

An Algebra of Boolean Processes ^{*}

Costas Courcoubetis
Department of Computer Science
University of Crete
Heraklion, Greece

Susanne Graf Joseph Sifakis
IMAG-LGI
BP 53X
F-38041 Grenoble

Abstract

This work has been motivated by the study of the S/R models which allow to represent systems as a set of communicating state machines cooperating through a shared memory.

We show that S/R models can be expressed in terms of a process algebra called Boolean SCCS which is a special case of Milner's SCCS, in the sense that the actions are elements of some boolean algebra. We define for Boolean SCCS an operational and a symbolic semantics modulo strong bisimulation equivalence. A complete axiomatisation of bisimulation and simulation equivalences on this algebra is proposed.

Furthermore, we propose a very general *renaming* operator, and show by means of examples that it allows the definition of *abstractions*.

1 Introduction

Most existing algebraic specification languages for concurrent systems such as process algebras, are based on the communicating processes model. They suppose that a system is composed of a set of components with disjoint state spaces, interacting by exchanging messages. Although of equal importance, models relying upon shared memory communication mechanisms did not attract so much the attention of researchers. A reason might be that the communicating processes model is sufficiently general to represent them. On the other hand, the use of shared memory formalisms leads to compact specifications due to the use of powerful communication mechanisms. By allowing processes to be labelled with complex boolean formulas instead of simple actions, we obtain processes with fewer states since a transition label can represent a set of atomic actions.

^{*}This work has been partially supported by ESPRIT Basic Research Action 'Spec'

Besides obtaining compact specifications, there are other issues which make such formalisms very interesting. They have to do with the possibility of doing reductions at the symbolic level, and in general, the possibility to perform a large part of the verification process at the same level. In order to achieve that, one can use the symbolic manipulation mechanisms provided by the boolean calculus. It is important to note that any reduction at the symbolic level will greatly enhance the applicability of the verification procedures by diminishing the state explosion effects. This paper attempts to define the notions of *symbolic bisimulation* and *abstraction* for such shared memory communicating processes. It also shows that the process algebra paradigm can be directly applied to shared memory models. Interestingly enough, abstraction and renaming in the above models are richer concepts than abstraction and renaming in the traditional communicating processes models.

In order to motivate the reader for using such shared memory formalisms, we start in Section 2 by describing the example of such a formalism which has been successfully used for specifying and verifying large concurrent systems. This is the Selection/Resolution model by R. Kurshan. A system is described by a set of processes modifying synchronously a common memory. The actions enabled in a process at each computation step, depend on the state of the global memory. In Section 3, we give the general definition of boolean transition systems, that is, transition systems whose labels are elements of a boolean algebra, and which is our model of the shared memory communicating processes. Section 4 presents an algebra of boolean processes, for which two different semantics in terms of boolean transition systems modulo bisimulation are defined: an ‘*operational*’ one, whose models are usual action-labelled transition systems, and a ‘*symbolic*’ one, whose models are transition systems whose labels are boolean expressions. We give notions of strong bisimulation for both semantics and show that they coincide on terms. Furthermore, we propose a complete axiomatisation of bisimulation on terms, showing that our algebra is a particular case of *SCCS* with boolean actions. In this section, we give also some results on renaming functions and illustrate their use for the definition of abstractions by an example. In Section 5, we define notions of simulation preorder and equivalence.

2 The Selection/Resolution Model

2.1 Informal presentation

The *selection/resolution* (*S/R*) model [AKS83a,AC85,GK80,Ku90,ABM86a] provides a method of describing a system as a set of coordinating finite state machines. Experience has shown that complex systems can be specified by using this model, and there are currently tools which automatically verify properties of the behaviours of such formal specifications, managing systems with millions of reachable states [Ku90]. An important feature is the fact that the coupling between machines is described in terms of predicates. This helps in many cases to obtain concise and understandable specifications.

A system is decomposed into a set of simple components; each component or *process* is an edge labelled directed graph (see Figure 1). The vertices of this graph are *states* of

the process; each directed edge describes a transition corresponding to one computation step. In each state, a process can nondeterministically choose from a set of *selections*, which are essentially values of a shared memory used for synchronization. In fact, there is a shared memory in the system consisting of a finite number of variables ranging over a finite domain. With each process is associated a subset of *selection variables* which are distinct for each process. A process can read all variables, whereas it can update only its own selection variables (selections are enclosed in braces next to the states in Figure 1, an example in which the selection functions are all deterministic).

A computation step of the system consists of a selection followed by a resolution phase. The *selection* of a process consists in choosing a value for each one of its selection variables. The *resolution* is done by calculating the *global selection*, i.e., the vector of all the current selections of the processes. Each process checks which transitions are consistent with the current selections of all processes, and then chooses one of these enabled transitions.

This is done as follows: Let \mathcal{B} denote the boolean algebra generated by the predicates over a set of variables. Predicates of this algebra are used as labels of both, states and edges. The predicate associated with a state characterizes the set of all the possible selections; the predicate associated with an edge is the enabling condition of the corresponding transition.

Consider k interacting processes SR_1, \dots, SR_k and let s_i be the predicate associated with the current state of SR_i at some state. Then, the global selection is characterized by the conjunction $s = s_1 \wedge \dots \wedge s_k$. A transition of SR_i labelled by condition ℓ_i is *enabled* iff $s \wedge \ell_i \neq 0$. A computation step consists in performing simultaneously in each process SR_i an enabled transition labelled by ℓ_i and such that $s \wedge \ell_1 \wedge \dots \wedge \ell_k \neq 0$.

2.2 The S/R -processes

Notation 2.1 \mathcal{B} is a boolean algebra with $\vee, \wedge, ^-, \Rightarrow$ denoting respectively disjunction, conjunction, complementation and implication. By convention, 0 and 1 represent respectively the bottom and the top element of \mathcal{B} and $\text{atoms}(\mathcal{B})$ is the set of atoms of \mathcal{B} .

Definition 2.2 (S/R -process)

An S/R -process on a boolean algebra \mathcal{B} is a triplet $SR = (Q, \delta, \sigma)$, where

- Q is a set of states,
- $\delta : Q \times Q \mapsto \mathcal{B}$ is a transition function,
- $\sigma : Q \mapsto \mathcal{B}$ is a selector function.

An S/R -process can be represented by a state- and edge-labelled directed graph whose vertices are the states. There is an edge from state q to q' labelled by ℓ iff $\delta(q, q') = \ell$ and $\ell \neq 0$.

\mathcal{B} can be considered as the boolean algebra generated by a set of atomic predicates on the process variables. In particular, the selections associated with the states of a process are elements of the boolean algebra generated by a set of atomic predicates on with its selection variables, which is a sub-algebra of \mathcal{B} .

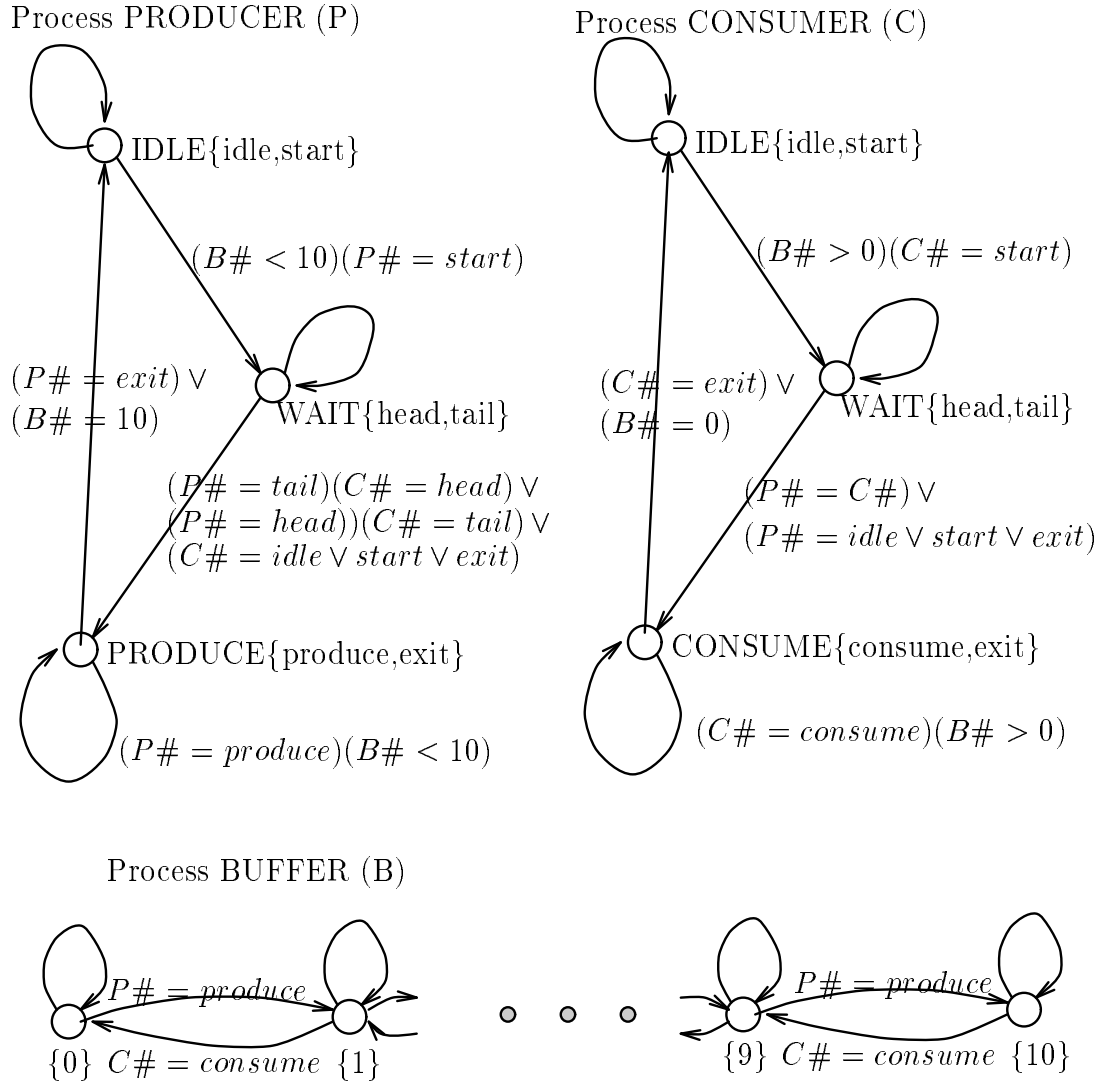


Figure 1: A producer/consumer system

Definition 2.3 (parallel composition on S/R -processes)

Let $SR_i = (Q_i, \delta_i, \sigma_i)$ for $i = 1, 2$ be two S/R -processes on \mathcal{B} . The parallel composition of SR_1 and SR_2 is the S/R -process $SR_1 \times SR_2 = (Q, \delta, \sigma)$ where,

- $Q = Q_1 \times Q_2$,
- $\sigma(q_1, q_2) = \sigma_1(q_1) \wedge \sigma_2(q_2)$
- $\delta((q_1, q_2), (q'_1, q'_2)) = \delta_1(q_1, q'_1) \wedge \delta_2(q_2, q'_2)$

Example We illustrate the use of the S/R model for the modelling of a simple producer/consumer system (see Figure 1).

There is a producer, a consumer and a buffer of capacity 10, and the access to the buffer is allowed to one process at a time. In case that both processes compete for entering the critical section they both “flip a coin”. If the result is the same for both, then the consumer wins else the producer.

In the description shown in Figure 1, we use the convention that the selections available at a particular state are enclosed in braces next to the states. For example, if the set of available selections in state q of process SR is the set $\{sel_1, sel_2\}$, then when in state q process SR can set its selection variable, denoted by $SR\#$, to assume the value sel_1 or sel_2 .

A self-loop without any label means that its label is the negation of the union of the labels of all the other outgoing edges.

We can feed the above specification to available tools such as COSPAN and SPANNER and prove properties of the infinite sequences of the global memory assignments (see [ABM86a,b], [ACW90], [KK86]).

3 Boolean Transition Systems

In this section, we define *boolean transition systems*, which are transition systems labelled by elements of a boolean algebra \mathcal{B} . They differ from S/R -processes only by the fact that they have no labels on states. We show the relationship between the two models.

3.1 S/R -processes as boolean transition systems (BTS)

Definition 3.1 (*Boolean transition systems*)

A boolean transition system on a boolean algebra \mathcal{B} is a pair $S = (Q, \rightarrow)$, where

- Q is a set of states,
- $\rightarrow \subseteq Q \times (\mathcal{B} - \{0\}) \times Q$ is the transition relation,

We write $q \xrightarrow{\ell} q'$ for $(q, \ell, q') \in \rightarrow$.

Definition 3.2 With an S/R -process $SR = (Q, \delta, \sigma)$ on \mathcal{B} , we associate a BTS, $\mathcal{B}ts(SR) = (Q, \rightarrow)$, where \rightarrow is defined as the least relation, subset of $Q \times (\mathcal{B} - \{0\}) \times Q$, such that:

$$(\delta(q, q') = \ell \text{ and } \ell \wedge \sigma(q) \neq 0) \text{ implies } q \xrightarrow{\ell \wedge \sigma(q)} q'.$$

The parallel composition on S/R -processes can easily be translated into the parallel composition of boolean transition systems:

Definition 3.3 (*parallel composition on BTS*)

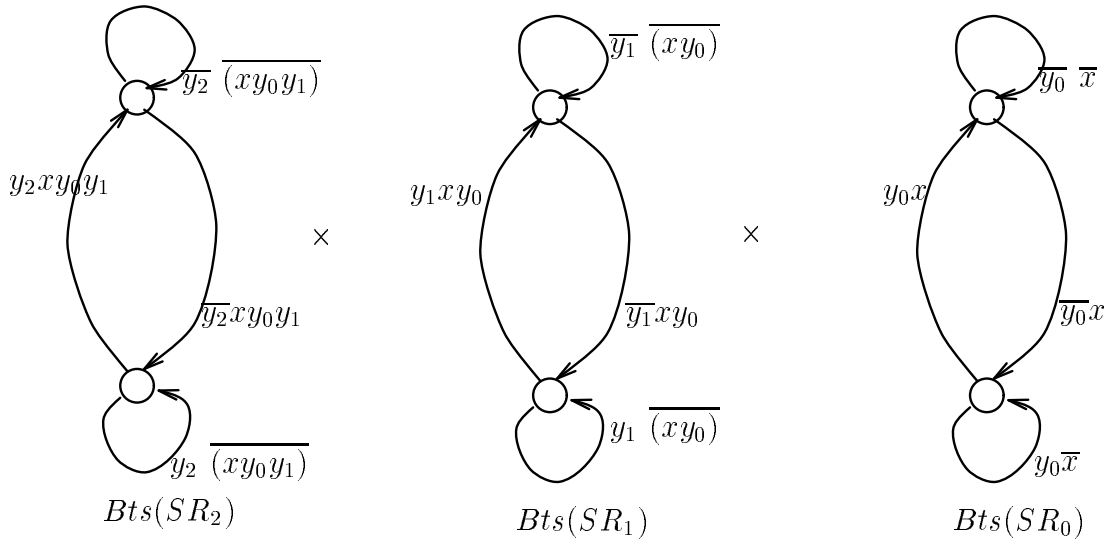
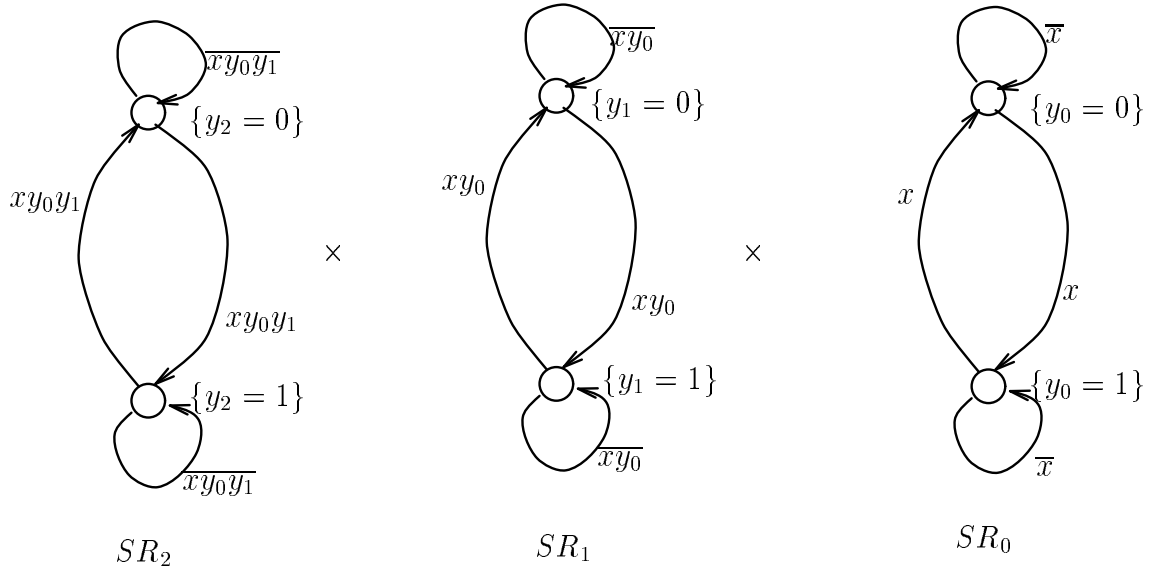
Let $S_i = (Q_i, \rightarrow_i)$ for $i = 1, 2$ be two BTSs. The parallel composition of S_1 and S_2 is the BTS, $S = S_1 \times S_2 = (Q, \rightarrow)$ where,

- $Q = Q_1 \times Q_2$,
- $(q_1, q_2) \xrightarrow{\ell} (q'_1, q'_2)$ iff $(\ell \neq 0 \text{ and } \exists \ell_1, \ell_2. (\ell = \ell_1 \wedge \ell_2 \text{ and } q_i \xrightarrow{\ell_i} q'_i \text{ for } i = 1, 2))$.

In fact, we obtain in a straight forward manner the following proposition:

Proposition 3.4 For any S/R -processes SR_1, SR_2 ,
 $\mathcal{Bts}(SR_1 \times SR_2) = \mathcal{Bts}(SR_1) \times \mathcal{Bts}(SR_2)$.

Example: translation of an S/R -process into a \mathcal{BTS}



Consider a counter modulo 8, whose output (the integers between 0 and 7) is represented by 3 boolean variables y_0, y_1, y_2 . Its input, the signal incrementing the counter, is represented by a boolean variable x .

The counter is modelled as the parallel composition of three S/R -processes SR_1, SR_2, SR_3 with selection variables respectively y_0, y_1 and y_2 (Figure 2.a)

The corresponding \mathcal{BTS} , $\mathcal{Bts}(SR_0 \times SR_1 \times SR_2)$ is obtained by calculating the parallel composition $\mathcal{Bts}(SR_0) \times \mathcal{Bts}(SR_1) \times \mathcal{Bts}(SR_2)$ as in Figure 2.b.

We introduce hereafter some useful notations for boolean transition systems.

Definition 3.5 *Let $S = (Q, \rightarrow)$ be a \mathcal{BTS} and $q \in Q$. Then,*

1. *The enabling condition of a state q , $enable(q)$ is the boolean expression $enable(q) = \bigvee_{\exists q'. q \xrightarrow{\ell_i} q'} \ell_i$.*
2. *q is called finitely branching iff $\forall \ell \in \mathcal{B}$ there is a finite number of labels ℓ' such that $(q \xrightarrow{\ell} \text{ and } \ell \Rightarrow \ell')$.*
3. *q is called deterministic iff $\forall \ell_1, \ell_2. q \xrightarrow{\ell_1} q_1 \text{ and } q \xrightarrow{\ell_2} q_2 \text{ implies } (\ell_1 = \ell_2 \text{ and } q_1 = q_2) \text{ or } \ell_1 \wedge \ell_2 = 0$.*
4. *q is called complete iff $enable(q) = 1$*
5. *q is called canonical iff $\forall q' (q \xrightarrow{\ell} q' \text{ implies } \ell \in atoms(\mathcal{B}))$.*

S is respectively called finitely branching, deterministic, complete or canonical if all its states have the corresponding property.

4 Boolean SCCS, an algebra for boolean transition systems

In this section, we define a process algebra that can be considered as a particular case of SCCS [Mi83], i.e., a process algebra with a synchronous parallel operator. Its action operators are the elements of some boolean algebra \mathcal{B} . Processes of this algebra have boolean transition systems as underlying models. We study in particular bisimulation semantics for this algebra.

Notation 4.1 *(renaming function)*

Let \mathcal{B} be a boolean algebra. Any mapping $\phi : \mathcal{B} \mapsto \mathcal{B}$ satisfying $\phi(0) = 0$ and which is distributive over disjunction ($\forall \ell_1, \ell_2 \in \mathcal{B}. \phi(\ell_1 \vee \ell_2) = \phi(\ell_1) \vee \phi(\ell_2)$) is called a renaming function on \mathcal{B} .

Definition 4.2 *(Syntax of BSCCS)*

Let \mathcal{B} be a boolean algebra, ϕ a renaming function and Z a set of variables. Represent by ℓ and z respectively, elements of \mathcal{B} and Z . Consider the term language defined by the

following grammar:

$$\begin{aligned} t_s &::= \circlearrowleft | z | \ell t_s | t_s + t_s | \text{recz}.t_s, \\ t &::= t_s | t \times t | t[\phi] | \ell t | t + t \end{aligned}$$

We call \mathcal{BSCCS} the sub-algebra of the closed terms, named also processes. As usually, a term is called guarded if in any subterm of the form $\text{recz}.t$ all occurrences of z in t are in the scope of an action-operator ℓ .

Notice that a term of \mathcal{BSCCS} has no occurrences of \times within the scope of a recursion operator as we want to restrict ourselves to regular processes.

4.1 Operational semantics

First, we give an operational semantics, i.e. we associate with each term in the usual manner a transition system whose states are terms and whose transitions are labelled by atoms of \mathcal{B} . In the case that \mathcal{B} is generated by a set of program variables, each transition corresponds exactly to a valuation of these variables, which justifies that we call this semantics ‘operational’.

Definition 4.3 (*operational semantics*)

For $\ell \in \mathcal{B}$, $a, a' \in \text{atoms}(\mathcal{B})$, $t_1, t_2, t \in \mathcal{BSCCS}$, ϕ a renaming function and z a process variable, the transition relation \rightsquigarrow on \mathcal{BSCCS} is defined as the smallest relation specified by the following rules.

1. $\ell t \rightsquigarrow^a t$ iff $a \Rightarrow \ell$
2.
$$\frac{t_1 \rightsquigarrow^a t'_1}{t_1 + t_2 \rightsquigarrow^a t'_1} \quad , \quad \frac{t_2 \rightsquigarrow^a t'_2}{t_1 + t_2 \rightsquigarrow^a t'_2}$$
3.
$$\frac{t_1 \rightsquigarrow^a t'_1, \quad t_2 \rightsquigarrow^a t'_2}{t_1 \times t_2 \rightsquigarrow^a t'_1 \times t'_2}$$
4.
$$\frac{t \rightsquigarrow^a t', \quad a' \Rightarrow \phi(a)}{t[\phi] \rightsquigarrow^{a'} t'[\phi]}$$
5.
$$\frac{t \rightsquigarrow^a t'}{\text{recz}.t \rightsquigarrow^a t'[\text{recz}.t/z]}$$

Clearly, these rules associate with any term of Boolean SCCS a canonical \mathcal{BTS} by defining for any operator an operator on \mathcal{BTS} s. The set of atoms can also be considered as the set of labels of a usual labelled transition system.

Remark: The renaming operator $[\phi]$ plays also the role of both an *abstraction* and a *restriction* operator, depending on the nature of ϕ . If ϕ associates 0 with some atoms, and leaves the others unchanged, then it corresponds to a restriction operator. The use of renaming as an abstraction operator will be illustrated later (see Section 4.5).

We are interested in strong bisimulation semantics on \mathcal{BTS} .

Notation 4.4 (*strong bisimulation \sim*)

Let $t_1, t_2 \in \mathcal{BSCCS}$. The strong bisimulation \sim [Mi89] is the largest symmetric relation \mathcal{R} , solution of $\Phi_0(\mathcal{R}) = \mathcal{R}$ where, $\forall a \in \text{atoms}(\mathcal{B}) \forall t'_1 \in \mathcal{BSCCS}$

$$((t_1, t_2) \in \Phi_0(\mathcal{R}) \text{ iff } t_1 \xrightarrow{a} t'_1 \text{ implies } \exists t'_2. (t_2 \xrightarrow{a} t'_2 \text{ and } (t'_1, t'_2) \in \mathcal{R}))$$

As usually, we write $t_1 \sim t_2$ instead of $(t_1, t_2) \in \sim$.

We use also the notions of ‘bisimulation up to depth i ’, defined by:

$$- \sim_0 = \mathcal{BSCCS} \times \mathcal{BSCCS}$$

$$- \sim_{i+1} = \Phi_0(\sim_i)$$

We have $\sim = \bigcap_{i=1}^{\infty} \sim_i$ as even for infinite \mathcal{B} any term has only a finite number of ‘ a -derivations’ for any $a \in \text{atoms}(\mathcal{B})$.

Proposition 4.5 \sim is a congruence on \mathcal{BSCCS} .

The proof is routine, and similar to the one given in [Mi83]. Except that in order to prove the preservation of \sim by the renaming operators, we need the the fact that there defining functions are strict and distribute over disjunction.

4.2 Symbolic semantics for Boolean SCCS

In this section, we give a different semantics associating an arbitrary \mathcal{BTS} with a term of Boolean SCCS. We define a *symbolic bisimulation* which is proven to coincide with strong bisimulation on \mathcal{BSCCS} .

Definition 4.6 (*symbolic semantics*)

For $\ell_1, \ell_2, \ell \in \mathcal{B}$, $t_1, t_2, t \in \mathcal{BSCCS}$ and z a process variable, let \rightarrow be the transition relation, defined as the smallest relation specified by the following rules.

1. $\ell t \xrightarrow{\ell} t$ iff $\ell \neq 0$
2.
$$\frac{t_1 \xrightarrow{\ell} t'_1}{t_1 + t_2 \xrightarrow{\ell} t'_1}, \quad \frac{t_2 \xrightarrow{\ell} t'_2}{t_1 + t_2 \xrightarrow{\ell} t'_2}$$
3.
$$\frac{t_1 \xrightarrow{\ell_1} t'_1, \quad t_2 \xrightarrow{\ell_2} t'_2, \quad \ell_1 \wedge \ell_2 \neq 0}{t_1 \times t_2 \xrightarrow{\ell_1 \wedge \ell_2} t'_1 \times t'_2}$$

$$4. \frac{t \xrightarrow{\ell} t', \phi(\ell) \neq 0}{t[\phi] \xrightarrow{\phi(\ell)} t'[\phi]}$$

$$5. \frac{t \xrightarrow{\ell} t'}{recz.t \xrightarrow{\ell} t'[recz.t/z]}$$

Remarks:

- As for the operational semantics, these rules allow to associate in an obvious manner with any term of Boolean SCCS a \mathcal{BTS} (not necessarily a canonical one) by defining for any operator an operator on \mathcal{BTS} s.
- Conversely, with any finite \mathcal{BTS} can be associated a process in an obvious manner. Thus, in the sequel we identify a term of Boolean SCCS with its corresponding boolean transition system.
- Therefore, the notations of Definition 3.5 can be applied to terms. We say for a term t , $enable(t) = \ell$, t is respectively *finitely branching*, *deterministic*, *complete* or *canonical* if and only if this is the case for the \mathcal{BTS} associated via its symbolic semantics. Notice that the notion of t ‘*canonical*’ makes only sense for the symbolic semantics, and the notions of t ‘*deterministic*’ are different for the two semantics; e.g., $(a \vee b)t + at$ is deterministic for the operational but not for the symbolic semantics.

Definition 4.7 (*symbolic bisimulation*)

Let be $t_1, t_2 \in \mathcal{BSCCS}$. Then, \simeq is defined as the largest symmetric relation, solution of $\Phi(\mathcal{R}) = \mathcal{R}$, where

$$(t_1, t_2) \in \Phi(\mathcal{R}) \text{ iff } \forall \ell \in \mathcal{B} \forall t_1 \in \mathcal{BSCCS} \\ (t_1 \xrightarrow{\ell} t'_1 \text{ implies } \exists I. ((\ell \Rightarrow \bigvee_{i \in I} \ell_i) \text{ and } \forall i \in I \exists t_{2i}. (t_2 \xrightarrow{\ell_i} t_{2i} \text{ and } (t'_1, t_{2i}) \in \mathcal{R})))$$

As usually, we write $t_1 \simeq t_2$ instead of $(t_1, t_2) \in \simeq$ and we say that t_1 *symbolically bisimulates* t_2 .

Remarks:

- $t_1 \simeq t_2$ implies $enable(t_1) = enable(t_2)$.
- Any complete term symbolically bisimulates the process $\mathbf{1}$, defined as $\mathbf{1} = recz.1z$.

The characterization of bisimulation as the intersection of bisimulations up to depth i can also be shown for symbolic bisimulation.

Proposition 4.8 $\simeq = \bigcap_{i=1}^{\infty} \simeq_i$, where

- $\simeq_0 = \mathcal{BSCCS} \times \mathcal{BSCCS}$

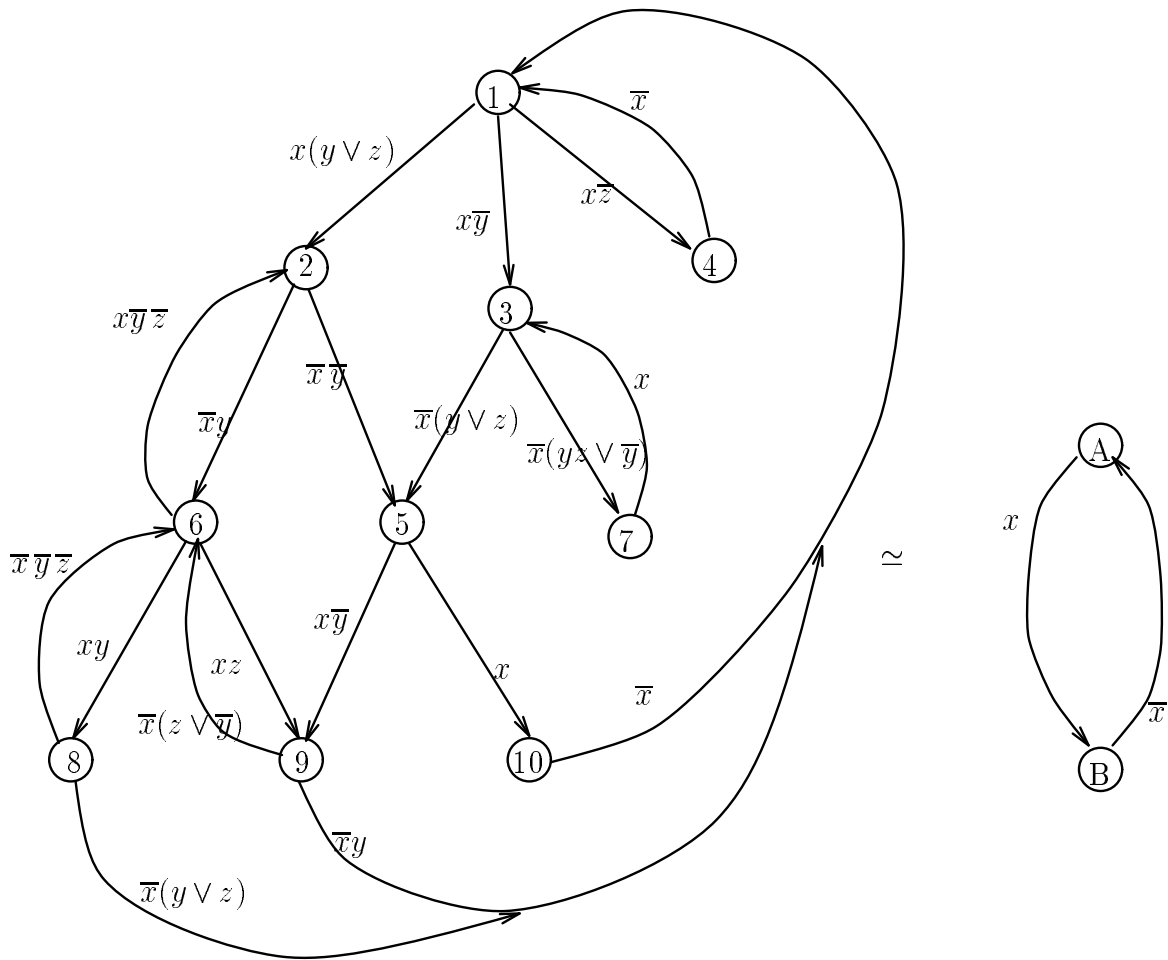


Figure 3: Reduction modulo symbolic bisimulation

- $\simeq_{i+1} = \Phi(\simeq_i) \quad \forall i \in \mathbb{N}, i > 0$

This result can be used to compute \simeq , and thus to reduce processes, and also finite \mathcal{B} TSSs. In Figure 3, we give two symbolically bisimilar \mathcal{B} TSSs. The small one is the quotient modulo \simeq of the other.

Proposition 4.9 *Symbolic and strong bisimulation agree on \mathcal{B} SCCS, i.e. $\simeq = \sim$.*

Proof: We prove that for any i , $\sim_i = \simeq_i$ by induction. The proof is easy if we use the observation that $\forall a \in \text{atoms}(\mathcal{B}) \quad t \stackrel{a}{\sim} t' \text{ iff } \exists \ell \in \mathcal{B}. (a \Rightarrow \ell \text{ and } t \xrightarrow{\ell} t')$. \square

Proposition 4.10 *\simeq is a congruence on \mathcal{B} SCCS.*

The proof is direct from the proposition above and the fact that \sim is a congruence on \mathcal{B} SCCS.

4.3 Results for deterministic processes

For deterministic processes the definition of \simeq can be simplified in the following manner:

Definition 4.11 *Let \simeq^d be the largest symmetric relation on \mathcal{B} SCCS, solution of $\Phi_1(\mathcal{R}) = \mathcal{R}$, where $(t_1, t_2) \in \Phi_1(\mathcal{R})$ iff*

- $\text{enable}(t_1) = \text{enable}(t_2)$
- $t_1 \xrightarrow{\ell} t'_1$ implies $\forall \ell', t'_2. ((\ell \wedge \ell' \neq 0 \text{ and } t_2 \xrightarrow{\ell'} t'_2) \text{ implies } (t'_1, t'_2) \in \mathcal{R})$

Proposition 4.12 *(characterization of \simeq on deterministic processes)*
For $t_1, t_2 \in \mathcal{B}$ SCCS, t_1, t_2 nondeterministic $(t_1 \simeq t_2 \text{ iff } t_1 \simeq^d t_2)$.

Proof: We have already noticed that $t_1 \simeq t_2$ implies $\text{enable}(t_1) = \text{enable}(t_2)$. Furthermore, the definition of \simeq says that for any ℓ -transition of t_1 leading to t'_1 , there exists a set of transitions from t_2 whose labels cover ℓ and which lead to equivalent terms. For a deterministic process the set of transitions whose labels cover ℓ is unique. In this case the condition in the definition of \simeq ,

$\forall \ell \in \mathcal{B} \exists I, \{t_i\}. ((\bigvee_{i \in I} \ell_i \Rightarrow \ell) \text{ and } \forall i \in I \quad t \xrightarrow{\ell_i} t_i)$
is equivalent to

$\forall \ell \in \mathcal{B} (\exists \ell' \in \mathcal{B}, t' \in \mathcal{B}$ SCCS. $(\ell \wedge \ell' \neq 0 \text{ and } t \xrightarrow{\ell'} t'))$.

The fact that \simeq and \simeq^d coincide, is easy to deduce from this observation. \square

Notice that for two terms t and t' , it is sufficient that one of them is deterministic in order that $t \simeq t'$ and $t \simeq^d t'$ coincide. Furthermore, the relation \simeq^d gives rise to a simpler verification algorithm.

4.4 An axiomatisation of bisimulation on Boolean SCCS

The axioms and rules characterizing \simeq on Boolean SCCS consist of the axioms characterizing strong bisimulation on SCCS and some additional axioms due to the laws of the action set \mathcal{B} .

Theorem 4.13 (*axiomatization*)

The axiomatization given in Table 1 is sound and complete for \simeq on Boolean SCCS.

Proof: The proof of soundness is standard, except for the axioms concerning renaming, for which we need the fact that ϕ is strict and distributes over disjunction.

The completeness can be deduced from the completeness of the axioms (1), (2) and (11) to (13) for strong bisimulation on terms in canonical form obtained in the following manner.

In a first step, a term is transformed into an equivalent one without occurrences of \times and renaming operators by means of the axioms (4) to (10), commutativity and associativity. In a second step, such a term can be transformed by using (14) to (16) into canonical form, in which the only action names are atoms of \mathcal{B} .

We have already shown that \simeq coincides with strong bisimulation, and on canonical terms strong bisimulation can be characterized by the axioms and rules (1),(2) and (11) to (13) [Mi84]. \square

4.5 Some results on renaming

In the following propositions we give some sufficient conditions on functions ϕ in order that the corresponding renaming operators $[\phi]$ preserve particular properties of terms.

Proposition 4.14 *Let ϕ be a renaming function on \mathcal{B} .*

$\forall t_1, t_2 \in \mathcal{BSCCS} \quad (t_1 \times t_2)[\phi] = t_1[\phi] \times t_2[\phi]$, *i.e., $[\phi]$ distributes over \times ,*
iff

$\forall l_1, l_2 \in \mathcal{B} \quad \phi(l_1 \wedge l_2) = \phi(l_1) \wedge \phi(l_2)$, *i.e. ϕ distributes over conjunction.*

Proof: Suppose that $t_1 \xrightarrow{\ell_1} t'_1$ and $t_2 \xrightarrow{\ell_2} t'_2$. Then,

$t_1 \times t_2 \xrightarrow{\ell_1 \wedge \ell_2} t'_1 \times t'_2$ iff $\ell_1 \wedge \ell_2 \neq 0$, and

$(t_1 \times t_2)[\phi] \xrightarrow{\phi(\ell_1 \wedge \ell_2)} (t'_1 \times t'_2)[\phi]$ iff $\phi(\ell_1 \wedge \ell_2) \neq 0$. Whereas,

$(t_1[\phi] \times t_2[\phi]) \xrightarrow{\phi(\ell_1) \wedge \phi(\ell_2)} (t'_1[\phi] \times t'_2[\phi])$ iff $\phi(\ell_1) \wedge \phi(\ell_2) \neq 0$.

From this it follows easily that for arbitrary t_1, t_2 $[\phi]$ distributes over \times if and only if ϕ distributes over conjunction on \mathcal{B} . \square

Notice that distributivity over conjunction is a very strong requirement for a renaming function, and the renaming functions used for abstraction of our example given at the end of the section do not have this property.

Proposition 4.15 *Let ϕ be a renaming function on \mathcal{B} and t a term of Boolean SCCS.*

(1) Axioms of SCCS:

1. $+$ is commutative, associative and idempotent,
2. $t + \emptyset = t$
3. \times is commutative and associative
4. $t \times \emptyset = \emptyset$
5. $t \times (t_1 + t_2) = (t \times t_1) + (t \times t_2)$
6. $\ell_1 t_1 \times \ell_2 t_2 = (\ell_1 \wedge \ell_2)(t_1 \times t_2)$
7. $\mathbf{1} \times t = t$
8. $\emptyset[\phi] = \emptyset$
9. $(at)[\phi] = a(t[\phi])$
10. $(t_1 + t_2)[\phi] = t_1[\phi] + t_2[\phi]$
11. $recz.(z + t') = recz.t'$
12. $recz.t = t[recz.t/z]$
13. $\frac{t' = t[t'/z]}{t' = recz.t}$ provided that t' is guarded

(2) Axioms and rules specific to $\mathcal{B}SCCS$:

14. $0t = \emptyset$
 15. $\ell_1 t + \ell_2 t = (\ell_1 \vee \ell_2)t$
 16. $\frac{\ell_1 \equiv \ell_2}{\ell_1 t = \ell_2 t}$
-

Table 1: Axiomatisation of \simeq on $\mathcal{B}SCCS$

1. If ϕ is increasing, i.e. $\forall \ell \in \mathcal{B}(\ell \Rightarrow \phi(\ell))$, then $[\phi]$ preserves completeness of t , i.e. if t is complete then $t[\phi]$ is also complete.
2. If ϕ is such that $(\forall \ell_1, \ell_2 \in \mathcal{B} \ell_1 \wedge \ell_2 = 0 \text{ implies } \phi(\ell_1) \wedge \phi(\ell_2) = 0)$ then $[\phi]$ preserves determinism of t , i.e. if t is deterministic then $t[\phi]$ is also deterministic.
3. If ϕ maps atoms to atoms, i.e. $\phi : \text{atoms}(\mathcal{B}) \mapsto \text{atoms}(\mathcal{B})$, then $[\phi]$ preserves canonicity of t , i.e. if t is canonical then $t[\phi]$ is also canonical.

Notice that the condition of (3) implies the condition of (2).

An Example: modulo 8 counter (see [Ma91])

In this example, we illustrate the use of renaming functions to obtain abstractions. Consider again a modulo 8 counter, defined in a slightly different manner than in Section 3. C_8 is defined as the parallel composition $C_8 = C(\beta, y_2, 1) \times C(\alpha, y_1, \beta) \times C(x, y_0, \alpha)$ where the subterms $C(v_1, v_2, v_3)$, defined in Figure 4, represent modulo 2 counters changing their state on input signal v_1 with state variable v_2 and output variable v_3 , representing the overflow bit. The observable variables of the modulo 8 counter are the global input x and the state variables y_2, y_1, y_0 whereas α and β are only used for synchronization. The renaming function ϕ_1 defined by

$$\phi_1(\ell(x, y_2, y_1, y_0, \alpha, \beta)) = \exists \alpha \beta \ell(x, y_2, y_1, y_0, \alpha, \beta) \equiv \ell(x, y_2, y_1, y_0, 0, 0) \vee \ell(x, y_2, y_1, y_0, 0, 1) \vee \ell(x, y_2, y_1, y_0, 1, 0) \vee \ell(x, y_2, y_1, y_0, 1, 1)$$

allows to make abstraction from the overflow variables α and β .

The \mathcal{BTS} corresponding to $C_8[\phi_1]$ has 8 states and cannot be reduced modulo symbolic bisimulation, but its boolean expressions are simpler than that of the \mathcal{BTS} of C_8 .

Consider the renaming function ϕ_2 :

$$\phi_2(\ell(x, y_2, y_1, y_0)) = \exists y_2 \ell(x, y_2, y_1, y_0) \equiv \ell(x, 0, y_1, y_0) \vee \ell(x, 1, y_1, y_0)$$

which applied to $C_8[\phi_1]$ allows to abstract from y_2 .

The boolean transition system corresponding to the process $C_8[\phi_1][\phi_2]$ can be reduced to the one presented in Figure 5 and corresponds clearly to a counter modulo 4.

5 Simulation preorders and equivalences on \mathcal{BSCCS}

Bisimulation is a strong equivalence, and if we are interested in verifying safety properties much weaker equivalences are interesting [BGFRS90]. In this section, we study simulation preorders and the equivalences they introduce on Boolean SCCS.

Simulation preorder \sqsubseteq^s on Boolean SCCS is defined as follows:

Definition 5.1 (*simulation preorder \sqsubseteq^s*)

$$\forall t_1, t_2 \in \mathcal{BSCCS} \quad t_1 \sqsubseteq^s t_2$$

- $t_1 \xrightarrow{\ell} t'_1$ implies $\exists I. ((\ell \Rightarrow \bigvee_{i \in I} \ell_i) \text{ and } \forall i \in I \exists t_{2i}. (t_2 \xrightarrow{\ell_i} t_{2i} \text{ and } t'_1 \sqsubseteq^s t_{2i}))$

The simulation equivalence induced by \sqsubseteq^s is denoted by \simeq^s .

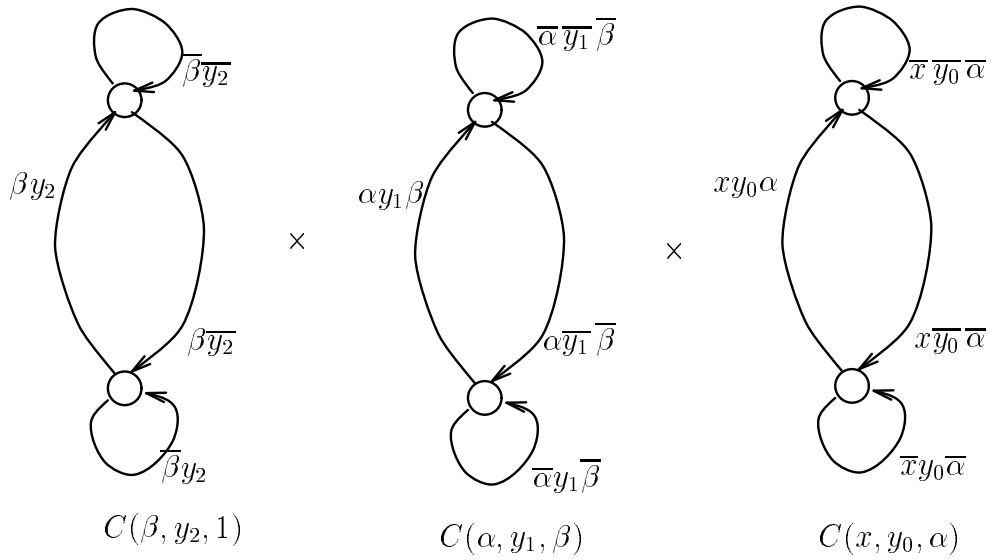


Figure 4: C_8 , a modulo 8 counter

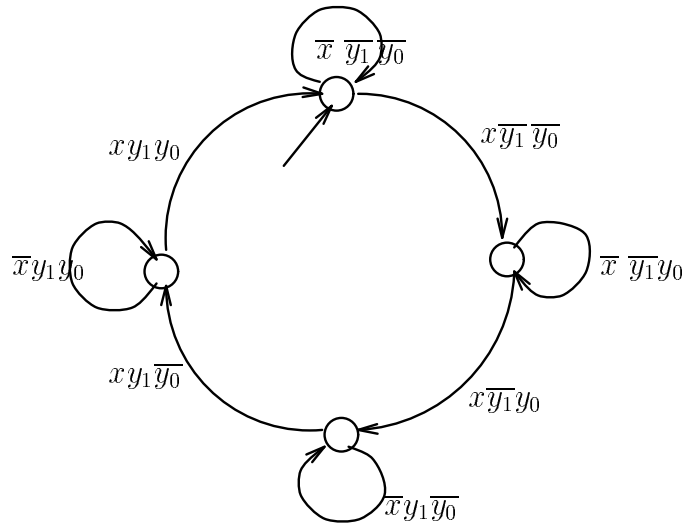


Figure 5: The reduced \mathcal{BTS} of $C_8[\phi_1][\phi_2]$

Remark: As in the case of bisimulation, it can be shown that the above defined simulation preorder coincides on canonical \mathcal{BTS} with the usual simulation preorder.

Proposition 5.2 (*characterization of \sqsubseteq^s*)

$$\forall t \in \mathcal{BSCCS} \quad \{t' \in \mathcal{BSCCS} \mid t' \sqsubseteq^s t\} = \{t' \mid \exists t'' \in \mathcal{BSCCS}. t' \simeq^s t'' \times t\}$$

-
1. All axioms of Table 1, where each equation $t_1 = t_2$ stands for two equations $t_1 \leq t_2$ and $t_2 \leq t_1$
 2. $\emptyset \leq t$
 3. $\frac{t_1 \leq t_2[t_1/z]}{t_1 \leq \text{rec}z.t_2}$ provided z guarded in t_2
 4. $\frac{t_2[t_1/z] \leq t_1}{\text{rec}z.t_2 \leq t_1}$ provided z guarded in t_2
 5. $\frac{l_1 \Rightarrow l_2}{l_1 t \leq l_2 t}$
-

Table 2: Axiomatisation of \simeq^s on Boolean SCCS

Proposition 5.3 (*Axiomatization of \sqsubseteq^s*)

The axiomatization given in Table 2 is sound and complete for \sqsubseteq^s on Boolean SCCS.

Proof: The soundness of this axiomatisation is easy to check. The completeness proof is very similar to the one of Theorem 4.13. As before, each term can be transformed into one in canonical form. In [BGFRS90] it has been shown for a term algebra isomorphic to the subalgebra of canonical terms that the above axiomatization (without rule (5) and based on the axioms of SCCS only) characterizes completely the usual simulation preorder. \square

6 Conclusion

This work establishes a connection between the S/R model and process algebras. For this, we introduce boolean transition systems, an extension of ordinary transition systems.

We believe that the Boolean Process Algebra and its underlying model deserve a further study as such, independently of the S/R model. In fact, they seem to be fairly appropriate formalisms to describe hardware and in general finite systems where data are coded by boolean variables.

Furthermore, symbolic bisimulation allows compare descriptions given by state transition models where labels represent sets of actions. The two given semantics show that boolean processes are more abstract.

It would be interesting to introduce weaker equivalences, such as stuttering equivalence on these models. Another interesting problem is to characterize the renaming functions introducing interesting abstraction criteria.

References

- [ABM86a] S. Aggarwal, D. Barbara, K. Z. Meth. "SPANNER - A Tool for the Specification, Analysis, and Evaluation of Protocols," IEEE Trans. on Software Engineering (to appear).
- [AC85] S. Aggarwal, C. Courcoubetis. "Distributed Implementation of a Model of Communication and Computation," Proceedings of the 18th Hawaii Int. Conf. on System Sciences, January, 1985, pp. 206-218.
- [AKS83a] S. Aggarwal, R. P. Kurshan, K. K. Sabnani. "A Calculus for Protocol Specification and Validation," in Protocol Specification, Testing and Verification, III (Rudin and West eds.), North-Holland, 1983, pp. 19-34.
- [Ku90] R. Kurshan, "Analysis of Discrete Event Coordination". LNCS 430 (1990), pp. 414-453.
- [ACW90] S. Aggarwal, C. Courcoubetis, P. Wolper. "Adding Liveness Properties to Coupled Finite-State Machines", ACM TOPLAS, Vol. 12, No 2, April 1990, pp. 303-339.
- [GK80] B. Gopinath, B. Kurshan. "The Selection/Resolution Model for Coordinating Concurrent Processes", AT&T Bell Laboratories Technical Report.
- [KK86] J. Katzenelson, B. Kurshan, "S/R: A Language for Specifying Protocols and other Coordinating Processes", Proc. 5th Ann. Int'l Phoenix Conf. Comput. Commun., IEEE, 1986, pp. 286-292.
- [BGFRS90] A. Bouajjani, J.-C. Fernandez, S. Graf, C. Rodriguez, J. Sifakis. *Safety for Branching Semantics*, ICALP 91, Madrid, LNCS Vol. 510, 1991.
- [Ma91] F. Maraninchi. *Argos: a graphical synchronous language for the description of reactive systems*, Report Spectre C-29, Grenoble, March 91, submitted to SCP
- [Mi80] R. Milner. *A Calculus for Communicating Systems*, LNCS 92, 1980
- [Mi83] R. Milner. *Calculi for Synchrony and Asynchrony*, Theoret. Comp. Sci. 25 p267-310, 1983
- [Mi84] R. Milner. *A Complete Inference System for a Class of Regular Behaviours*, Journal of Comp. and Syst. Sci. Vol. 28, 1984
- [Mi89] R. Milner. *Communication and Concurrency*, Prentice Hall, 1989