

Covering Step Graph

François Vernadat, Pierre Azéma, François Michel
e-mail : {vernadat,azema,fmichel}@laas.fr

LAAS-CNRS
7 avenue du Colonel Roche F-31077 Toulouse cedex - France

Abstract. Within the framework of concurrent systems, several verification approaches require as a preliminary step the complete derivation of the state space. Partial-order methods are efficient for reducing the state explosion due to the representation of parallelism by interleaving. The covering step graphs are introduced as an alternative to labelled transition systems. A transition step consists of several possibly concurrent events. In a covering step graph, steps of independent transitions are substituted as much as possible to the subgraph which would result from the firing of the independent transitions. Attention must be paid to the case of conflict and confusion. An algorithm for the "on the fly" derivation of step graphs is proposed. This algorithm is then extended to behaviour analysis by means of observational equivalence. A performance evaluation is made with respect to other methods.

Keywords: concurrent systems, state space exploration, partial-order, verification methods

1 Introduction

The state space derivation represents the preliminary step of several verification methods for concurrent systems. This approach is made attractive by the existence of efficient and automatic verification techniques, such as bisimulation and model-checking. The combinational explosion is the main limitation of this approach.

Many studies are currently in progress for reducing this problem: on the fly bisimulation [FM 90], symbolic marking graph derivation through symmetrical folding [Jen 87], so-called partial order techniques which attempt to avoid the combinational explosion resulting from the concurrency interpretation by means of interleaving: persistent sets [Val 89], sleep sets [GW 91]. An other direction, followed in [Esp 93, McMil 95], avoids the state explosion by using model checker directly on the description of the system (unfolding of Petri Nets).

The partial order techniques (see [WG 93] for a general survey) represent the framework of the approach developed in this paper. The basic principle of these approaches consists into considering a single specific path among all the sequences which possess the same Mazurkiewicz's trace [Maz 87]. In the case of persistent or sleep sets, only a subset of enable transitions is examined, the derived graph is then a subgraph of the whole graph.

The proposed approach visit all the transitions, but some independent events are put together to build a single transition step, the firing of this transition step is then atomic. The resulting graph is referred as covering step graph (CSG). A notion of step graph appears in [Rei 85]; this graph is obtained in a standard way except in presence of independently enable transitions where a step gathering these transitions is considered as an additional event and put to the set of transitions. The standard LTS consequently is a sub-graph of the Reisig's step graph. The covering step graph proposed here, is radically different because steps of independent transitions are substituted as much as possible to the subgraph which would result from the firing of the independent transitions. The potential benefit of such a substitution may be exponential with respect to the number of "merged" independent transitions. Some examples, in the paper, show that this potential benefit may effectively be obtained.

The covering step graph supplies an alternative structure to the traditional reachable state graph. The CSG analysis allows the control of global reachability properties, such as liveness or presence of deadlock. Furthermore, step graphs may be used for checking behaviour properties such as observational equivalence. An algorithm for the "on the fly" derivation of CSG is proposed. This algorithm is then extended to behaviour analysis by means of observational equivalence. A performance evaluation is made with respect to other methods.

Section 2 presents the covering step graph definition and the general properties preserved by such structures. Section 3 shows how to modify a standard enumeration algorithm in order to obtain a step graph. This algorithm is adapted in section 4 for computing a step graph observationally equivalent to the regular state graph. A preliminary assessment of the proposed approach is developed in section 5, by comparing it with stubborn and symmetry methods in the case of a standard data base example. Current work and extensions are finally mentioned.

2 Covering Step Graphs

2.1 Basic Notions

Labelled Transition Systems (LTS) (definition)

A LTS is a quadruple $\langle S, s_o, T, \rightarrow \rangle$ where S is a set of states, s_o a distinguished state in S , T is a set of transition labels, \rightarrow is a set of transitions ($\rightarrow \subset S \times T \times S$).

Notations:

$s \xrightarrow{t}$ iff $\exists s' \in S : (s, t, s') \in \rightarrow$, $s \not\xrightarrow{t}$ iff not $s \xrightarrow{t}$,

$s \xrightarrow{t} s'$ to mean that $(s, t, s') \in \rightarrow$

$s_0 \xrightarrow{w} s_n$ iff $w = t_1.t_2 \dots t_n$ and $s_0 \xrightarrow{t_1} s_1, s_1 \xrightarrow{t_2} s_2, \dots, s_{n-1} \xrightarrow{t_n} s_n$

2.1.1 Independency Relation

Let $\Sigma = \langle S, s_o, T, \rightarrow \rangle$ be a LTS and \imath a binary relation over the set of transitions ($\imath \subset T \times T$)

definition: [WG 93] \wr is an independent relation over Σ iff $\forall s, s_1, s_2 \in S, \forall t_1, t_2 \in T$:

$$[t_1 \neq t_2, s \xrightarrow{t_1} s_1, s \xrightarrow{t_2} s_2 \text{ and } t_1 \wr t_2] \Rightarrow s_1 \xrightarrow{t_2} s' \text{ and } s_2 \xrightarrow{t_1} s'$$

if t_1 and t_2 are enabled in s then if $s \xrightarrow{t_1} s_1$ (resp $s \xrightarrow{t_2} s_2$) then t_2 is enabled in s_1 (resp t_1 is enabled in s_2), moreover, there exists a unique state s' such that both $s_1 \xrightarrow{t_2} s'$ $s_2 \xrightarrow{t_1} s'$ (commutativity of enabled independent transitions)

The complement relation of \wr is called conflict, or dependency, relation. The conflict relation is denoted $\#$.

Independency relation in Petri Net [Rei 85]

Let $R = \langle P, T, Pre, Post \rangle$ a (Place/Transition or Condition/Event) net, m_0 an initial marking and $G(R, m_0)$ the reachability graph associated with the marked net $\langle R, m_0 \rangle$

For each transition $t \in T$ we define: $\bullet t =_{def} \{p \in P : Pre(p, t) \neq 0\}$,

$$t^\bullet =_{def} \{p \in P : Post(p, t) \neq 0\} \text{ and } \bullet t^\bullet =_{def} \bullet t \cup t^\bullet$$

Place/Transition Nets Let $|_{P/T} \subset T \times T$ defined as $t_1 |_{P/T} t_2$ iff $\bullet t_1 \cap \bullet t_2 = \emptyset$

For each Place/Transition marked net $\langle R, m_0 \rangle$,

$|_{P/T}$ is an independency relation over $G(R, m_0)$.

Condition/Event Nets Let $|_{C/E} \subset T \times T$ defined as $t_1 |_{C/E} t_2$ iff $\bullet t_1 \bullet \cap \bullet t_2^\bullet = \emptyset$

For each Condition/Event marked net $\langle R, m_0 \rangle$,

$|_{C/E}$ is an independency relation over $G(R, m_0)$.

2.1.2 Mazurkiewicz's Traces [Maz 87]

Concurrent Alphabet: A concurrent alphabet is a couple $\mathcal{E} = (\alpha, \#)$ where α is an alphabet and $\#$ the *dependency* in α ($\#$ is a reflexive and symmetric relation). The complementary relation $\#^C$ of relations¹ $\#$ in α is called independency relation in α

Mazurkiewicz's Traces:

Let \mathcal{E} be concurrent alphabet $(\alpha, \#)$, the equivalence relation in α^* , $\equiv_{\mathcal{E}}$ is defined by: $W \equiv_{\mathcal{E}} W'$ iff there exists a finite sequence (W_0, W_1, \dots, W_n) such that

$$W_0 = W \text{ and } W_n = W' \text{ and}$$

$$\forall i, 1 \geq i \geq n : \exists u, v \in \alpha^*, \exists (a, b) \in \#^C \text{ such that } W_{i-1} = ubav \text{ et } W_i = uabv$$

Relation $\equiv_{\mathcal{E}}$ defines the trace equivalence over \mathcal{E} . Equivalence classes of $\equiv_{\mathcal{E}}$ are called traces over \mathcal{E} . A trace generated by a string w is denoted by $[w]_{\mathcal{E}}$.

Trace properties:

* $\equiv_{\mathcal{E}}$ is the least congruence in the monoïd $(\alpha, \cdot, \epsilon)$ satisfying:

$$(a, b) \in \#^C \Rightarrow a.b \equiv_{\mathcal{E}} b.a$$

* Let Σ, \wr, \mathcal{E} be respectively a LTS $\langle S, s_0, T, \rightarrow \rangle$, an independency relation over Σ and an associated concurrent alphabet (T, \wr^C) , then:

$$\forall s \in S, w_1, w_2 \in T^* \text{ such that } s \xrightarrow{w_1} s_1 \ \& \ s \xrightarrow{w_2} s_2 : [w_1]_{\mathcal{E}} = [w_2]_{\mathcal{E}} \Rightarrow s_1 = s_2$$

2.2 Transition Steps

In what follows, Σ denotes a LTS and \wr an independency relation over Σ .

¹ For a binary relation $R \subset U \times U$, R^C denotes the binary relation $(U \times U) \setminus R$

2.2.1 Transition Steps (definition)

The transition set E defines a **transition step** wrt \wr iff $\forall t_1, t_2 \in E : t_1 \wr t_2$. In the sequel, $Step(T, \wr)$ denotes the set of all transition steps derived from T wrt \wr . When no confusion is possible, we note $Step(T)$ instead of $Step(T, \wr)$.

Property: $E \in Step(T, \wr), F \subset E \Rightarrow F \in Step(T, \wr)$

2.2.2 Traces and transition steps

Set of sequences associated with a transition step

Let Seq be the mapping $\mathcal{P}(T) \mapsto \mathcal{P}(T^*)$ defined by:

$$Seq(\emptyset) = \{\epsilon\} \text{ and } Seq(E) = \bigcup_{e \in E} Seq(E \setminus \{e\}) \otimes e$$

where for $\Omega \in \mathcal{P}(T^*)$ and $w \in T^* : \Omega \otimes w = \{\omega.w : \omega \in \Omega\}$

Property: $\forall E \in Step(T, \wr), \omega_1 \text{ and } \omega_2 \in Seq(E) \Rightarrow [\omega_1]_{(T, \wr)} = [\omega_2]_{(T, \wr)}$

Trace of a transition step The former property allows to define the transition step trace as the trace of a string associated with the step by means of function Seq . For $E \in Step(T, \wr)$, $[E]_{(T, \wr)} =_{def} [\mathcal{E}]_{(T, \wr)}$ where $\mathcal{E} \in Seq(E)$

Trace of a sequence of transition steps is defined as the trace of the string obtained by concatenation of the strings associated with each step of the sequence: For $E_1.E_2 \dots E_n \in Step(T, \wr)^*$,

$$[E_1.E_2 \dots E_n]_{(T, \wr)} =_{def} [\mathcal{E}_1.\mathcal{E}_2 \dots \mathcal{E}_n]_{(T, \wr)} \text{ where } \mathcal{E}_i \in Seq(E_i)$$

2.2.3 Extension of the reachability relation to transition steps

Let \rightarrow , be the extension of \rightarrow to transition steps defined by:

$$\forall s, s' \in S, \forall E \in Step(T) : s \xrightarrow{E} s' \text{ iff } \forall e \in E : s \xrightarrow{e} \text{ and } \exists \omega \in Seq(E) : s \xrightarrow{\omega} s'$$

2.2.4 Extensions of \wr to string and transition steps

Support of a string: Let $\|\cdot\|$, be the mapping $T^* \mapsto \mathcal{P}(T)$ defined by:

$$\|\epsilon\| = \emptyset \text{ and } \|u.\omega\| = \{u\} \cup \|\omega\|$$

Extensions of \wr

Let $\wr \subset Step(T) \times Step(T)$ defined by $T_1 \wr T_2$ iff $T_1 \times T_2 \subset \wr$

Let $\wr \subset Step(T) \times T^*$ defined by $T_1 \wr w$ iff $T_1 \times \|w\| \subset \wr$

Let $\wr \subset T^* \times T^*$ defined by $w_1 \wr w_2$ iff $\|w_1\| \times \|w_2\| \subset \wr$

2.2.5 Diamond properties

1. For $t \in T, w \in T^* : \text{If } s \xrightarrow{t}, s \xrightarrow{w} \text{ and } t \wr w \text{ then } s \xrightarrow{t.w} s' \text{ and } s \xrightarrow{w.t} s'$
2. For $w_1, w_2 \in T^* : \text{If } s \xrightarrow{w_1}, s \xrightarrow{w_2} \text{ and } w_1 \wr w_2 \text{ then } s \xrightarrow{w_1.w_2} s' \text{ and } s \xrightarrow{w_2.w_1} s'$
3. For $w \in T^*, E \in Step(T) : \text{If } s \xrightarrow{w}, s \xrightarrow{E} \text{ and } E \wr w \text{ then } s \xrightarrow{\mathcal{E}.w} s' \text{ and } s \xrightarrow{w.\mathcal{E}} s' \forall \mathcal{E} \in Seq(E)$
4. For $E_1, E_2 \in Step(T) : \text{If } s \xrightarrow{E_1}, s \xrightarrow{E_2} \text{ and } E_1 \wr E_2 \text{ then } s \xrightarrow{E_1 \cup E_2} s'$
And $s \xrightarrow{\mathcal{E}_1.\mathcal{E}_2} s' \text{ and } s \xrightarrow{\mathcal{E}_2.\mathcal{E}_1} s' \forall \mathcal{E}_i \in Seq(E_i)$

Sketches of proofs: (1) by recurrence on $|w|$. (2) by recurrence on $|w_i|$ and property (1). (3) is a consequence of (2) ($E \wr w \Rightarrow \mathcal{E} \wr w$). (4) $E_1 \wr E_2 \Rightarrow E_1 \cup E_2 \in \text{Step}(T)$ and property 3.

2.3 Covering Step Graph

This section introduces the Covering Step Graph definition. The general reachability properties preserved by Step Graph are presented.

2.3.1 Covering Step Graph (definition) Let $\Sigma = \langle S, s_o, T, \rightarrow \rangle$, be a LTS \wr an independency relation over Σ (we let $\# = \wr^C$)

$\Sigma = \langle S', s_o, \mathcal{T}, \rightsquigarrow \rangle$ is a Covering Step Graph for Σ , wrt \wr iff

- (1) $S' \subset S$ (2) $\mathcal{T} \subset \text{Step}(T, \wr)$

(3) $\forall s, s' \in S \cap S' : s \rightsquigarrow^E s' \text{ implies } s \xrightarrow{E} s'$

(4) $\forall s \in S \cap S', \forall s' \in S' :$

$$s \xrightarrow{\omega} s' \text{ implies } \begin{cases} \exists s'' \in S', \exists \omega' \in T^*, \exists P_{\omega.\omega'} \in \mathcal{T}^* : \\ s' \xrightarrow{\omega'} s'', s \rightsquigarrow^{P_{\omega.\omega'}} s'' \text{ and } [w.\omega']_{(T,\#)} = [P_{\omega.\omega'}]_{(T,\#)} \end{cases}$$

note: any LTS may be viewed as a CSG by considering $\wr = \emptyset$

Milner's Scheduler This scheduler is described in [Mil 85]. n sites cyclically execute action a_i then action b_i . A scheduler constrains the execution of the whole system in such a way that the n sites alternatively perform actions a_1, a_2, \dots, a_n . Figure 1 depicts the net of the whole system (scheduler + n sites), then the corresponding LTS, and a derived CSG.

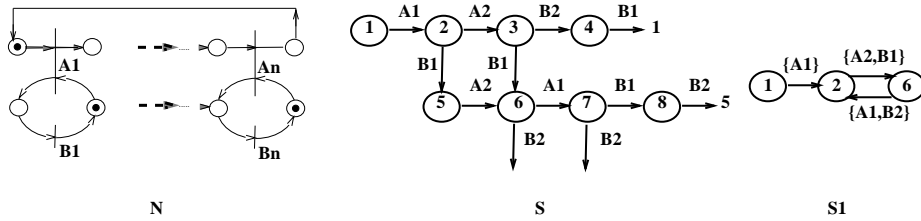


Fig.1. Milner's Scheduler

Condition (1), each state of the step graph is a state of the standard LTS. For the Milner's scheduler, $t_1 \wr t_2$ iff $t_1 \neq t_2$, consequently each subset of T constitutes a step of transitions and (2) holds. Condition (3) means that each transition step in the CSG ($S1$) corresponds to a firing sequence in the standard LTS (S): for instance $:2 \rightsquigarrow^{\{A2,B1\}} 6$ corresponds to $2 \xrightarrow{A2,B1} 6$ (or $2 \xrightarrow{B1,A2} 6$).

Finally, condition (4) expresses a "covering condition" between the firing sequences of the standard LTS and the step sequences of the CSG.

In S , $1 \xrightarrow{a_1 \cdot a_2 \cdot b_2 \cdot b_1} 1$, condition 4 of definition 2.3.1 ensures that a “covering” step sequence exists in S . The initial sequence, $a_1 \cdot a_2 \cdot b_2 \cdot b_1$, is extended by a_1 , in the CSG the following step sequence $\Omega = \{a_1\} \cdot \{a_2, b_1\} \cdot \{a_1, b_2\}$ covers it: $1 \xrightarrow{\Omega} 2$ and $1 \xrightarrow{a_1 \cdot a_2 \cdot b_2 \cdot b_1 \cdot a_1} 2$ with $[a_1 \cdot a_2 \cdot b_2 \cdot b_1 \cdot a_1] = [\{a_1\} \cdot \{a_2, b_1\} \cdot \{a_1, b_2\}]$

Section 3.2 presents examples satisfying conditions 1), 2) and 3) and violating condition 4)

2.3.2 Reachability Preservation This section shows how general reachability properties such as liveness or deadlock may be checked directly on CSG.

Liveness

$\Sigma = \langle S, s_o, T, \rightarrow \rangle$, is E-Live for $E \subset T$ iff $\forall s \in S, \forall e \in E, \exists \sigma \in T^*$ such that $s \xrightarrow{\sigma, e}$
 $\mathcal{L} = \langle S', s_o, \mathcal{T}, \rightsquigarrow \rangle$ is E-Live iff $\forall s \in S, \forall e \in E, \exists \sigma \in \mathcal{T}^*, \exists u \in \mathcal{T}$ such that $s \xrightarrow{\sigma, u}$
with $e \in u$

property If \mathcal{L} is a CSG for Σ then \mathcal{L} is E-Live iff Σ is E-Live

(\Leftarrow) (resp (\Rightarrow)) is a direct consequence of condition 3 (resp 4) of definition 2.3.1

Deadlock preservation Let $Sink(\Sigma) =_{def} \{s \in S : s \not\rightarrow^t \ \forall t \in T\}$

Property: If \mathcal{L} is a CSG of Σ then $Sink(\Sigma) = Sink(\mathcal{L})$

$Sink(\Sigma) \subset Sink(\mathcal{L})$ (resp $Sink(\mathcal{L}) \subset Sink(\Sigma)$): By absurd by considering a state $\in S' - Sink(\mathcal{L})$ (resp $\in S' - Sink(\Sigma)$) and by applying condition 4 (resp 3) of definition 2.3.1, this leads to a contradiction

3 On the Fly Covering Step Graph Derivation

This section introduces a basic algorithm for the covering step graph derivation directly from the description of the system. The general conditions to be fulfilled are first described. These conditions are shown to be sufficient conditions. The corresponding complete algorithm is then presented.

3.1 Basic Principles

Table 1 describes the general structure of the basic algorithm. This algorithm is similar to a standard algorithm for computing a reachable marking graph, that is $Enabled(q)$ computes the set of transitions enabled by state q , and $fire(q, t)$ computes the state obtained from state q by firing transition t .

The changes with respect to a standard algorithm are the following.

- The independence relation \wr is supplied,
- The enable transitions are split in two subsets by means of functions T_U and T_M :
 - T_u , defines transitions to be explored in a standard way,
 - T_m defines transitions whose exploration will be conducted within a step. In the sequel, such transitions will be referred as “mergeable” transitions.
- The set of transition steps Π_{T_M} built by means of function Π , whose domain

<pre> 1. Initialise: <i>Stack</i> is empty ; push s_0 onto <i>Stack</i> H is empty ; enter s_0 in H A is empty ; 2. Loop : while $Stack \neq \emptyset$ loop { pop(q) from stack $T \leftarrow Enabled(q)$; IF $T = \emptyset$ Then Print "Deadlock" ELSE{ $T_u \leftarrow T_U(T, l)$ $T_m \leftarrow T_M(T, l)$ $\Pi_{T_m} \leftarrow \Pi(T_m, l)$ $\forall \pi \in \Pi_{T_m}$ do { $q'' \leftarrow q$; $\forall p \in \pi$ do { $q' = fire(q'', p)$; $q'' \leftarrow q'$; enter $\langle q, s, q'' \rangle$ in A; if $q'' \in H$ then {enter q'' in H; put q' onto <i>Stack</i>} } $\forall t \in T_u$ do { $q' = fire(q, t)$; enter $\langle q, \{t\}, q' \rangle$ in A if $q' \in H$ then {enter q' in H; put q' onto <i>Stack</i>} } } </pre>

Table 1. General algorithm

is the set of mergeable transitions T_m

- The implementation of a firing step of transitions.

3.2 Sufficient Conditions

Some sufficient conditions are now defined on the sets T_u , T_m , and Π_{T_m} . These conditions have to be satisfied in order to ensure that the algorithm of Table 1 produces indeed a CSG. These conditions are parameterised by a transitive conflict relation weaker than the initial one. This transitive relation, denoted $\#\#$, is such that $\# \subset \#\#$, consequently $\#\#^C \subset \wr$.

3.2.1 Sufficient Conditions Table 2 describes the conditions which lead to a CSG derivation. Conditions CA_1 , CA_2 deal with elements T_u and T_m , respectively. Conditions CB_1 , CB_2 concern the set of transition steps Π_{T_m} .

Condition CA_1 defines a partition of the enable transition set. Condition CA_2 implies that no (initial) conflict may occur between a mergeable transition and a disable transition.

$\forall q \in S :$ $(CA_1) T_m \cup T_u = Enabled(q) \text{ and } T_m \cap T_u = \emptyset$ $(CA_2) \text{ If } t \in T_m \text{ then } t' \# t \Rightarrow t' \in Enabled(q)$ $(CB_1) \forall \pi \in \Pi_{T_m} : \pi \in Step(T, \#^C)$ $(CB_2) \forall P \in Step(Enabled(q), \#^C), \exists \pi \in \Pi_{T_m} : P \subseteq \pi$

Table 2. Sufficient conditions on T_u, T_m and Π_{T_m}

Condition CB_1 implies that no (direct or indirect) conflict exists between transitions within a step. Condition CB_2 implies that the considered computation steps *subsume* any subset of strongly independent transitions. Conditions CA_2, CB_1 , and condition CB_2 are illustrated in subsections 3.2.2, 3.2.3, respectively.

3.2.2 Confusion Case Petri net N and its associated LTS S depicted by Figure 2 represent a *confusion* case [Rei 85]: Transition D is in conflict with transitions A_1, B_1 . These latter transitions are independent. From state 1 of the LTS, transitions A_1 et B_2 may occur. When B_2 first occurs, D may then occur; however, when A_1 first occurs, D cannot happen.

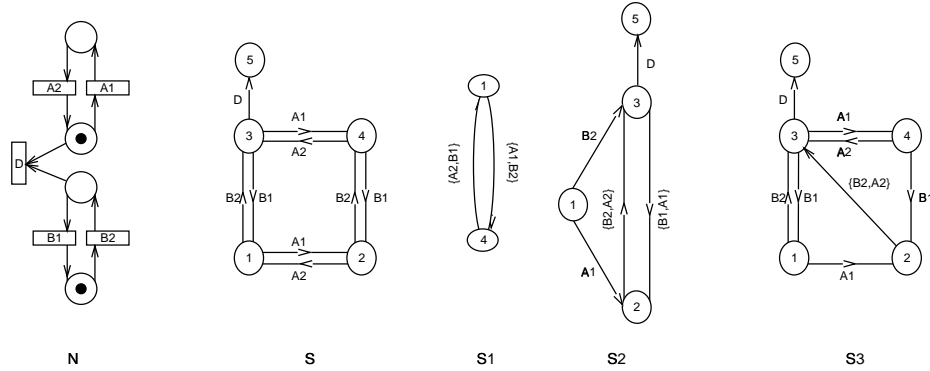


Fig. 2. Confusion Case: Illustration of Conditions CA_2 et CB_1

Motivation for CA_2 :

Consider LTS S_1 (Fig 2). From state 1, computation step $\{A_1, B_2\}$ is built, transition A_1 is in a (potential) conflict with (still disable) transition D . The atomic computation step A_1, B_2 *hides* some internal states, namely states 2 and

3. In state 3, transitions A_1, B_1 , and D are simultaneously enable. Step $\{A_1, B_2\}$ would prevent the firing of transition D , and a deadlock state would be omitted. In this case, “covering” condition (def 2.3.1.4) is made false.

Motivation for CB_1 : Consider LTS S_2 (Fig 2). From state 3, computation step $\{A_1, B_1\}$ might be derived, whereas A_1, B_1 are in an un-direct conflict, through D (i.e. $\{A_1, B_1\} \in Step(T, t)$ but $A_1 \# D$ and $B_1 \# D$)

The sequence $1 \xrightarrow{B_2.B_1.B_2.D} 5$ in S (cf. Fig 2) is not “covered” in S_2 , in the sense of definition 2.3.1. Because state 5 is a deadlock state, a sequence from state 1 to state 5 must exist. The single sequence of length 4 in S_2 is the following²:
 $1 \xrightarrow{\{A_1\}.\{B_2, A_2\}.\{D\}} 5$ or $[\{A_1\}.\{B_2, A_2\}.\{D\}] \neq [B_2.B_1.B_2.D]$.
 S_3 is a CSG for S .

3.2.3 Maximal Step Coverage The net depicted by Figure 3 consists of two independent conflict sets. The associated LTS is represented by LTS S . CSG S_2 is a correct one with respect to the net, but step graph S_1 is an incorrect one, because it does not satisfy condition CB_2 . With respect to S , sequence $0 \xrightarrow{e_1, e_4} 2$ must be a possible sequence of any correct associated CSG. This is not the case for step graph S_1 : $0 \xrightarrow{\{e_1, e_3\}} 2$ and $0 \xrightarrow{\{e_2, e_4\}} 2$ but $[\{e_1, e_3\}] \neq [e_1.e_4]$ and $[\{e_2, e_4\}] \neq [e_1.e_4]$

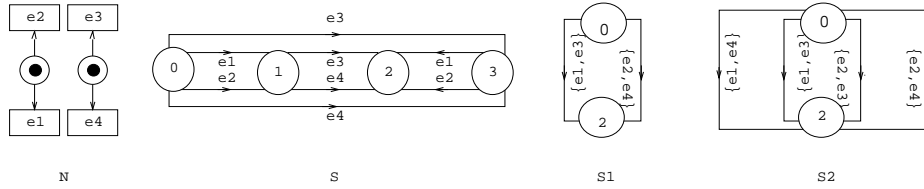


Fig. 3. Wrong Coverage: Illustration of condition CB_2

3.3 Sufficient Conditions of Step Graph Computation

The basic algorithm, depicted by Table 1 leads to the correct CSG computation, as far as the conditions given by Table 2 hold. By using the notations of Table 1, the conditions to be verified are given by Table 3.

3.3.1 Proof of conditions (1), (2), and (3)

(2) and (3) are immediate consequences of conditions CA_1, CB_1 and of the relation $\#^C \subset \{$

² Trace equality ensures that the two sequences have the same length

$s \xrightarrow{E} s'$ means $\langle s, E, s' \rangle \in A$ and $\mathbf{T} = \{E \in \mathcal{P}(T) : \langle s, E, s' \rangle \in A\}$		
(1) $H \subset S$	(2) $\mathbf{T} \subset \text{Step}(T, \iota)$	(3) $s \xrightarrow{E} s'$ implies $s \xrightarrow{E} s'$
$\exists s'' \in S, \exists \omega' \in T^*, \exists P_{\omega, \omega'} \in \mathbf{T}^* :$		
(4) $s \xrightarrow{\omega} s'$ implies		
$s' \xrightarrow{\omega'} s'', s \xrightarrow{P_{\omega, \omega'}} s''$ where $[w, w']_{(T, \iota)} = [P_{\omega, \omega'}]_{(T, \iota)}$		

Table 3. Verification Conditions

When $s \xrightarrow{E} s'$, two cases have to be considered:

either $E = \{t\}$ then $s \xrightarrow{t} s'$ and $\{t\} \in \text{Step}(T, \iota)$ or $E = \{t_1, t_2, \dots, t_k\}$ then $E \in \text{Step}(\text{Enabled}(s), \iota)$ (cf CB_1) and $s \xrightarrow{E} s'$ (cf prop 2.2.3)
The enumeration starts with $H = \{s_0\}$ ($s_0 \in S$), (2) et (3) imply that any state derived from a state in H is indeed a S state, condition (1) consequently holds.

3.3.2 Proof of condition (4) The following lemma is required; the proof of this lemma is shown in section 3.3.3.

Factorisation Lemma $\forall s \in S \cap S', \forall w \in T^* : s \xrightarrow{w} s' \Rightarrow$ (a) or (b) where

(a) $\exists t \in T_u, s \xrightarrow{t, w_1} s'$ with $[w] = [t, w_1]$

(b) $\exists F \in \Pi_{T_m}, \exists E \subset F : s \xrightarrow{E, w_1} s'$ with $[w] = [E, w_1]$ and $(F \setminus E) \upharpoonright |w_1|$

Proof of (4) is performed by induction on $|\omega|$ (obvious when $|\omega| = 0$)

the property is assumed to hold for rank k , and let sequence ω be such that $|\omega| = k + 1$. Factorisation lemma is valid and two cases occur:

Case (a) is obvious: $t \in t_u$ implies $s \xrightarrow{\{t\}} s_1$ and $s_1 \xrightarrow{w_1} s'$, the induction hypothesis is then applied on s_1 .

Case (b): $\exists F \in \Pi_{T_m}, \exists E \subset F : s \xrightarrow{E, w_1} s'$ with $[w] = [E, w_1]$ and $(F \setminus E) \upharpoonright |w_1|$

Let $R = F \setminus E$, then $s \xrightarrow{R, w_1} s'$ with $R \upharpoonright w_1$ and $E \upharpoonright R$

The diamond property (2.2.5.3) implies: for $r \in \text{Seq}(R)$, then $s \xrightarrow{r, w_1} s''$ and $s \xrightarrow{E, w_1, r} s''$ (cf fig 4.a)

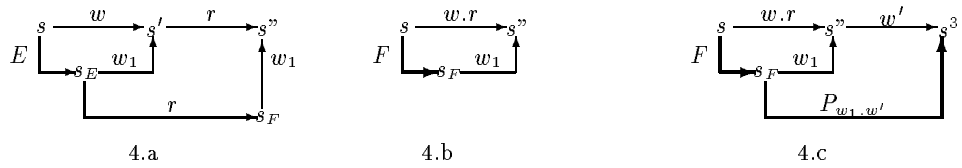


Fig. 4. Case (b)

The case depicted by Figure 4.b is a consequence of $E \oplus R = F$, where $[w.r] = [F.w_1]$. From state s_F , $s_F \xrightarrow{w_1} s''$ where $|w_1| \leq k$ and the induction hypothesis may be used: $\exists w' \in T^*, \exists P_{w_1.w'} \in Step(T)^* : s_F \xrightarrow{w_1.w'} s^3, s_F \xrightarrow{P_{w_1.w'}} s^3$ and $[w_1.w'] = [P_{w_1.w'}]$ (cf fig 4.c).

Finally, $s \xrightarrow{w.r.w'} s^3$ and $s \xrightarrow{F.P_{w_1.w'}} s^3$. Because of $[w.r] = [F.w_1]$ and $[w_1.w'] = [P_{w_1.w'}]$ the result $[w.r.w'] = [F.P_{w_1.w'}]$, holds.

3.3.3 Proof of Factorisation Lemma For technical reasons, *factorisation* operator π is first introduced. This operator extracts from any transition sequence the maximal computation step, or the maximal step prefix. Operator π is defined according to the selected conflict relation.

π *Definition:* Let π be the mapping $S \times Step(T) \times T^* \times T^* \mapsto Step(T) \times T^*$ such that

$$\begin{aligned} \pi(s, E, w, \epsilon) &= (E, w) \\ \pi(s, E, w_1, w_2) &= (E, w_1.w_2) \text{ if } E \in \Pi_{T_m} \\ \pi(s, E, w_1, t.w') &= \begin{cases} \pi(s, E \cup \{t\}, w_1, w') & \text{if } t \in T_m, \{t\} \upharpoonright |w_1| \\ & \text{and } E \cup \{t\} \in Step(T, \#\#^C) \\ (\{t\}, \mathcal{E}.w_1.w') & \text{if } t \in T_u, \{t\} \upharpoonright |w_1| \text{ and } \mathcal{E} \in Seq(E) \\ \pi(s, E, w_1, t.w') & \text{otherwise} \end{cases} \end{aligned}$$

π Properties

Let $s \in S, w, w_1 \in T^*$ be such that $s \xrightarrow{w} s'$ and $\pi(s, \emptyset, \epsilon, w) = (E, w_1)$ Then (1) and (2)

- (1) $s \xrightarrow{E} s_1, s_1 \xrightarrow{w_1} s'$ & where $[w] = [E.w_1]$
- (2) $\forall t \in |w_1| : t \xrightarrow{s} \Rightarrow \exists t' \in E : t \#\# t'$

proof of factorisation lemma:

Let $s \in S, w, w_1 \in T^*$ be such that $s \xrightarrow{w} s'$ and $\pi(s, \emptyset, \epsilon, w) = (E, w_1)$

As a result of the π construction, three cases occur: $E = \{t\}$ and $t \in T_u, E \in \Pi_{T_m}, E \notin \Pi_{T_m}$. The first two cases are similar, and the result directly follows from π property (1). The third case $E \notin \Pi_{T_m}$ needs to be developed.

By construction, $E \in Step(T, \#\#^C)$, condition CB_2 implies that transition subset $F \in \Pi_{T_m}$ exists, verifying $E \subset F$; π property (1) implies the trace equality, the relation $F \setminus E \upharpoonright w_1$ is the last to be proved.

By absurd, let R be $F \setminus E$ and assume that there exists transition $t' \in |w_1|$ in conflict with $t \in R$. Because of $t \in R \Rightarrow s \xrightarrow{t}$, condition CA_2 implies $s \xrightarrow{t'}$ and π property (2) then holds: $\exists t'' \in E$ and $t' \equiv_{\#} t''$. Finally, $t, t'' \in F$ such that $t \#\# t''$ and $F \in Step(T, \#\#^C)$, this is a contradiction.

3.4 Step Graph of Crossed Conflicts

This section presents a particular conflict relation and definitions of functions T_M, T_U , and Π providing an algorithm for the step graph derivation, which is a specific instantiation of the former basic algorithm (cf. Table 1).

3.4.1 Weak Conflict and Strong Independence Let relation $\equiv_{\#}$ be the reflexive and transitive closure of relation $\#$. This relation is called *weak conflict relation*. Relation $\equiv_{\#}$ is an equivalence relation and $\# \subset \equiv_{\#}$. Relation $\equiv_{\#}^C$, i.e. the complement relation, is called strong independence relation and $\equiv_{\#}^C \subset \wr$

3.4.2 Crossed Conflict Step An easy way to obey to rule CB_2 is to consider only maximal computation steps. Since conflict relation is an equivalence relation ($\equiv_{\#}$), the associated conflict sets supply a partition of the transition set. The set of maximal computation steps results from the *orthoproduct* [PF 90] of the equivalence classes of relation $\equiv_{\#}$

Orthoproduct : Let \mathbb{E} be a set of sets, $\mathbb{E} = \{E_1, E_2, \dots, E_n\}$ $\mathbb{E} \in \mathcal{P}(\mathcal{P}(U))$
 $\Pi_C(\mathbb{E}) =_{def} \{\{e_1, e_2, \dots, e_n\} : (e_1, e_2, \dots, e_n) \in E_1 \times E_2 \dots \times E_n\}$

Crossed Conflict Step: The set $\Pi_C(T_m / \equiv_{\#})$ is called Crossed Conflict Step.

Properties:

- $E \in \Pi_C(T_m / \equiv_{\#}) \Rightarrow E \in Step(T_m, \equiv_{\#}^C)$ a fortiori $E \in Step(T_m, \wr)$
- $E \in \Pi_C(T_m / \equiv_{\#}), t \notin E \Rightarrow E \cup \{t\} \notin Step(T_m, \equiv_{\#}^C)$
these properties follow from Π_C definition, and from $\equiv_{\#}^C \subset \wr$

3.4.3 Crossed Conflict Step Construction

The final algorithm is derived from the basic one (cf table 1) by considering the following functions:

$$\begin{aligned} T_M(T, \wr) &= \{t \in T : t' \# t \Rightarrow t' \in T\} \\ T_U(T, \wr) &= T \setminus T_M(T, \wr) \\ \Pi(T_M(T, \wr)) &= \Pi_C((T, \wr) / \equiv_{\#}) \end{aligned}$$

Property : This algorithm produces a CSG. Conditions CA_1, CA_2 et CB_1 hold (trivial). Condition CB_2 results from properties of crossed conflict steps.

Examples: The former (right) CSG are computed by this algorithm. In the case of Milner's scheduler, (section 2.3.1), the gain is exponential: for n sites, the standard LTS consists of $n \times 2^n$ states and $(n^2 + n) \times 2^{n-1}$ transitions, the CSG consists of $n + 1$ states and $n + 1$ transitions.

4 Covering Step Graphs and Observational Equivalence

This section shows how to adapt the previous algorithm in order to generate, according to a set of observable events T_{Obs} , a covering step graph observationally equivalent (\equiv) to the standard LTS (i.e the LTS underlying the generated

step graph and the standard LTS are \equiv). The observational equivalence definition is recalled, an additional condition required for observational equivalence is given and shown to be sufficient. Finally, a specific algorithm for producing observational equivalent step graphs is supplied.

4.1 Observational Equivalence - Weak Bisimulation

Observation: Observational Equivalence is defined with respect to a set of observable events T_{Obs} . The definition of Observational Equivalence introduces a particular label τ representing internal (un-observable) computation and new transition relation \Rightarrow is defined as follows:

$$\begin{aligned} s \xrightarrow{\tau^*} s' \text{ iff } \exists w \in (T \setminus T_{Obs})^* : s \xrightarrow{w} s', \\ \text{and for all observable label } a, s \xrightarrow{\tau^* a \tau^*} s' \text{ iff } s \xrightarrow{\tau^*} s_1, s_1 \xrightarrow{a} s_2 \text{ and } s_2 \xrightarrow{\tau^*} s' \\ \text{In the sequel, } s \xrightarrow{\tau^* \epsilon \tau^*} s' \text{ stands for } s \xrightarrow{\tau^*} s' \end{aligned}$$

Observational equivalence (over finite LTS) may be defined as the limit of a decreasing sequence of equivalence relations over the states of the LTS ($\equiv_n \subset S_1 \times S_2$) [Mil 85]

$$\begin{aligned} \equiv &= \bigcap_{n \geq 0} \equiv_n \text{ where } \equiv_0 = S_1 \times S_2 \\ s \xrightarrow{\tau^* a \tau^*} s' &\Rightarrow \exists q' : q \xrightarrow{\tau^* a \tau^*} q' \text{ and } s' \equiv_{n-1} q' \text{ (i)} \\ \text{and } s \equiv_n q &\text{ iff } \forall a \in T_{Obs} \cup \{\epsilon\} \\ q \xrightarrow{\tau^* a \tau^*} q' &\Rightarrow \exists s' : s \xrightarrow{\tau^* a \tau^*} s' \text{ and } s' \equiv_{n-1} q' \text{ (ii)} \end{aligned}$$

Weak-Bisimulation: A binary relation B over $S_1 \times S_2$ is a weak bisimulation iff

$$\begin{aligned} s \xrightarrow{\tau^* a \tau^*} s' &\Rightarrow \exists q' : q \xrightarrow{\tau^* a \tau^*} q' \text{ and } (s', q') \in B \text{ (i)} \\ (s, q) \in B &\text{ iff } \forall a \in T_{Obs} \cup \{\epsilon\} \\ q \xrightarrow{\tau^* a \tau^*} q' &\Rightarrow \exists s' : s \xrightarrow{\tau^* a \tau^*} s' \text{ and } (s', q') \in B \text{ (ii)} \end{aligned}$$

property: Observational Equivalence is the largest Weak-Bisimulation.

Extensions to LTS For $i \in \{1, 2\} : \Sigma_i = \langle S_i, s_{0i}, T_i, \rightarrow_i \rangle$

$$\Sigma_1 \equiv \Sigma_2 \text{ iff } s_{01} \equiv s_{02}$$

Σ_1 and Σ_2 are weak-bisimilar iff there exists a weak-bisimulation B containing (s_{01}, s_{02})

4.2 Sufficient conditions for observational equivalence

The basic algorithm (table 1) for step graphs generation and the associated sufficient conditions (table 2) are maintained. A new condition is introduced with respect to the set of observed events. Notations are those of table 2.

Free Transitions

Let $\#(t)$ be the transition subset which are in conflict with t :

$$\#(t) =_{def} \{t' \in T : t' \# t\}$$

and $Free(T, \#)$ the transition subset without conflict:

$$Free(T, \#) =_{def} \{t \in T : \#(t) = \{t\}\}$$

Additional condition for observational equivalence: Each “mergeable” transition t has to be included in a step transition π such that: all transitions in π (possibly except t) are conflict free and according whether t is observable or not, then either t is the single observable transition in π or any transition in π is unobservable. More precisely, condition CB_3 has to be fulfilled: $\forall t \in T_m, \exists \pi \in \Pi_{T_m} : t \in \pi, \pi \cap T_{Obs} = T_{Obs} \cap \{t\} \ \& \ \pi \setminus \{t\} \subset Free(T, \#)$

CB_3 admits the two following consequences:

c_1 : Within a step, there exists at most a single observable transition or a conflict transition (a transition not belonging to $Free(T, \#)$).

c_2 : For each unobservable (enabled) transition, there exists a step containing it, and this step is composed of only free and unobservable transitions.

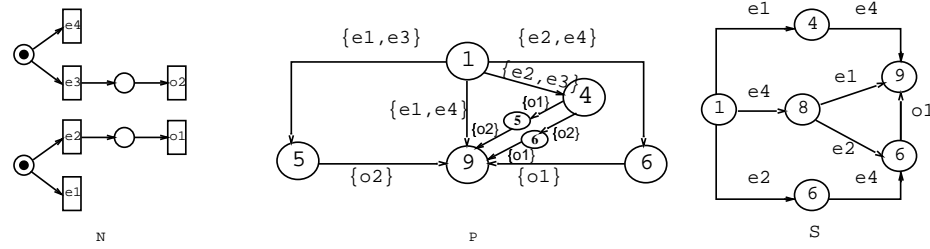


Fig. 5. Step Graph not observationally equivalent: example 1

Example 1: Figure 5 depicts a net where two “independent conflicts” are initially enabled, P , a step graph for which some steps contain two conflict transitions ($e2$ and $e4$ are independent but $e2, e4 \in Free(T_m, l)^C$, then c_1 is violated) and S a fragment of the complete LTS associated with the net. Observed events are o_1 and o_2 .

State 8 of LTS S is considered. 8 admits the following experiments: $8 \xrightarrow{\tau^*} 9$ and $8 \xrightarrow{\tau^* o_1 \tau^*} 9$. State 8 does not belong to the states of the step graph P and no state of P is observationally equivalent to 8. Now, we consider state 1, a common state of S and P . In S , $1 \xrightarrow{\tau^*} 8$ while in P $1 \xrightarrow{\tau^*} q$ where $q \in \{1, 4, 5, 6, 9\}$. Since $q \not\equiv 8$ for each $q \in \{1, 4, 5, 6, 9\}$, we conclude that $S \not\equiv P$.

Example 2: Figure 6 depicts net N , step graph P associated with and fragment S of the whole LTS associated with the net. Observed events are o_1 , o_2 and o_3 . State 3 of LTS S is considered. 3 admits the following experiments: $3 \xrightarrow{\tau^*} 9$ et $3 \xrightarrow{\tau^* o_3 \tau^*} 9$. State 3 does not belong to the states of P and no state of P is observationally equivalent to 3. Now, we consider state 1, a common state of S

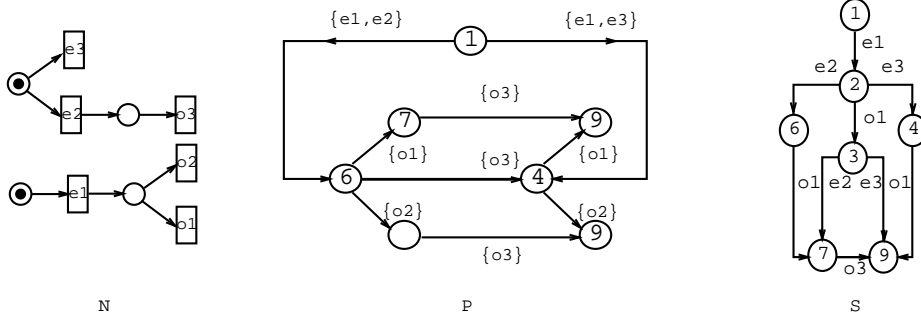


Fig. 6. Step Graph not observationally equivalent: Example 2

and P . In S , $1 \xrightarrow{\tau^* o_1 \tau^*} 3$ while in P $1 \xrightarrow{\tau^* o_1 \tau^*} q$ where $q \in \{7, 9\}$. Since $3 \not\equiv 7$ and $3 \not\equiv 9$, we conclude that $S \not\equiv P$.

c_2 (and CB_3) is violated in state 1 of P : transition e_1 is enabled in 1 but no step containing e_1 is composed of only free transitions³.

4.3 Observational Equivalence between Σ and \mathcal{L}

This section shows that the basic algorithm, depicted by table 1, furnishes an observational equivalent step graph, as far as the conditions of table 2 (sufficient conditions for a step graph generation) and condition CB_3 described in 4.2 (specific to observational equivalence) hold.

The proof is organised as follows: The two next sections show basic properties of free transitions with respect to observational equivalence. Section 4.3.3 shows that for each state, common to the standard LTS and its step graph, each “classical” experiment ($s \xrightarrow{w} q$) may be reproduced by means of \xrightarrow{w} , that is $s \xrightarrow{w} q'$ where $q \equiv q'$ and \equiv is the observational equivalence defined on the standard LTS. This result is used in section 4.3.4 to build a bisimulation relation between the standard LTS and its step graph.

4.3.1 Diamond properties over free transitions

Property: For $s, s', q \in S, w \in T^* : s \xrightarrow{w} q$ and $t \in Free(T, \#)$

If $s \xrightarrow{t} s'$ then $(s' \xrightarrow{w} q' \text{ and } q \xrightarrow{t} q')$ or $(w = w_1.t.w_2 \text{ and } s' \xrightarrow{w_1.w_2} q)$

proof: If $t \parallel w$ then the result is an immediate consequence of diamond property 2.2.5. Now, consider the case where $t \# w$. Since $t \in Free(T, \#) \Rightarrow t \in \parallel w \parallel$, w may be rewritten as $w_1.t.w_2$ where $t \parallel w_1$. The result follows by applying diamond property 2.2.5.

Corollary: For $s, q \in S, a \in T_{Obs} : s \xrightarrow{\tau^* a \tau^*} q$ and $t \in Free(T, \#), t \notin T_{Obs}$

If $s \xrightarrow{t} s'$ Then $(s' \xrightarrow{\tau^* a \tau^*} q' \text{ and } q \xrightarrow{t} q')$ or $(s' \xrightarrow{\tau^* a \tau^*} q)$

³ ($e_1 \in \{e_1, e_2\}$ and $e_1 \in \{e_1, e_3\}$ but $\{e_2, e_3\} \subset Free(T, \#)^C$)

4.3.2 Free Transitions and Observational Equivalence

Property: If $s \xrightarrow{t} s', t \in Free(T, \#)$ and $t \notin T_{Obs}$ then $s \equiv s'$

proof: direct consequence of the former corollary.

Corollary: If $s \xrightarrow{\sigma} s'$ with $\sigma \in Free(T, \#)^*$ and $\|\sigma\| \cap T_{Obs} = \emptyset$ Then $s \equiv s'$

proof by induction on $|\sigma|$

4.3.3 Experiment Preservation

Hiding Morphisms

For $T_{Obs} \subset T$ and $\mathbb{T} \subset \mathcal{P}(T)$ verifying : $\forall E \in \mathbb{T} \text{ Card}(E \cap T_{Obs}) \leq 1$, we define:

$$F_{Obs} \text{ a morphism } T^* \mapsto T_{Obs}^* \text{ by: } F_{Obs}(o.w) = \begin{cases} o.F_{Obs}(w) & \text{if } o \in T_{Obs} \\ \epsilon.F_{Obs}(w) & \text{otherwise} \end{cases}$$

$$\text{and } IF_{Obs} : \mathbb{T}^* \mapsto T_{Obs}^* \text{ by } IF_{Obs}(P_1.P_2) = \begin{cases} o.IF_{Obs}(P_2) & \text{if } P_1 \cap T_{Obs} = \{o\} \\ \epsilon.IF_{Obs}(P_2) & \text{otherwise} \end{cases}$$

Experiment preservation in Σ

For $s \in S \cap S', s \xrightarrow{w} q \Rightarrow s \xrightarrow{W} q'$ with $F_{Obs}(w) = IF_{Obs}(W)$ and $q \equiv q'$ in Σ

Proof by induction on $|w|$. Obvious when $|w| = 0$, the property is assumed to hold when $|w| = k$ and let sequence $t.w$, state s_1 such that $s \xrightarrow{t} s_1$

According whether $t \in T_u$ or $t \in T_m$, two cases occur.

- If $t \in T_u$ then $s \xrightarrow{t} s_1$ and induction hypothesis may be applied from s_1 implying that $s_1 \xrightarrow{W} q'$ where $q \equiv q'$ and $F_{Obs}(w) = IF_{Obs}(W)$, Since $IF_{Obs}(t.W) = IF_{Obs}(t).IF_{Obs}(W)$ the result follows.

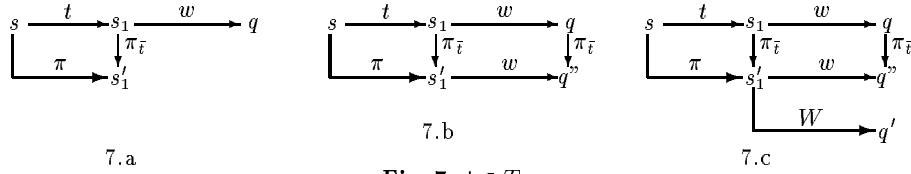


Fig. 7. $t \in T_m$

- If $t \in T_m$ then because of condition CB_3 :

$$\exists \pi \in \Pi_{T_m} : t \in \pi, (\pi \setminus \{t\}) \subset Free(T, \#) \text{ \& } F_{Obs}(t) = IF_{Obs}(\pi)$$

(Fig 7.a) Let s'_1 such that $s \xrightarrow{\pi} s'_1$, the following holds: $s_1 \xrightarrow{\pi_{\bar{t}}} s'_1$ where $\pi_{\bar{t}} \in Seq(\pi \setminus \{t\})$ (cf def 2.3.1). Since $\pi \setminus \{t\} \subset Free(T, \#)$, corollary 4.3.2 is valid and it follows that $s_1 \equiv s'_1$.

(Fig 7.b) We have $s_1 \xrightarrow{w} q, s_1 \xrightarrow{\pi_{\bar{t}}} s'_1$ and $\pi_{\bar{t}} \wr w$ then $s'_1 \xrightarrow{w} q'$ and $q \xrightarrow{\pi_{\bar{t}}} q''$ (cf prop 4.3.1). Moreover, $q \xrightarrow{\pi_{\bar{t}}} q'' \Rightarrow q \equiv q''$ (corollary 4.3.2)

Finally, as depicted in Fig 7.c, $s'_1 \in S \cap S'$ and $s'_1 \xrightarrow{w} q''$ with $|w| = k$, and the induction hypothesis may be applied from s'_1 implying that $s'_1 \xrightarrow{W} q'$, $q' \equiv q''$ and $F_{Obs}(w) = F_{Obs}(W)$. Since $q'' \equiv q$ and $F_{Obs}(t) = F_{Obs}(\pi)$, the result follows.

Corollary: For $s \in S \cap S'$, $a \in T_{Obs} \cup \{\epsilon\}$: $s \xrightarrow{\tau^* a \tau^*} q \Rightarrow s \xrightarrow{\tau^* a \tau^*} q'$ and $q \equiv q'$ in Σ

4.3.4 Σ and \mathcal{L} are Weak-Bisimilar

Let B , be the subset of $S \times S'$ defined by $B = \{(q, q') \in S \times S' : q \equiv q' \text{ in } \Sigma\}$
property: B is a bisimulation between Σ and \mathcal{L} . (proof is a direct consequence of corollary 4.3.3 and CSG definition)

Corollary: Σ and \mathcal{L} are observationally equivalent

4.3.5 Effective Algorithm In this section a specific conflict relation and specific functions T_M, T_U and Π are proposed to obtain an algorithm for the derivation of an observational equivalent step graph which is a specific instantiation of the former basic algorithm (cf Table 1).

1. Functions T_M and T_U , introduced in section 3.4.3, are maintained:
 $T_M(T, \iota) = \{t \in T : t \# t \Rightarrow t' \in T\}$ and $T_U(T, \iota) = T \setminus T_M(T, \iota)$
2. Conflict equivalence relation $\#$ is defined by the following partition of T_m :
$$T_m = \bigcup_{t \in Free(T_m, \#) \setminus T_{Obs}} \{t\} \oplus ((T_{Obs} \cap T_m) \cup Free(T_m, \#)^C)$$

Within this partition, each un-observable free transition leads to a block reduced to itself, and a last block gathers all the transitions which are observable or involved in a conflict. As a consequence, equivalence relation $\#$ is weaker than the initial one ($\# \subset \#$).
3. $\Pi(T_M(T, \iota)) = \Pi_C((T_M(T, \iota)) / \#) \cup (Free(T_M(T, \iota)) \setminus T_{Obs})$

Property: This algorithm computes a step graph which is weak-bisimilar (w.r.t T_{Obs}) to the standard LTS

Step Graph derivation Since $\#$ is transitive and subsumes $\#$, associated functions T_M, T_U and Π fulfils sufficient conditions expressed in table 2 for the step graph derivation (cf prop 3.4.3). Note that condition CB_2 holds as soon as $\Pi_C((T_M(T, \iota)) / \#) \subset \Pi(T_M(T, \iota))$ and the addition of step $Free(T_M(T, \iota)) \setminus T_{Obs}$ does not make any change.

Weak-bisimulation w.r.t T_{Obs} : $\Pi(T_M(T, \iota))$ fulfils CB_3 because of (1) the definition of partition $\#$, (2) the definition of $\Pi(T_M(T, \iota))$ by means of the orthoproduct and (3) the introduction of step $Free(T_M(T, \iota)) \setminus T_{Obs}$.

5 Evaluation and Conclusion

Evaluation: The considered example is a data base system presented in [Jen 87] as illustrative example for a state space reduction using equivalent markings.

The same example has been used in [Val 89] to illustrate state space reduction method using stubborn set.

The data base system consists in $n \geq 2$ managers and a mechanism ensuring mutual exclusion for critical operations. Initially, all managers are *idle* and the base is *open*. The modification of the value by a manager is modelled by transition “update and send messages” ($usm(k)$) for which a message is sent to each partners of manager k . The sending manager (k) waits until all other managers have received his message ($rm(k, p)$), performed an update and sent an acknowledgement ($sa(k, p)$). When all acknowledgements are available, the sending manager returns to idle and the base to open ($ra(k)$).

Fig 8.a depicts the general structure of the whole state space in the case of n managers. $Cube_k$ represents the interleaved execution of transitions $rm(k, p)$ and $sa(k, p)$ for $p \in [1, N], p \neq k$. Fig 8.b depicts “cube” 1 in the case of 3 managers. For this system, pair of transitions in conflict have the following form: (t, t) or $(usm(i), usm(j))$ or $(usm(i), rm(i, j))$. Consequently each “cube” will be represented in a step graph by 3 states and 2 transition Steps $(\{rm(1,2), rm(1,3)\})$ and $(\{sa(1,2), sa(1,3)\})$ for the cube depicted Fig 8.a). Finally the whole Step graph for a date base consisting of n managers is composed of $3n + 1$ states and $4n$ edges.

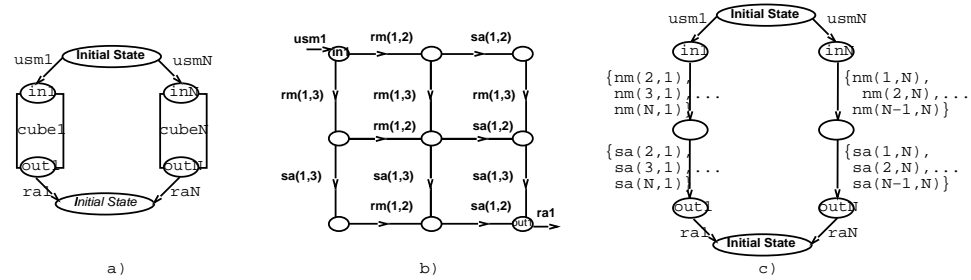


Fig. 8. Covering Step Graph for a n managers data base

Comparative Results: The following table gives the state space size for n managers, first when no reduction is performed, then with a partial exploration by means of stubborn sets [Val 89], with a symbolic exploration by means of equivalent markings [Jen 87], finally with a partial and symbolic exploration.

	states	edges
no reduction	$n3^{n-1} + 1$	$\approx n^2 \times 3^{n-2}$
Stubborn sets	$2n^2$	$2n^2$
Equivalent markings	$n^2/2$	n^2
Stubborn sets + Equivalent markings	$2n$	$2n$
Covering Step Graph	$3n + 1$	$4n$
Covering Step Graph + Equivalent markings	4	4

All the different techniques allow the size of the state space to be reduced from exponential to polynomial. Covering Step graphs take advantage over stubborn sets and equivalent markings: the size is linear for Step graphs, while it is quadratic when stubborn sets or Equivalent markings are used. Finally, as mentioned in [Val 89], “Symbolic” methods (using symmetries) and “Partial Order” methods (using independence relation) are complementary: for this particular example, an exploration by means of Step graphs and equivalent markings furnishes the optimal result since the size of the reduced state space is constant.

Behavioural analysis using Covering Step Graphs:

The same example is used to illustrate the reduction power of the proposed approach in the framework of Weak-Bisimulation. Now the step graph derivation is conducted with respect to a set of observable events. Roughly speaking, observable events are chosen according to the set of properties to be analysed [HM 85]. In the context of the present example, two kinds of observation are considered: first, observed events are relative to critical operations ($T_{Obs_1} = \{usm(k) : \forall k \in [1, N]\} \cup \{ra(k) : \forall k \in [1, N]\}$) allowing mutual exclusion property to be verified, second, the “local service” of each manager ($T_{Obs_2}(k) = \{usm(k), RM(k), SA(k)\}$)⁴ is considered.

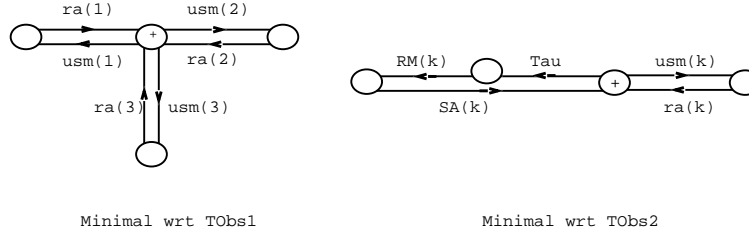


Fig. 9. Minimal Weak Equivalent

In the case of T_{Obs_1} , the obtained step graph is those obtained by general step graph derivation (Fig. 8.c). It consists in $3n + 1$ states and $4n$ edges while the minimal weak bisimilar LTS consists in $n + 1$ states and $2n$ transitions (Fig 9)

The following table summarises the obtained results for T_{Obs_2} . In this case, the minimal weak bisimilar LTS is constant (4 states and 5 edges) while the size of the obtained equivalent step graph seems quadratic in k .

K	3	4	6	10
no reduction	28/42	109/224	1459/1872	590.491/11.810.000
Equivalent step graph	10/12	13/16	19/24	58/94
minimal weak	4/5	4/5	4/5	4/5

⁴ In order to have a better reduction with observational equivalence: events $rm(k, j)$ and $sa(k, j)$ for $k \neq j$ are respectively renamed by $RM(k)$ and $SA(k)$

Conclusion and Future works: This paper has presented Covering Step Graphs as an alternative to LTS. CSG permits the verification of general reachability properties (deadlock detection, liveness) while Weak Bisimilar CSG (derived with respect to a set of observable events) permits the verification of specific properties expressed in Hennesy-Milner Logic [HM 85]. Two algorithms are proposed for an “on the fly generation” of such graphs. Two standard examples permit us to obtain a first evaluation of our approach: the obtained results are promising, but significant experiments remain necessary. With respect to behavioural analysis, other equivalences (maximal traces, co-simulation, ...) may be also considered and associated step graph derivation algorithms proposed. Finally, the interest of step graphs in the framework of “True concurrency” semantics may be envisaged.

References

- [FM 90] J. FERNANDEZ, L. MOUNIER *Verifying Bisimulation on the Fly*
3 rd. Int. Conf on Formal Description Techniques, Madrid, 1990
- [GW 91] P. GODEFROID, P. WOLPER
Using Partial Orders for efficient verification of deadlock freedom and safety properties In Computer Aided Verification, 1991, LNCS 575
- [GP 93] P. GODEFROID, D. PIROTIN
Refining Dependencies Improves Partial-Order verification methods
In Computer Aided Verification, 1993, LNCS 697
- [Esp 93] J. ESPARZA
Model Checking using net unfoldings In TAPSOFT'93, 1993, LNCS 668
- [HM 85] M. HENNESSY, R. MILNER *Algebraic Laws for Nondeterminism and Concurrency* Journal of the A.C.M Volume 32 1985
- [Jen 87] K. JENSEN *Coloured Petri Nets*.
In Brauer, W., Reisig, W. & Rozenberg, G. (Ed.): Petri Nets: Central Models and their Properties. Advances in Petri Nets LNCS 254
- [PF 90] D. H. PITT, D. FREESTONE *The derivation of conformance tests from lotos specifications* IEEE Transactions on Software Engineering, 16(12), 1990
- [McMil 95] K. L. McMILLAN *Trace theoretic verification of asynchronous circuits using unfoldings* In Computer Aided Verification, 1995, LNCS 939
- [Maz 87] A. MAZURKIEWICZ *Trace Theory* In “Petri Nets: Applications and Relationship to other models of concurrency” LNCS 255
- [Mil 85] R. MILNER *Communication and Concurrency* Prentice Hall.
- [Rei 85] W. REISIG *Petri Nets: an Introduction*
EATCS, Monographs on Theoretical Computer Science, Springer Verlag, 1985
- [Val 89] A. VALMARI *Stubborn Sets for reduced state space generation*
10 th Int. Conf on Application and Theory of Petri Nets, Bonn, 1989, LNCS 483
- [WG 93] P. WOLPER, P. GODEFROID *Partial Order Methods for Temporal Verification*
Proceedings of CONCUR'93, LNCS 715