from CCS-VP terms.

The notion of a *substitution*, which we also call *assignment*, is defined as follows. A *substitution* is any function $\theta\colon X \to Expr$, such that $\theta(x) \neq x$ for a finite number of $x \in X$. Given a substitution θ, the *support* (or *domain*) of θ is the set of variables $D(\theta) = \{x \,|\, \theta(x) \neq x\}$. A substitution whose support is empty is called the *identity substitution*, and is denoted by Id. When $|D(\theta)| = 1$, we use $[\theta(x)/x]$ for the substitution θ. Given two substitutions θ and σ, the *composition* of θ and σ is the substitution denoted by $\theta;\sigma$ such that for every variable x, $\theta;\sigma(x) = \sigma(\theta(x))$. $\theta\sigma$ for $\theta;\sigma$.

An SGA is a rooted directed graph where each node n has an associated finite set of free variables $fv(n)$ and each edge is labeled by a guarded action with assignment [3, 8]. Note that a node in SGA is a CCS-VP term. Furthermore, we use $\sqcup$ to denote the empty action the purpose of which is explained later.

Definition 2.1 (SGA) A Symbolic Graph with Assignment (SGA) for CCS-VP is a rooted directed graph where each node n has an associated CCS-VP term and each edge is labeled by boolean, action, assignment, (b, α, θ) or by boolean, the empty action and assignment, $(b, \sqcup, \theta)$. □

$$\textbf{(1)}\quad \frac{-}{\alpha.P \overset{true,\alpha,\mathsf{Id}}{\longmapsto} P} \qquad\qquad \textbf{(2)}\quad \frac{\mathsf{C}(\vec{e}) \overset{b,\sqcup,\theta}{\longmapsto} \mathsf{C}}{\alpha.\mathsf{C}(\vec{e}) \overset{b,\alpha,\theta}{\longmapsto} \mathsf{C}}$$

$$\textbf{(3)}\quad \frac{-}{\mathsf{C}(\vec{e}) \overset{true,\sqcup,\vec{x}:=\vec{e}}{\longmapsto} \mathsf{C}} \quad \mathsf{C}(\vec{x}) \overset{\text{def}}{=} P \qquad \textbf{(4)}\quad \frac{P \overset{b,\alpha,\theta}{\longmapsto} P'}{\mathsf{C} \overset{b,\alpha,\theta}{\longmapsto} P'} \quad \mathsf{C}(\vec{x}) \overset{\text{def}}{=} P$$

$$\textbf{(5)}\quad \frac{P \overset{b,\alpha,\theta}{\longmapsto} P'}{b' \to P \overset{b\wedge b',\alpha,\theta}{\longmapsto} P'}$$

$$\textbf{(6)}\quad \frac{P \overset{b,\alpha,\theta}{\longmapsto} P'}{P \;+\; Q \overset{b,\alpha,\theta}{\longmapsto} P'} \qquad\qquad \textbf{(7)}\quad \frac{P \overset{b,\alpha,\theta}{\longmapsto} P'}{Q \;+\; P \overset{b,\alpha,\theta}{\longmapsto} P'}$$

$$\textbf{(8)}\quad \frac{P \overset{b,\alpha,\theta}{\longmapsto} P'}{P\backslash F \overset{b,\alpha,\theta}{\longmapsto} P'\backslash F} \qquad \begin{array}{l} \tau \notin F \\ l(\alpha) \notin F \end{array}$$

$$\textbf{(9)}\quad \frac{P \overset{b,\alpha,\vec{x}:=\vec{e}}{\longmapsto} P'}{P\|Q \overset{b,\alpha,\vec{x},\vec{y}:=\vec{e},\vec{y}}{\longmapsto} P'\|Q} \quad fv(Q) = \{\vec{y}\} \qquad \textbf{(10)}\quad \frac{P \overset{b,\alpha,\vec{x}:=\vec{e}}{\longmapsto} P'}{Q\|P \overset{b,\alpha,\vec{x},\vec{y}:=\vec{e},\vec{y}}{\longmapsto} Q\|P'} \quad fv(Q) = \{\vec{y}\}$$

$$\textbf{(11)}\quad \frac{P \overset{b_1,c?z,\theta_1}{\longmapsto} P' \qquad Q \overset{b_2,c!e,\theta_2}{\longmapsto} Q'}{P\|Q \overset{b_1\wedge b_2,\tau,(\theta_1\cup\theta_2);\{z:=e\}}{\longmapsto} P'\|Q'} \quad z \notin fv(P) \cup fv(Q)$$

Figure 1: Rules to construct Symbolic Graph with Assignment

Given an CCS-VP term, a SGA can be generated using the rules in Figure 1. Note that the purpose of action $\sqcup$ is to decorate transitions that involve no action, but are nonetheless necessary for registering substitutions, see Rule (3). Transition $P \overset{b,\alpha,\theta}{\longmapsto} P'$ denotes that given the truth of boolean expression b, P can evolve to P' by performing actions α and putting into effect the assignment θ. The interpretation of these rules is straightforward and we explain them by an example: Consider the following process. Process $P(1)$ can output $a!1$ infinitely many times.

$$\begin{array}{lcl} P(x) & = & (x<2) \rightarrow a!1.P'(x+1) \\ P'(x) & = & (x<3) \rightarrow P(1) \end{array}$$

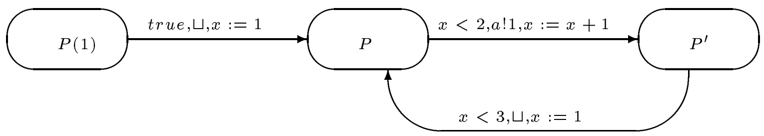
By applying the rules above we can obtain the transitions of process $P(1)$ as follows:

$$\frac{-}{P(1) \overset{true,\sqcup,x:=1}{\longmapsto} P}\ (Rule(3))$$

$$\dfrac{\dfrac{\dfrac{\dfrac{-}{P'(x+1) \overset{true,\sqcup,x:=x+1}{\longmapsto} P'}\ (Rule(3))}{a!1.P'(x+1) \overset{true,a!1,x:=x+1}{\longmapsto} P'}\ (Rule(2))}{(x<2) \rightarrow a!1.P'(x+1) \overset{true\wedge x<2,a!1,x:=x+1}{\longmapsto} P'}\ (Rule(5))}{P \overset{x<2,a!1,x:=x+1}{\longmapsto} P'}\ (Rule(4))$$

$$\dfrac{\dfrac{P(1) \overset{true,\sqcup,x:=1}{\longmapsto} P}{(x<3) \rightarrow P(1) \overset{true\wedge x<3,\sqcup,x:=1}{\longmapsto} P}\ (Rule(5))}{P' \overset{x<3,\sqcup,x:=1}{\longmapsto} P}\ (Rule(4)\ and\ true \wedge x<3 = x<3)$$

This gives rise to the following SGA:



One possible interpretation of our SGA can be given along the lines of programming languages: Process P can be thought of as a procedure, so that $P(1)$ represents a call of P with actual parameter 1 which is accepted by P with formal parameter x declared in P's body. According to its definition, P checks if $x < 2$ and if this boolean expression holds, P outputs $a!1$ and calls process P' with actual parameter $x + 1$. Process P' then checks the validity of condition $x < 3$. If this is satisfied, process P' calls P with actual parameter 1. We believe that this interpretation, being similar to that of function calls and parameter passing in programming languages, is an intuitive way of interpreting the CCS-VP terms.

Although introduction of the empty action $\sqcup$ appears useful in constructing SGA's from process terms we would like to be able to construct symbolic graphs without this type of action. We achieve this by introducing function $\mathcal{N}$, the normalization function, which, whenever applied to an SGA G, it introduces to G new transitions as defined in Figure 2. It is not difficult to see that $\mathcal{N}$ has a (unique) fixed point such that $\mathcal{N}(\mathcal{G}) = \mathcal{G}$. On reaching this graph it is possible to remove all empty actions to obtain a *normalized SGA* (NSGA). From this point on we restrict our attention to NSGA's.

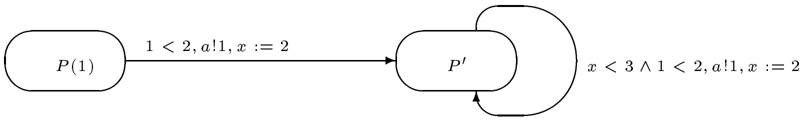
So given the above SGA, by applying the normalization process described above the following NSGA is obtained.

$$\frac{l \overset{b_1,\sqcup,\theta_1}{\longmapsto} m \qquad m \overset{b_2,\alpha,\theta_2}{\longmapsto} n}{l \overset{b_1 \wedge b_2\theta_1,\alpha\theta_1,\theta_2;\theta_1}{\longmapsto} n} \qquad (l \neq m, \alpha \equiv \tau \text{ or } c!e)$$

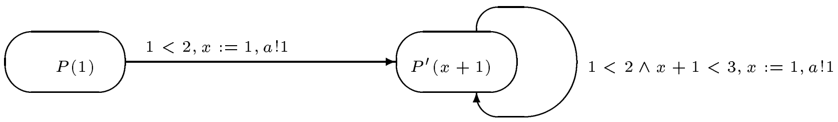
$$\frac{l \overset{b_1,\sqcup,\theta_1}{\longmapsto} m \qquad m \overset{b_2,c?x,\theta_2}{\longmapsto} n}{l \overset{b_1 \wedge b_2\theta_1,c?y,\theta_2[y/x];\theta_1}{\longmapsto} n} \qquad (l \neq m, y \text{ is a fresh variable})$$

$$\frac{l \overset{b_1,\sqcup,\theta_1}{\longmapsto} m \qquad m \overset{b_2,\sqcup,\theta_2}{\longmapsto} n}{l \overset{b_1 \wedge b_2\theta_1,\sqcup,\theta_2;\theta_1}{\longmapsto} n} \qquad (l \neq m \neq n)$$

Figure 2: Rules to normalize Symbolic Graph with Assignment



Before concluding this section we briefly comment on the relation between our SGA's and the symbolic transition graphs with assignment (STGA) presented in [3]. Here follows the STGA corresponding to process $P(1)$ as generated by the rules of that paper.



Both symbolic graphs represent the same behavior. However, our NSGA seems more intuitive than Lin's STGA. In particular, our definition follows an approach which closely resembles function calls and parameter passing in programming style. An additional advantage of viewing process constants as procedures and performing all evaluations within transitions is that it enables the construction of more compact symbolic graphs.

We say that NSGA$(N, E, \mapsto)$ is finite if $|N|$ is finite. For the remainder of the paper we only consider finite NSGA's.

3 Operational Semantics

The definitions of strong and weak bisimulation equivalence is based on ground terms. Since the nodes of a NSGA contain open terms, we need to define the ground operational semantics relative to a substitution and an an environment. An assignment in a transition plays a role of a substitution. Given a finite set V of variables, an environment $\rho_V \colon V \to Val$ is a function, and if it is clear from the context, we omit the subscript V. We define $dom(\rho)$ as V. Given an open term t and an environment ρ, we write (t, ρ) for the term $t\rho$ which is the term obtained by replacing every variable x by a value $\rho(x)$. We say that (t, ρ) is ground if $fv(t) \subseteq dom(\rho)$, that is, $t\rho$ is ground. A (t, ρ) is said to be a *state* if (t, ρ) is ground, that is, $t\rho$ is ground. Given an environment ρ and value v, we write $\rho[x \to v]$ as a new function ρ' such that $\rho'(y) = v$ if $y \equiv x$, $\rho'(y) = \rho(y)$, otherwise.

The late operational semantics are given in Figure 3. We assume that (m_σ, ρ) is ground.

Given a boolean expression b, $\rho \models b$ denotes that $b\rho$ is true.

$$\frac{m \overset{b,\tau,\theta}{\longmapsto} n}{(m_\sigma, \rho) \overset{\tau}{\longrightarrow} (n_{\theta\sigma}, \rho)} \quad \rho \models b\sigma$$

$$\frac{m \overset{b,c!e,\theta}{\longmapsto} n}{(m_\sigma, \rho) \overset{c!(e\sigma)\rho}{\longrightarrow} (n_{\theta\sigma}, \rho)} \quad \rho \models b\sigma$$

$$\frac{m \overset{b,c?x,\theta}{\longmapsto} n}{(m_\sigma, \rho) \overset{c?y}{\longrightarrow} (n_{\theta[y/x]\sigma}, \rho)} \quad \rho \models b\sigma, y \notin fv(m_\sigma)$$

Figure 3: Ground Operational Semantics

The sequence of transitions from $P(1)$ in the example of Section 2 is as follows: Since $(x < 3 \wedge 1 < 2)[2/x]$ is true,

$$(P(1), \emptyset) \overset{a!1}{\longrightarrow} (P'_{[2/x]}, \emptyset) \overset{a!1}{\longrightarrow} (P'_{[2/x][2/x]}, \emptyset) \overset{a!1}{\longrightarrow} \cdots$$

3.1 Weak Bisimulation

In this section, we investigate early and late *weak* bisimulations. We propose new formulations of the definitions, which differ from the traditional ones in that they can be built on the basis of a single weak transition relation. This is unlike [8] where two weak transition relations are defined for each of the semantics. We prove that the corresponding definitions coincide. We begin by introducing a useful piece of notation and a simple fact.

Definition 3.1 When $p \equiv (m_\sigma, \rho)$, we use $p[x \to v]$ and $p[e/x]$ to denote $(m_\sigma, \rho[x \to v])$, $(m_{\sigma[e/x]}, \rho)$, respectively. $\square$

Lemma 3.2 Given $v \in Val$, $p[v/x] = p[x \to v]$. $\square$

We may now define the weak transition relation we will be using. As usual we define $\hat{\alpha}$ to be ϵ when $\alpha = \tau$ and α otherwise.

Definition 3.3 (Weak Transition Relation) The weak transition relation "$\Rightarrow$" is defined for $\alpha \in Act$ as follows:

1. $p \overset{\varepsilon}{\Longrightarrow} p$
2. $p \overset{\alpha}{\longrightarrow} p'$ implies $p \overset{\hat{\alpha}}{\Longrightarrow} p'$
3. $p \overset{\varepsilon}{\Longrightarrow} \overset{\hat{\alpha}}{\Longrightarrow} p'$ implies $p \overset{\hat{\alpha}}{\Longrightarrow} p'$
4. $p \overset{\hat{\alpha}}{\Longrightarrow} \overset{\varepsilon}{\Longrightarrow} p'$ implies $p \overset{\hat{\alpha}}{\Longrightarrow} p'$ $\square$

The following definition is useful for the new formulation of weak late bisimulation that we propose. Essentially it allows us to associate transitions (as given by the operational semantics) with branches of an NSGA.

Definition 3.4 Given an action $\alpha' \not\equiv \tau$ and an edge $m \overset{be,\alpha',\theta}{\longmapsto} m'$ in a NSGA, we say that a transition $p \overset{\alpha}{\longrightarrow} p'$ is *over* $m \overset{be,\alpha',\theta}{\longmapsto} m'$ if $p \overset{\alpha}{\longrightarrow} p'$ is obtained from the edge by the ground operational semantics in Figure 3. Similarly, we say that $p \overset{\alpha}{\Longrightarrow} p'$ is *over* $m \overset{be,\alpha',\theta}{\longmapsto} m'$ if there exist p_1 and p_2 such that $p \overset{\varepsilon}{\Longrightarrow} p_1 \overset{\alpha}{\longrightarrow} p_2 \overset{\varepsilon}{\Longrightarrow} p'$ and $p_1 \overset{\alpha}{\longrightarrow} p_2$ is *over* $m \overset{be,\alpha',\theta}{\longmapsto} m'$ $\square$

We may now propose the following late weak bisimulation definition:

Definition 3.5 (Late Weak Bisimulation) A late weak bisimulation is a symmetric relation R over states such that, if $(p, q) \in R$ then,

- whenever $p \xrightarrow{c?x} p'$ then there exists $n_1 \overset{b,c?z,\theta}{\longmapsto} n_2$ such that for all $v \in Val$, there is a q' with $q[y \to v] \overset{c?y}{\Longrightarrow} q'$ over $n_1 \overset{b,c?z,\theta}{\longmapsto} n_2$ and $(p'[x \to v], q') \in R$,
- whenever $p \xrightarrow{\alpha} p'$ $(l(\alpha) \neq c?)$ then $q \overset{\hat{\alpha}}{\Longrightarrow} q'$ for some q' such that $(p', q') \in R$.

We write $p \approx_L q$ if there exists a late weak bisimulation R such that $(p, q) \in R$. $\square$

The conventional definition of late weak bisimulation is given below, where $\Longrightarrow_1$ denotes the weak symbolic transition relation of [8]. The definition of $\Longrightarrow_1$ is given almost identically to Definition 3.3 with the addition of the condition "$l(\alpha) \neq a?$" to clause (4). Note that as a result of this modification, $t \overset{c?x}{\Longrightarrow}_1 t'$ implies $t \overset{\varepsilon}{\Longrightarrow}_1 \overset{c?x}{\longrightarrow}_1 t'$.

Definition 3.6 (Late Weak Bisimulation [8]) A late weak bisimulation is a symmetric relation R over states such that if $(p, q) \in R$ then

- whenever $p \xrightarrow{c?x} p'$ then $q \overset{c?y}{\Longrightarrow}_1 q''$ for some y such that for all $v \in Val$, there is a q' such that $q''[y \to v] \overset{\varepsilon}{\Longrightarrow}_1 q'$ and $(p'[x \to v], q') \in R$,
- whenever $p \xrightarrow{\alpha} p'$ $(l(\alpha) \neq c?)$ then $q \overset{\hat{\alpha}}{\Longrightarrow}_1 q'$ for some q' such that $(p', q') \in R$.

We write $p \approx_1 q$ if there exists a late weak bisimulation R such that $(p, q) \in R$. $\square$

As stated earlier the two definitions coincide. First we present a useful lemma:

Lemma 3.7 Given an environment ρ, a variable y, and a term m_σ, if $y \notin dom(\rho) \cup fv(m_\sigma)$, then $(m_\sigma, \rho) \overset{\alpha}{\Longrightarrow}_1 (m'_{\sigma'}, \rho)$ if and only if $(m_\sigma, \rho[y \to v]) \overset{\alpha}{\Longrightarrow} (m'_{\sigma'}, \rho[y \to v])$. $\square$

We now have the main theorem of this section.

Theorem 3.8 $p \approx_L q$ iff $p \approx_1 q$.

PROOF: $p \approx_1 q$

iff whenever $p \xrightarrow{c?x} p'$ then there exists $q \overset{c?y}{\Longrightarrow}_1 q''$ for some y such that for each $v \in Val$, there is a q' such that $q''[y \to v] \overset{\varepsilon}{\Longrightarrow}_1 q'$ and $(p'[x \to v], q') \in R$

iff whenever $p \xrightarrow{c?x} p'$ then there exists $q \overset{c?y}{\Longrightarrow}_1 q''$ for some y such that for each $v \in Val$, there is a q' such that $q \overset{c?y}{\Longrightarrow}_1 q''$ and $q''[y \to v] \overset{\varepsilon}{\Longrightarrow}_1 q'$ and $(p'[x \to v], q') \in R$

iff let (n_σ, ϱ) be q. Since $y \notin fv(\varrho) \cup fv(n_\sigma)$ we have the following: whenever $p \xrightarrow{c?x} p'$ then there exists $q \overset{c?y}{\Longrightarrow}_1 q''$ for some y such that for each $v \in Val$, there is a q_1 such that $q[y \to v] \overset{c?y}{\Longrightarrow} q''[y \to v]$ (by Lemma 3.7) and $q''[y \to v] \overset{\varepsilon}{\Longrightarrow} q_1$ and $(p'[x \to v], q_1) \in R$

iff whenever $p \xrightarrow{c?x} p'$ then there exists $q \overset{c?y}{\Longrightarrow}_1 q''$ for some y such that for each $v \in Val$, there is a q_1 such that $q[y \to v] \overset{c?y}{\Longrightarrow} q_1$ and $(p'[x \to v], q_1) \in R$

iff whenever $p \xrightarrow{c?x} p'$ then there exists $n_1 \overset{be,c?y,\theta}{\longmapsto} n_2$ and for each $v \in Val$, there is a q_1 such that $q[y \to v] \overset{c?y}{\Longrightarrow} q_1$ over $n_1 \overset{be,c?y,\theta}{\longmapsto} n_2$ and $(p'[x \to v], q_1) \in R$

iff $p \approx_L q$. $\square$

It is easy to see that early weak bisimulation can be given the following characterization, along lines similar to Definition 3.5:

Definition 3.9 (Early Weak Bisimulation) An early weak bisimulation is a symmetric relation R over states such that, if $(p,q) \in R$ then,

- whenever $p \xrightarrow{c?x} p'$ then for each $v \in Val$, there exist $n_1 \overset{b,c?z,\theta}{\longmapsto} n_2$ and a q' such that $q[y \to v] \overset{c?y}{\Longrightarrow} q'$ over $n_1 \overset{b,c?z,\theta}{\longmapsto} n_2$ and $(p'[x \to v], q') \in R$,
- whenever $p \xrightarrow{\alpha} p'$ $(l(\alpha) \neq c?)$ then $q \overset{\hat{\alpha}}{\Longrightarrow} q'$ for some q' such that $(p', q') \in R$.

We write $p \approx_E q$ if there exists an early weak bisimulation R such that $(p,q) \in R$. $\square$

We may see that the first condition of the definition above can be rewritten as

- whenever $p \xrightarrow{c?x} p'$ then for each $v \in Val$, there is a q' such that $q[y \to v] \overset{c?y}{\Longrightarrow} q'$ and $(p'[x \to v], q') \in R$,

thus yielding the standard early bisimulation definition.

We say that a ground term t is *input-deterministic* if there exist no n in t's SGA with more than one, distinct, outgoing edge using the same input channel, that is, if $n \overset{b,c?x,\theta}{\longmapsto} n_1$ and $n \overset{b',c?x',\theta'}{\longmapsto} n_2$ then $n_1 = n_2$, $b = b'$ and $\theta = \theta'$. Similarly, we can define when a ground term is *weakly input-deterministic*. We conclude this section with the observation that early and late bisimulations coincide for input-deterministic terms, where $\sim_L$ and $\sim_E$ denote the standard late and early strong bisimulations respectively.

Lemma 3.10 Given two ground terms s and t,

1. if s and t are input-deterministic, then $s \sim_L t$ iff $s \sim_E t$.
2. if s and t are weakly input-deterministic, then $s \approx_L t$ iff $s \approx_E t$.

$\square$

4 Symbolic Operational Semantics

In this section we define symbolic operational semantics, and symbolic early and late weak bisimulations. The symbolic operational semantics are given in Figure 4. Note that m_σ need not be ground in the symbolic operational semantics. This semantics is referred to as *symbolic* because no evaluation of expressions is carried out, instead evaluations are carried symbolically along transitions with the aid of substitutions.

Symbolic strong early and late bisimulations, $\sim_E^b$, and $\sim_E^b$, can be defined for NSGA's as is done for STGA's in [3]. We omit their definition but we point out that they are well-behaved, in the sense that they capture the corresponding ground strong bisimulations, $\sim_L$ and $\sim_E$:

Theorem 4.1 1) $m_\sigma \sim_L^b n_\mu$ iff $(m_\sigma, \rho) \sim_L (n_\mu, \rho)$ for every ρ such that $\rho \models b$. 2) $m_\sigma \sim_E^b n_\mu$ iff $(m_\sigma, \rho) \sim_E (n_\mu, \rho)$ for every ρ such that $\rho \models b$
PROOF: Similar to [1] $\square$

$$\frac{m \overset{b,\tau,\theta}{\longmapsto} n}{m_\sigma \overset{b\sigma,\tau}{\longrightarrow} n_{\theta\sigma}}$$

$$\frac{m \overset{b,c!e,\theta}{\longmapsto} n}{m_\sigma \overset{b\sigma,c!e\sigma}{\longrightarrow} n_{\theta\sigma}}$$

$$\frac{m \overset{b,c?x,\theta}{\longmapsto} n}{m_\sigma \overset{b\sigma,c?z}{\longrightarrow} n_{\theta\sigma[z/x]}} \qquad z \notin fv(m_\sigma)$$

Figure 4: Symbolic Operational Semantics

4.1 Symbolic Weak Bisimulation

We now define symbolic late and early weak bisimulations. As with the ground case, this is achieved via a single symbolic weak transition relation, as opposed to the two relations of [8]. We begin with the definition of our symbolic weak transition relation, and then define symbolic late weak bisimulation.

Definition 4.2 (Symbolic Weak Transition Relation) We define the symbolic version of the weak transition relation "$\Rightarrow$" as follows:

1. $m_\sigma \overset{true,\varepsilon}{\Longrightarrow} m_\sigma$
2. $m_\sigma \overset{b,\alpha}{\longrightarrow} n_{\sigma'}$ implies $m_\sigma \overset{b,\hat{\alpha}}{\Longrightarrow} n_{\sigma'}$
3. $m_\sigma \overset{b,\varepsilon}{\Longrightarrow} \overset{b',\hat{\alpha}}{\Longrightarrow} n_{\sigma'}$ implies $m_\sigma \overset{b\wedge b',\hat{\alpha}}{\Longrightarrow} n_{\sigma'}$
4. $m_\sigma \overset{b,\hat{\alpha}}{\Longrightarrow} \overset{b',\varepsilon}{\Longrightarrow} n_{\sigma'}$ implies $m_\sigma \overset{b\wedge b',\hat{\alpha}}{\Longrightarrow} n_{\sigma'}$ □

We write $\Longrightarrow_2$ for the symbolic weak transition relation defined in [8]. This can be obtained from the previous definition by the addition of condition "$l(\alpha) \neq a?$" to clause (4).

Definition 4.3 (Symbolic Late Weak Bisimulation) Let $\mathbf{S} = \{S^b\}$ be a boolean indexed family of symmetric relations. Define $\mathcal{SWB}(\mathbf{S})$ to be the family of relations such that $(m_\sigma, n_\mu) \in \mathcal{SWB}(\mathbf{S})^b$ if and only if

- whenever $m_\sigma \overset{b_1,a?x}{\longrightarrow} m'_{\sigma'}$ there exists a variable z such that $z \notin fv(b, m_\sigma, n_\mu)$ and a $b \wedge b_1$-partition B such that for each $b' \in B$, $z \notin fv(b')$ and there exists a $n_1 \overset{be,a?y',\theta}{\longmapsto} n_2$ and a b'-partition B', and for each $b'' \in B'$, there exists a $(n_\mu \overset{b_2,a?y}{\Longrightarrow} n'_{\mu'})$ over $n_1 \overset{be,a?y',\theta}{\longmapsto} n_2$ such that $b'' \models b_2[z/y]$ and $(m'_{\sigma'[z/x]}, n'_{\mu'[z/y]}) \in S^{b''}$,
- whenever $m_\sigma \overset{b_1,\alpha}{\longrightarrow} m'_{\sigma'}$ there exists a variable z such that $z \notin fv(b, m_\sigma, n_\mu)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $n_\mu \overset{b_2,\widehat{\beta}}{\Longrightarrow} n'_{\mu'}$ such that $b' \models b_2$ and
  - if $\alpha \equiv \tau$ then $\beta \equiv \tau$ and $(m'_{\sigma'}, n'_{\mu'}) \in S^{b'}$,
  - if $\alpha \equiv c!e$ then $\beta \equiv c!e'$ with $b' \models e = e'$ and $(m'_{\sigma'}, n'_{\mu'}) \in S^{b'}$.

S^b is a *symbolic late weak bisimulation* if $S^b \subseteq \mathcal{SWB}(\mathbf{S})^b$ for each b. We denote the largest such $\mathbf{S}$ by $\approx^b$. We write $t \approx^b u$ if there is a symbolic late weak bisimulation $\mathbf{S}$ such that $(t, u) \in S^b$. □

We proceed with the definition of late weak bisimulation presented in [8].

Definition 4.4 (Symbolic Late Weak Bisimulation [8]) Let $\mathbf{S} = \{S^b\}$ be a boolean indexed family of symmetric relations. Define $\mathcal{SWB}(\mathbf{S})$ to be the family of relations such that $(t,u) \in \mathcal{SWB}(\mathbf{S})^b$ if and only if

> whenever $t \xrightarrow{b_1,\alpha} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $u \overset{b_2,\widehat{\beta}}{\Longrightarrow}_2 u'$ such that $b' \models b_2$ and
>
> - if $\alpha \equiv \tau$ then $\beta \equiv \tau$ and $(t',u') \in S^{b'}$,
> - if $\alpha \equiv c!e$ then $\beta \equiv c!e'$ with $b'nn \models e = e'$ and $(t',u') \in S^{b'}$,
> - if $\alpha \equiv c?x$ then $\beta \equiv c?y$ for some y and there exists a b'-partition B' such that for each $b'' \in B'$ there is a u'' such that $u'[z/y] \overset{b_2',\varepsilon}{\Longrightarrow}_2 u''$ with $b'' \models b_2'$ and $(t'[z/x],u'') \in S^{b''}$.

S^b is a *symbolic late weak bisimulation* if $S^b \subseteq \mathcal{SWB}(\mathbf{S})^b$ for each b. We denote the largest such $\mathbf{S}$ by $\approx_2^b$. We write $t \approx_2^b u$ if there is a symbolic late weak bisimulation $\mathbf{S}$ such that $(t,u) \in S^b$.
□

We now prove our claim that the two definitions are equivalent to each other. In the proof we make use of the following notation: given $t \equiv m_\sigma$, we write $t[e/x]$ for $m_{\sigma[e/x]}$.

Theorem 4.5 $m_\sigma \approx^b n_\mu$ iff $m_\sigma \approx_2^b n_\mu$ □

The definition of symbolic early weak bisimulation follows easily from the ground definition.

Definition 4.6 (Symbolic Early Weak Bisimulation) Let $\mathbf{S} = \{S^b\}$ be a boolean-indexed family of symmetric relations. Define $\mathcal{SWB}(\mathbf{S})$ to be the family of relations such that $(m_\sigma, n_\mu) \in \mathcal{SWB}(\mathbf{S})^b$ if and only if

- whenever $m_\sigma \xrightarrow{b_1,a?x} m'_{\sigma'}$ there exists a variable z such that $z \notin fv(b,m_\sigma,n_\mu)$ and a $b \wedge b_1$-partition B, for each $b' \in B$, there exists a $n_\mu \overset{b_2,a?y}{\Longrightarrow} n'_{\mu'}$ with $b' \models b_2[z/y]$ and $(m'_{\sigma'[z/x]}, n'_{\mu'[z/y]}) \in S^{b'}$,
- whenever $m_\sigma \xrightarrow{b_1,\alpha} m'_{\sigma'}$ there exists a $b \wedge b_1$-partition B such that for each $b' \in B$ and there exists a $n_\mu \overset{b_2,\widehat{\beta}}{\Longrightarrow} n'_\mu$ such that $b' \models b_2$ and
  - if $\alpha \equiv \tau$ then $\beta \equiv \tau$ and $(m'_{\sigma'}, n'_{\mu'}) \in S^{b'}$,
  - if $\alpha \equiv c!e$ then $\beta \equiv c!e'$ with $b' \models e = e'$ and $(m'_{\sigma'}, n'_{\mu'}) \in S^{b'}$.

S^b is a *symbolic early weak bisimulation* if $S^b \subseteq \mathcal{SWB}(\mathbf{S})^b$ for each b. We denote the largest such $\mathbf{S}$ by $\approx_E^b$. We write $t \approx_E^b u$ if there is a symbolic early weak bisimulation $\mathbf{S}$ such that $(t,u) \in S^b$. □

Note that the fresh variable z may appear in partition B. The next theorem states that the corresponding symbolic and ground weak bisimulations coincide.

Theorem 4.7 1) $m_\sigma \approx^b n_\mu$ iff $(m_\sigma,\rho) \approx_L (n_\mu,\rho)$ for every ρ such that $\rho \models b$. 2) $m_\sigma \approx_E^b n_\mu$ iff $(m_\sigma,\rho) \approx_E (n_\mu,\rho)$ for every ρ such that $\rho \models b$.
PROOF: Similar to [8]. □

5 Algorithms

Once we have two NSGAs rooted with node n_1 and m_1, we want to decide whether or not these two nodes are bisimilar. For late strong bisimulation, one can use the symbolic late strong bisimulation algorithm presented in [3]. It is not too difficult to adapt the symbolic late algorithm to check early strong bisimulation following the approach described in this section for checking early weak bismilation. In this section, we first explain how to describe an algorithm for checking symbolic late weak bisimulation and then show how it can be modified for symbolic early weak bisimulation.

5.1 Symbolic Late Weak Bisimulation

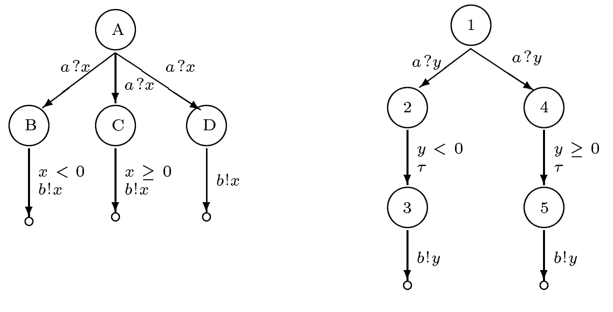
To capture the semantics of symbolic weak bisimulation on the symbolic graph, we define the relation $\rightsquigarrow$ on NSGA as follows.

- $m \overset{true,\tau,\mathsf{Id}}{\rightsquigarrow} m$
- $m \overset{b,\alpha,\theta}{\longmapsto} n$ implies $m \overset{b,\alpha,\theta}{\rightsquigarrow} n$
- $m \overset{b,\tau,\theta}{\rightsquigarrow} \overset{b',\alpha,\theta'}{\rightsquigarrow} n$ implies $m \overset{b\wedge b'\theta,\alpha\theta,\theta';\theta}{\rightsquigarrow} n$
- $m \overset{b,\alpha,\theta}{\rightsquigarrow} \overset{b',\tau,\theta'}{\rightsquigarrow} n$ implies $m \overset{b\wedge b'\theta,\alpha\theta,\theta';\theta}{\rightsquigarrow} n$
- $m \overset{b,\tau,\theta}{\rightsquigarrow} \overset{b',\tau,\theta'}{\rightsquigarrow} n$ implies $m \overset{b\wedge b'\theta,\tau,\theta';\theta}{\rightsquigarrow} n$

The relation $\rightsquigarrow$ essentially captures the symbolic weak transition relations defined in Definition 4.2. We also need to define the notion of *cover* similar to the notion of *over* in Definition 3.4. The difference is that the *over* is applied to transition, whereas *cover* is applied to graph edge.

Definition 5.1 Given an edge $m \overset{b,\alpha',\theta}{\longmapsto} m'$ in a NSGA, we say that an edge $n \overset{b',\alpha',\theta'}{\rightsquigarrow} n'$ *covers* $m \overset{b,\alpha',\theta}{\longmapsto} m'$ if $n \overset{b',\alpha',\theta'}{\rightsquigarrow} n'$ is obtained from the edge. $\square$

Our symbolic late weak bisimulation algorithm is shown in Figure 5. $bisim(\mathcal{G}, \sigma, \mathcal{G}', \sigma')$ takes two root nodes of SGA $\mathcal{G}$ and $\mathcal{G}'$ with initial substitution σ and σ'. The functions $match_\tau$, $match_{c!}$ and $match_{c?}$ are to match τ, output and input transitions, respectively. The result of each function call with n and n' is a predicate equation system which states a condition under which two nodes n and n' are bisimilar. Our algorithm also generates the predicate equation system defined in [3].



Due to space limitation, we illustrate the algorithm using the above example. In the above graphs, if the condition is *true* and an assignment θ is Id, then we omit them from an edge. Instead of presenting the whole equation systems, we only show some fragments of it.

Before applying our bisimulation algorithm, we add edges obtained from relation $\rightsquigarrow$ to NSGA, and then we apply the algorithm in Figure 5 to NSGA with both original edges and new $\rightsquigarrow$ edges. Let us consider the nodes A and 1 with function $match_{c?}$ in Figure 5. We denote the original edge from A to B as $A \mapsto B$. For $A \mapsto B$ takes place, there has to be at least one edge

$bisim\ (\mathcal{G}, \sigma, \mathcal{G}', \sigma')$:

  $Queue = \{n_0, n'_0\}$

  while $Queue \neq \emptyset$

    pop (n, n') from $Queue$

    $X_{n,n'} = match(n, \sigma, n', \sigma')$

    $\sigma = \sigma' = \emptyset$

$match\ (n, \sigma, n', \sigma')$:

  $\bigwedge\limits_{\alpha} match_\alpha(n, \sigma, n', \sigma')$ for all $\alpha \in \{\tau, c?, c!, A\}$

$match_\tau\ (n, \sigma, n', \sigma')$:

  for each $n \xmapsto{b_i,\tau,\theta_i} n_i$ with all $n' \overset{b'_j,\tau,\theta'_j}{\rightsquigarrow} n'_j$

    $\bigwedge\limits_i (b_i\sigma \rightarrow \bigvee\limits_j (b'_j\sigma' \wedge X_{n_i,n'_j}[\theta_i\sigma, \theta'_j\sigma']))$

    and for each i, j, if (n_i, n'_j) is not visited then push (n_i, n'_j) into $Queue$

  $\wedge$

  for each $n' \xmapsto{b'_j,\tau,\theta'_j} n'_j$ with all $n \overset{b_i,\tau,\theta_i}{\rightsquigarrow} n_i$

    $\bigwedge\limits_j (b'_j\sigma' \rightarrow \bigvee\limits_i (b_i\sigma \wedge X_{n_i,n'_j}[\theta_i\sigma, \theta'_j\sigma']))$

    and for each i, j, if (n_i, n'_j) is not visited then push (n_i, n'_j) into $Queue$

$match_{c!}\ (n, \sigma, n', \sigma')$:

  for each $n \xmapsto{b_i,c!e_i,\theta_i} n_i$ with all $n' \overset{b'_j,c!e'_j,\theta'_j}{\rightsquigarrow} n'_j$

    $\bigwedge\limits_i (b_i\sigma \rightarrow \bigvee\limits_j (b'_j\sigma' \wedge e_i\sigma = e'_j\sigma' \wedge X_{n_i,n'_j}[\theta_i\sigma, \theta'_j\sigma']))$

    and for each i, j, if (n_i, n'_j) is not visited then push (n_i, n'_j) into $Queue$

  $\wedge$

  for each $n' \xmapsto{b'_j,c!e'_j,\theta'_j} n'_j$ with all $n \overset{b_i,c!e_i,\theta_i}{\rightsquigarrow} n_i$

    $\bigwedge\limits_j (b'_j\sigma' \rightarrow \bigvee\limits_i (b_i\sigma \wedge e_i\sigma = e'_j\sigma' \wedge X_{n_i,n'_j}[\theta_i\sigma, \theta'_j\sigma']))$

    and for each i, j, if (n_i, n'_j) is not visited then push (n_i, n'_j) into $Queue$

$match_{c?}\ (n, \sigma, n', \sigma')$:

  for each $n \xmapsto{b_i,c?x,\theta_i} n_i$ with all $m \xmapsto{b'_j,c?y,\theta'_j} m_j$

    and $n' \overset{b''_k,c?y,\theta''_k}{\rightsquigarrow} n''_k$ that cover $m \xmapsto{b'_j,c?y,\theta'_j} m_j$

      $\bigwedge\limits_i (b_i\sigma \rightarrow \bigvee\limits_j (\forall z (\bigvee\limits_k (b''_k\sigma'[y \mapsto z] \wedge X_{n_i,n''_k}[\theta_i\sigma[x \mapsto z], \theta''_k\sigma'[y \mapsto z]]))))$

      and for each i, j, k, if (n_i, n''_k) is not visited then push (n_i, n''_k) into $Queue$

  $\wedge$

  for each $n' \xmapsto{b'_j,c?y,\theta'_j} n'_j$ with all $m \xmapsto{b_i,c?x,\theta_i} m_i$

    and $n \overset{b''_k,c?y,\theta''_k}{\rightsquigarrow} n''_k$ that cover $m \xmapsto{b_i,c?x,\theta_i} m_i$

      $\bigwedge\limits_j (b'_j\sigma' \rightarrow \bigvee\limits_i (\forall z (\bigvee\limits_k (b''_k\sigma[x \mapsto z] \wedge X_{n''_k,n'_j}[\theta''_k\sigma[x \mapsto z], \theta'_j\sigma'[y \mapsto z]]))))$

      and for each i, j, k, if (n''_k, n'_j) is not visited then push (n''_k, n'_j) into $Queue$

Figure 5: Symbolic Late Weak Bisimulation Algorithm

from node 1 that matches the action to make A and 1 be bisimilar. We first pick the edge $1 \mapsto 2$ and find all the transitions that cover the edge $1 \mapsto 2$; they are edges $1 \leadsto 2$ and $1 \leadsto 3$. We then match them with the edge $A \mapsto B$. The result is the boolean equation

$$true \rightarrow \forall z_1(true \wedge X_{B2} \vee z_1 < 0 \wedge X_{B3}).$$

The predicate variables such as X_{B2} are the abbreviation of $X_{B2}[z_1/x, z_1/y]$.

Since there is another nondeterministic choice from node 1 that can be matched with the edge $A \mapsto B$, we consider the edge $1 \mapsto 4$ and also all the edges that cover $1 \mapsto 4$; they are edges $1 \leadsto 4$ and $1 \leadsto 5$. Along with previous boolean expression, this gives the boolean expression

$$true \rightarrow (\forall z_1(true \wedge X_{B2} \vee z_1 < 0 \wedge X_{B3}) \vee \forall z_2(true \wedge X_{B4} \vee z_2 \geq 0 \wedge X_{B5})).$$

So we have a boolean expression for the edge $A \mapsto B$ which is the first line of the predicate equation system for X_{A1} shown below. For the edges $A \mapsto C$ and $A \mapsto D$, we can apply the same procedure as the edge $A \mapsto B$.

If we continue with the edges $A \mapsto C$ and $A \mapsto D$, we eventually get the following fragment of predicate equation system from our symbolic late weak bisimualtion algorithm. Notice that symmetric cases are omitted.

$$\begin{aligned} X_{A1} = {} & true \rightarrow (\forall z_1\ (true \wedge X_{B2} \vee z_1 < 0 \wedge X_{B3}) \vee \forall z_2\ (true \wedge X_{B4} \vee z_2 \geq 0 \wedge X_{B5})) \\ & \wedge\ true \rightarrow (\forall z_3\ (true \wedge X_{C2} \vee z_3 < 0 \wedge X_{C3}) \vee \forall z_4\ (true \wedge X_{C4} \vee z_4 \geq 0 \wedge X_{C5})) \\ & \wedge\ true \rightarrow (\forall z_5\ (true \wedge X_{D2} \vee z_5 < 0 \wedge X_{D3}) \vee \forall z_6\ (true \wedge X_{D4} \vee z_6 \geq 0 \wedge X_{D5})) \end{aligned}$$

X_{B2} is true for all z_1 and X_{B4} is also true for all z_2. Thus, the expression in the first line of X_{A1} is true. Consider the boolean expression in the third line. To decide the bisimilarity with D and 2, if we only concern output actions $b!x$ and $b!y$ for simplicity (i.e., ignoring τ edge), then we get the equation $X_{D2} = (true \rightarrow y < 0) \wedge (y < 0 \rightarrow true)$. So, $X_{D2}[z_5/x, z_5/y]$ only holds when $z_5 < 0$, and $true \wedge X_{D2}$ only holds when $z_5 < 0$, and $z_5 < 0 \wedge X_{D3}$ is also only true when $z_5 < 0$. We can conclude that $\forall z_5(true \wedge X_{D2} \vee z_5 < 0 \wedge X_{D3})$ is false. From the similar reasoning, $\forall z_6(true \wedge X_{D4} \vee z_6 \geq 0 \wedge X_{D5})$ is also false. So, the boolean expression in the third line of X_{A1} is false. We thus conclude that X_{A1} does not hold, which implies that the nodes A and 1 are not late weak bisimilar.

5.2 Symbolic Early Weak Bisimulation

In our approach described in Section 4, we only require to have one symbolic weak transition relation for early and late weak bisimuluation. Furthermore, it is straight forward to get the symbolic early weak bisimulation algorithm from the late one as shown in Figure 6. The early algorithm, however, does not use the notion of "cover," and the scope of "$\forall z$" in boolean expressions generated is different as shown in the example below.

Let us reconsider the nodes A and 1 of the previous example with the new $match_{c?}$ function. For the edge $A \mapsto B$, it matches with any of the four $\leadsto$ edges: $1 \leadsto 2$, $1 \leadsto 3$, $1 \leadsto 4$, $1 \leadsto 5$. If we continue with the edges $A \mapsto C$ and $A \mapsto D$, we eventually get the following fragment of predicate equation system from our symbolic early weak bisimualtion algorithm.

$$\begin{aligned} X_{A1} = {} & \forall z_1\ (true \rightarrow (true \wedge X_{B2} \vee z_1 < 0 \wedge X_{B3} \vee true \wedge X_{B4} \vee z_1 \geq 0 \wedge X_{B5})) \\ & \wedge\ \forall z_2\ (true \rightarrow (true \wedge X_{C2} \vee z_2 < 0 \wedge X_{C3} \vee true \wedge X_{C4} \vee z_2 \geq 0 \wedge X_{C5})) \\ & \wedge\ \forall z_3\ (true \rightarrow (true \wedge X_{D2} \vee z_3 < 0 \wedge X_{D3} \vee true \wedge X_{D4} \vee z_3 \geq 0 \wedge X_{D5})) \end{aligned}$$

$match_{c?}\ (n, \sigma, n', \sigma')$:

for each $n \overset{b_i, c?x, \theta_i}{\longmapsto} n_i$ with all $n' \overset{b'_j, c?y, \theta'_j}{\rightsquigarrow} n'_j$

$\bigwedge_i \forall z (b_i\sigma \to \bigvee_j (b'_j\sigma'[y \mapsto z] \wedge X_{n_i, n'_j}[\theta_i\sigma[x \mapsto z], \theta'_j\sigma'[y \mapsto z]]))$

and for each i, j, if (n_i, n'_j) is not visited then push (n_i, n'_j) into $Queue$

$\wedge$

for each $n' \overset{b'_j, c?y, \theta'_j}{\longmapsto} n'_j$ with all $n \overset{b_i, c?x, \theta_i}{\rightsquigarrow} n_i$

$\bigwedge_j \forall z (b'_j\sigma' \to \bigvee_i (b_i\sigma[x \mapsto z] \wedge X_{n_i, n'_j}[\theta_i\sigma[x \mapsto z], \theta'_j\sigma'[y \mapsto z]]))$

and for each i, j, if (n''_k, n'_j) is not visited then push (n''_k, n'_j) into $Queue$

Figure 6: $match_{c?}$ of Symbolic Early Weak Bisimulation Algorithm

It is easy to prove that X_{A1} is true, which means that A and 1 are early weak bisimilar. Notice that since both of our algorithms use the same symbolic weak transition relation, called $\rightsquigarrow$ relation, it is easy to see the difference from early and late weak bisimulation.

6 Conclusion

We have described a representation for CCS-VP processes as symbolic graphs with assignment of [3], and defined operational semantics based on the graph. The representation in this paper, which is similar to that considered in [7], is different from that of Lin [3] with respect to the position of a substitution on a transition. Positioning the assignment last on a transition, the interpretation becomes more natural than that of [3].

We have also considered weak bisimulations for our symbolic graphs. According to traditional definitions of ground weak, early and late bisimulations, as for example considered in [8], two different weak transition relations are necessary in order to express the difference in the semantics. In particular, while a weak input action can be described by a single step in early semantics, it has to be divided by two relations in late semantics. This is necessary in order to represent the two steps in performing an input action according to late process semantics (the first step involves the choosing of the channel via which the communication is to be performed and the second being the receipt of an actual value via that channel). However, we show that for the purpose of deciding weak bisimulations it is sufficient to work with a single weak transition relation. In particular, we propose a reformulation of late weak bisimulation in which the early transition relation is used, that is receipt of an input takes place at the beginning of an input action, but, on matching an input action, a weakly late bisimilar process may use exactly one of its branches. This provides an alternative perspective for distinguishing the two weak bisimulations in terms of choice of transitions in matching input actions: given terms P and Q, for the purpose of early bisimulation, in order to match an input transition of P, term Q may use any appropriate branch for different input values. On the other hand, in the case of late bisimulation a single branch must be used to match all possible inputs. We have proved that the new formulation is equivalent to the original definition. In this sense, the definitions of early and weak bisimulations in this paper are very similar.

As far as the authors know, [8] is the firsty work introducing symbolic weak bisimulation. This is thereby achieved by lifting the traditional ground bisimulations to the symbolic setting (and also requires the presence two weak relations for the early and late cases). In this paper we have proposed symbolic weak bisimulations based on our ground definitions (and similarly based on a single transition relation) and we show that they correctly capture the ground definitions. Given the basic correspondence between our ground late bisimulation with the traditional definition, we conclude that our symbolic bisimulations coincide with those of [8].

We have given an algorithm for symbolic late weak bisimulation, based on the definition described here. This algorithm is very similar to that of Lin and Henessy [1, 3]. We have also shown how it can be slightly modified for symbolic early weak bisimulation.

Based on the symbolic strong (or, weak) bisimulation on CCS-VP developed in this paper, we are extending these results to the real-time process algebra ACSR-VP which is an extension of ACSR [2].

References

[1] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138:353–389, 1995.

[2] I. Lee, P. Brémond-Grégoire, and R. Gerber. A Process Algebraic Approach to the Specification and Analysis of Resource-Bound Real-Time Systems. *Proceedings of the IEEE*, pages 158–171, Jan 1994.

[3] H. Lin. Symbolic graphs with assignment. In V.Sassone U.Montanari, editor, *Proceedings CONCUR 96*, volume 1119 of *Lecture Notes in Computer Science*, pages 50–65. Springer-Verlag, 1996.

[4] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[5] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts 1 and 2. *Information and Computation*, 100:1–77, 1992.

[6] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114:149–171, 1993.

[7] P. Paczkowski. Characterizing bisimilarity of value-passing parameterised processes. In *Proc.Infinity Workshop on Verification of Infinite State Systems*, pages 47–55, 1996.

[8] Julian Rathke. *Symbolic Techniques for Value-passing Calculi.* PhD thesis, University of Sussex, 1997.

[9] Julian Rathke. Unique fixpoint unduction for value-passing processes. In *Proceedings* 12^{th} *Annual Symposium on Logic in Computer Science*, 1997.

Appendix

Proof of Lemma 4.5

PROOF: Since the two definitions coincide for non-input actions, it is sufficient to prove the following: Let t, u be m_σ, n_μ, respectively. $t \approx_2^b u$

iff whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u'$ for some y such that $b' \models b_2$ and there exists a b'-partition B' such that for each $b'' \in B'$ there is a u'' such that $u'[z/y] \stackrel{b_2',\varepsilon}{\Longrightarrow}_2 u''$ with $b'' \models b_2'$ and $(t'[z/x], u'') \in S^{b''}$

iff whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u'$ for some y such that $b' \models b_2$ and there exists a b'-partition B' such that for each $b'' \in B'$ there is a u'' such that $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u'$ and $u'[z/y] \stackrel{b_2',\varepsilon}{\Longrightarrow}_2 u''$ with $b'' \models b_2'$ and $(t'[z/x], u'') \in S^{b''}$

iff let b_2'' be a boolean expression such that $b_2''[z/y] = b_2'$ and let u''' be a term such that $u'''[z/y] = u''$. Then we have whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u'$ for some y such that $b' \models b_2$ and there exists a b'-partition B' such that for each $b'' \in B'$ there is a u''' such that $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u' \stackrel{b_2'',\varepsilon}{\Longrightarrow}_2 u'''$ with $b'' \models b_2''[z/y]$ and $(t'[z/x], u'''[z/y]) \in S^{b''}$

iff whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $u \stackrel{b_2,a?y}{\Longrightarrow}_2 u'$ for some y such that $b' \models b_2$ and there exists a b'-partition B' such that for each $b'' \in B'$ there is a u'' such that $u \stackrel{b_2 \wedge b_2'',a?y}{\Longrightarrow} u'''$ with $b'' \models b_2''[z/y]$ and $(t'[z/x], u'''[z/y]) \in S^{b''}$

iff whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B, z \notin fv(b')$ and there exists a $n_1 \stackrel{be,a?y',\theta}{\longmapsto} n_2$ such that $u \stackrel{b_2,p_2'}{\Longrightarrow}_2 u'$ is over $n_1 \stackrel{be,a?y',\theta}{\longmapsto} n_2$, $b' \models b_2$ and there exists a b'-partition B' such that for each $b'' \in B'$ there is a $u \stackrel{b_2 \wedge b_2'',a?y}{\Longrightarrow} u'''$ over $n_1 \stackrel{be,a?y',\theta}{\longmapsto} n_2$, with $b'' \models b_2''[z/y]$ and $(t'[z/x], u'''[z/y]) \in S^{b''}$

iff since $b'' \in B'$ is a b'-partition, $b'' \models b_2$. Since $b'' \models b_2''[z/y]$, $b'' \models b_2 \wedge b_2''[z/y]$. Letting b_3 be $b_2 \wedge b_2''$, we have $b'' \models b_3[z/y]$. Then we have, whenever $t \stackrel{b_1,a?x}{\longrightarrow} t'$ there exists a variable z such that $z \notin fv(b,t,u)$ and a $b \wedge b_1$-partition B, such that for each $b' \in B$, $z \notin fv(b')$ and there exists a $n_1 \stackrel{be,a?y',\theta}{\longmapsto} n_2$ and a b'-partition B' such that for each $b'' \in B'$, there exists $u \stackrel{b_3,a?y}{\Longrightarrow} u'''$ over $n_1 \stackrel{be,a?y',\theta}{\longmapsto} n_2$ with $b'' \models b_3[z/y]$ and $(t'[z/x], u'''[z/y]) \in S^{b''}$

iff $p \approx^b q$. □