On Boolean Encodings of Transition Relation for Parallel Compositions of Transition Systems*

Extended abstract

Andrzej Zbrzezny

IMCS, Jan Długosz University in Częstochowa,
Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland.
a.zbrzezny@ajd.czest.pl

Abstract. We present and compare different Boolean encodings of the transition relation for the parallel composition of transition systems, both for the asynchronous and the synchronous semantics. We compare the encodings considered by applying them to the SAT-based bounded model checking (BMC) of ECTL* properties. We provide experimental results which show that our new encoding for the asynchronous semantics significantly increases the efficiency of the SAT-based BMC.

Keywords: transition system, parallel composition, SAT-based bounded model checking, Boolean encoding.

1 Introduction

One of the most important practical problems in model checking is the exponential growth of number of states of the transition system, depending on the number of components of the modelled system. Edmund M. Clarke, co-inventor of the model checking, stressed that the problem of overcoming the exponential explosion in the number of states has been one of the most important research questions he was dealing with since the inventing of model checking [1–3]. Reducing the negative impact of the exponential explosion of the state space requires in particular the methods and algorithms to have the highest possible efficiency.

The aim of this paper is to present and compare different Boolean encodings of the transition relation for the parallel composition of transition systems, both for the asynchronous and the synchronous semantics. We provide experimental results which show that our new encoding for the asynchronous semantics significantly increases the efficiency of the SAT-based bounded model checking.

2 Preliminaries

2.1 Transition systems

Transition systems ([4]) are often used as models to describe the behaviour of systems. They are basically directed graphs where nodes represent states, and edges model transi-

* Partly supported by National Science Center under the grant No. 2011/01/B/ST6/05317.

tions, i.e., state changes. A state describes some information about a system at a certain moment of its behaviour.

Definition 1. *A* transition system *is a tuple $\mathcal{M} = (S, Act, \longrightarrow, I, AP, L)$, where S is a nonempty finite set of states, Act is a set of actions, $I \subseteq S$ is the set of initial states, $\longrightarrow \subseteq S \times Act \times S$ is a transition relation, AP is a set of atomic propositions, and $L : S \to 2^{AP}$ is a* labelling function *that assigns to each state a set of atomic propositions that are assumed to be true at that state. Transition systems are also called* models.

For convenience, we write $s \xrightarrow{\sigma} s'$ instead of $(s, \sigma, s') \in \longrightarrow$. Moreover, we write $s \longrightarrow s'$ if $s \xrightarrow{\sigma} s'$ for some $\sigma \in Act$.

From now on we assume that transition systems are finite, i.e. the sets S, Act and AP are finite We also assume that a transition system has no terminal states, i.e. for every $s \in S$ there exist $s' \in S$ such that $s \longrightarrow s'$. The set of all natural numbers is denoted by $\mathbb{N}$ and the set of all positive natural numbers by $\mathbb{N}_+$. A *path* in $\mathcal{M}$ is an infinite sequence $\rho = (s_0, s_1, \ldots)$ of states such that $s_j \longrightarrow s_{j+1}$ for each $j \in \mathbb{N}$.

2.2 Parallel compositions of transition systems

Definition 2. *Let J be a non-empty, finite set of indices and let $\{\mathcal{M}_j \mid j \in J\}$ be a family of transition systems, i.e. $\mathcal{M}_j = (S_j, Act_j, \longrightarrow_j, I_j, AP_j, L_j)$. Assume that $AP_i \cap AP_j = \emptyset$, for $i \neq j$. Moreover, let $J(\sigma) = \{j \in J \mid \sigma \in Act_j\}$, and $\varepsilon \notin Act_j$ for each $j \in J$.*

1. *The* asynchronous parallel composition *of the family $\{\mathcal{M}_j \mid j \in J\}$ of transition systems is the transition system $\mathcal{M} = (S, Act, \longmapsto, I, AP, L)$ such that $S = \prod_{j \in J} S_j$, $Act = \bigcup_{j \in J} Act_j$, $I = \prod_{j \in J} I_j$, $AP = \bigcup_{j \in J} AP_j$, $L = \bigcup_{j \in J} L_j$, and $\longmapsto$ is defined as follows: for every $s, s' \in S$ and every $\sigma \in Act$: $s \xmapsto{\sigma} s'$ if and only if the following conditions hold:*
   (a) $s_j \xrightarrow{\sigma}_j s'_j$ for $j \in J(\sigma)$,
   (b) $s'_j = s_j$ for $j \in J \setminus J(\sigma)$.
2. *The* synchronous parallel composition *of the family $\{\mathcal{M}_j \mid j \in J\}$ of transition systems is the transition system $\mathcal{M} = (S, \overline{Act}, \longrightarrow, I, AP, L)$ such that $S = \prod_{j \in J} S_j$, $\overline{Act} = \prod_{j \in J} (Act_j \cup \{\varepsilon\})$, $I = \prod_{j \in J} I_j$, $AP = \bigcup_{j \in J} AP_j$, $L = \bigcup_{j \in J} L_j$, and $\longrightarrow$ is defined as follows: for every $s, s' \in S$ and every $\overline{\sigma} \in \overline{Act}$: $s \xrightarrow{\overline{\sigma}} s'$ if and only if the following conditions hold:*
   (a) $(\forall j \in J)\ (\sigma_j = \varepsilon \implies s'_j = s_j)$,
   (b) $(\exists j \in J)((\sigma_j \neq \varepsilon)$,
   (c) $(\forall j \in J)\ (\sigma_j \neq \varepsilon \implies s_j \xrightarrow{\sigma_j}_j s'_j)$,
   (d) $(\forall i \in J)(\sigma_i \neq \varepsilon \implies (\forall j \in J(\sigma_i))\ \sigma_j = \sigma_i)$.

Recall that if $|J(\sigma)| > 1$ for an action $\sigma \in \bigcup_{j \in J} Act_j$, then the action σ is called a *shared* action; otherwise, i.e. if $|J(\sigma)| = 1$, it is called a *local* action. The actions from $\overline{Act}$ are called *joint* actions.

In the transition system $\mathcal{M}$ being the asynchronous parallel composition of a family of transition systems only one local or shared action may be performed at a given time in a given global state of $\mathcal{M}$. Moreover, a shared action σ has to be performed in all the components $\mathcal{M}_j$, for $j \in J(\sigma)$.

In the transition system $\mathcal{M}$ being the synchronous parallel composition of a family of transition systems a joint action $\overline{\sigma} \in \overline{Act}$ is to be performed by the system at a given time in a given global state of $\mathcal{M}$. It means, that every component $\mathcal{M}_j$ of the system can perform an action (also an empty action ε). Notice however, that at least one component has to perform a non-empty action. Moreover, if one of the components performs a shared action σ, then the action σ has to be performed in all the components $\mathcal{M}_j$, for $j \in J(\sigma)$.

Observe that if $s \longmapsto s'$ then there exists $\overline{\sigma} \in \overline{Act}$ such that $(s, \xrightarrow{\overline{\sigma}}, s')$. Thus every path of the asynchronous parallel composition of transition systems is also a path of the synchronous parallel composition of transition systems. It follows that every ECTL^* formula valid in the asynchronous parallel composition of a given family of transition systems is also valid in the synchronous parallel composition of this family. Note that the converse of the implication does not hold. However, let us note that each formula of the form $\mathbf{E}(\mathbf{F}\psi_1 \wedge \ldots \wedge \mathbf{F}\psi_m)$, where $\psi_1, \ldots, \psi_m$ are propositional formulae, valid in the synchronous parallel composition of the given family of transition systems is also valid in the asynchronous parallel composition of this family.

3 Boolean Encodings of States, Actions and Transition Relations

We start with describing the Boolean encoding of states, actions and the transition relation of a given model by means of propositional formulae, which are built over a set PV of propositional variables plus the constants $\mathbf{true}$ and $\mathbf{false}$, with help of the propositional connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$.

Let $\{\mathcal{M}_j \mid j \in J\}$ be a finite family of transition systems and let $\mathcal{M}$ be the parallel composition (either asynchronous or synchronous) of $\{\mathcal{M}_j \mid j \in J\}$. Since the set S_j of states of each $\mathcal{M}_j$ is finite, every element of S_j can be encoded as a bit vector of the length $\lceil \log_2 |S_j| \rceil$. Therefore, each state of $\mathcal{M}_j$ can be represented by a valuation of a vector $\mathbf{w}_j$ (called a *symbolic local state*) of unique *propositional state variables*. Then, each state of $\mathcal{M}$ can be represented by a valuation of a vector $\mathbf{w}$ (called a *symbolic global state*) of the vectors $\mathbf{w}_j$.

Similarly, since the set Act_j of actions of each $\mathcal{M}_j$ is finite, the set $\overline{Act}$ is also finite, and it follows that every element of Act_j can be encoded as a bit vector of the length $\lceil \log_2 |Act| \rceil$. Therefore, each action of $\mathcal{M}_j$ can be represented by a valuation of a vector $\mathbf{a}_j$ (called a *symbolic action*) of unique *propositional action variables*. Then, each element of the set $\overline{Act}$ can be represented by a valuation of a vector $\mathbf{a}$ (called a *symbolic joint action*) of the vectors $\mathbf{a}_j$.

Let $SV = \{w_1, w_2, \ldots\}$ be a set of *propositional state variables* and $AV = \{v_1, v_2, \ldots\}$ be a set of *propositional action variables*. We assume that $SV \cap AV = \emptyset$.

Moreover, let $PV = SV \cup AV$ and $V : PV \to \{0,1\}$ be a *valuation of propositional variables* (a *valuation* for short). For every $m, r \in \mathbb{N}_+$ each valuation V induces the functions $\mathbf{S} : SV^m \to \{0,1\}^m$ and $\mathbf{A} : AV^r \to \{0,1\}^r$ defined as follows:

$$\mathbf{S}(w_{j_1}, \ldots, w_{j_m}) = (V(w_{j_1}), \ldots, V(w_{j_m}))$$
$$\mathbf{A}(v_{j_1}, \ldots, v_{j_r}) = (V(v_{j_r}), \ldots, V(v_{j_r}))$$

Our aim is to define either a Boolean formula $\mathcal{T}(\mathbf{w}, \mathbf{w}')$ so that for each valuation $V \in \{0,1\}^{SV}$, V satisfies $\mathcal{T}(\mathbf{w}, \mathbf{w}')$ iff $\mathbf{S}(\mathbf{w}) \longmapsto \mathbf{S}(\mathbf{w}')$ in $\mathcal{M}$ or a Boolean formula $\mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}')$ so that for each valuation $V \in \{0,1\}^{PV}$, V satisfies $\mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}')$ iff $\mathbf{S}(\mathbf{w}) \stackrel{\mathbf{S}(\mathbf{a})}{\longrightarrow} \mathbf{S}(\mathbf{w}')$ in $\mathcal{M}$.

3.1 The standard Boolean encoding of asynchronous parallel composition

This encoding is implemented in all the SAT-based verification modules of VerICS [5–10]. In this encoding we use the following auxiliary propositional formulae:

- $\mathcal{H}_j(\mathbf{w}_j, \mathbf{w}'_j)$ for $j \in J$ is a Boolean formula defined so that for each valuation $V \in \{0,1\}^{SV}$, V satisfies $\mathcal{T}(\mathbf{w}_j, \mathbf{w}'_j)$ iff $\mathbf{S}(\mathbf{w}'_j) = \mathbf{S}(\mathbf{w}_j)$;
- $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{w}'_j)$ for $j \in J$ and $\sigma \in Act_j$ is a Boolean formula defined so that for each $V \in \{0,1\}^{SV}$, V satisfies $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{w}'_j)$ iff $\mathbf{S}(\mathbf{w}_j) \stackrel{\sigma}{\longrightarrow} \mathbf{S}(\mathbf{w}'_j)$ in $\mathcal{M}_j$.

Now it is possible to define the Boolean formula $\mathcal{T}(\mathbf{w}, \mathbf{w}')$

$$\mathcal{T}(\mathbf{w}, \mathbf{w}') = \bigvee_{\sigma \in Act} \left(\bigwedge_{j \in J(\sigma)} \mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{w}'_j) \ \wedge \bigwedge_{j \notin J(\sigma)} \mathcal{H}_j(\mathbf{w}_j, \mathbf{w}'_j) \right)$$

which symbolically encodes the transition relation $\longmapsto$ of the asynchronous parallel composition of the family $\{\mathcal{M}_j \mid j \in J\}$.

3.2 The Boolean encoding of synchronous parallel composition

We have recently implemented this encoding in the module BMC4ELTLK of VerICS. This module was used for performing experimental results presented in the article [11], which was recently accepted for publication. However, the encoding in question was not described in [11]. In this encoding we use the following auxiliary propositional formulae:

- $\mathcal{H}_\sigma(\mathbf{a}_j)$ for $j \in J$ and $\sigma \in Act_j \cup \{\varepsilon\}$ is a Boolean formula defined so that for each valuation $V \in \{0,1\}^{AV}$, V satisfies $\mathcal{H}_\sigma(\mathbf{a}_j)$ iff $\mathbf{A}(\mathbf{a}_j) = \sigma$;
- $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j)$ for $j \in J$ and $\sigma \in Act_j$ is a Boolean formula defined so that for each valuation $V \in \{0,1\}^{PV}$, V satisfies $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j)$ iff $\mathbf{A}(\mathbf{a}_j) = \sigma$ and $\mathbf{S}(\mathbf{w}_j) \stackrel{\sigma}{\longrightarrow} \mathbf{S}(\mathbf{w}'_j)$ in $\mathcal{M}_j$.

Now we can define the Boolean formula $\mathcal{T}_j(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j)$

$$\mathcal{T}_j(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j) = \big(\mathcal{H}(\mathbf{w}_j, \mathbf{w}'_j) \wedge \mathcal{H}_\varepsilon(\mathbf{a}_j)\big) \ \vee \bigvee_{\sigma \in Act_j} \mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j)$$

which symbolically encodes the transition relation $\longrightarrow_j$ of $\mathcal{M}_j$. Eventually, we define the Boolean formula $\mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}')$ which symbolically encodes the transition relation $\longrightarrow$ of the synchronous parallel composition of the family $\{\mathcal{M}_j \mid j \in J\}$.

$$\mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}') = \bigwedge_{j \in J} T_j(\mathbf{w}_j, \mathbf{a}_j, \mathbf{w}'_j) \wedge \bigwedge_{\sigma \in Act} \left(\bigwedge_{j \in \mathbf{J}(\sigma)} \mathcal{H}_\sigma(\mathbf{a}_j) \ \vee \bigwedge_{j \in \mathbf{J}(\sigma)} \mathcal{H}_\varepsilon(\mathbf{a}_j) \right)$$

The first subformula of the above conjunction assures that executing a joint action $\overline{\sigma}$ consists of executing all the actions that are the components of $\overline{\sigma}$. The second subformula requires that each action σ being a component of a joint action has to be executed in all the components of the synchronous parallel composition in which it appears.

3.3 A new Boolean encoding of asynchronous parallel composition

This encoding is inspired by the encoding for the synchronous parallel composition. However, it is not directly derived from the encoding for the synchronous parallel composition. In the new encoding for asynchronous parallel composition $\mathbf{b}$ stands for a vector (of the length $\lceil \log_2 |Act| \rceil$) of propositional action variables so that every action in Act can be be represented by a valuation of propositional variables in $\mathbf{b}$. In the encoding we use the following auxiliary propositional formula:

- $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j)$ for $j \in J$ and $\sigma \in Act_j$ is a Boolean formula defined so that for each valuation $V \in \{0,1\}^{PV}$, V satisfies $\mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j)$ iff $\mathbf{A}(\mathbf{b}) = \sigma$ and $\mathbf{S}(\mathbf{w}_j) \xrightarrow{\sigma} \mathbf{S}(\mathbf{w}'_j)$ in $\mathcal{M}_j$.

Now we can define the Boolean formula $\mathcal{T}_j(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j)$ which symbolically encodes the transition relation $\longrightarrow_j$ of $\mathcal{M}_j$:

$$\mathcal{T}_j(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j) = \left(\mathcal{H}(\mathbf{w}_j, \mathbf{w}'_j) \wedge \bigwedge_{\sigma \in Act_j} \neg \mathcal{H}_\sigma(\mathbf{b}) \right) \ \vee \bigvee_{\sigma \in Act_j} \mathcal{T}_{j,\sigma}(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j)$$

Intuitively, $\mathcal{T}_j(\mathbf{w}_j, \mathbf{b}, \mathbf{w}'_j)$ assures that either $\mathbf{A}(\mathbf{b}) \notin Act_j$ and $\mathbf{S}(\mathbf{w}_j) = \mathbf{S}(\mathbf{w}'_j)$ (i.e. no action is performed in $\mathcal{M}_j$ in a given step) or $\mathbf{S}(\mathbf{w}_j) \xrightarrow{\mathbf{A}(\mathbf{b})} \mathbf{S}(\mathbf{w}'_j)$ in $\mathcal{M}_j$.

Eventually, we define the Boolean formula $\mathcal{T}(\mathbf{w}, \mathbf{b}, \mathbf{w}')$ which symbolically encodes the transition relation $\longmapsto$ of the synchronous parallel composition of the family $\{\mathcal{M}_j \mid j \in J\}$:

$$\mathcal{T}(\mathbf{w}, \mathbf{b}, \mathbf{w}') = \bigwedge_{j \in J} \mathcal{T}_j(\mathbf{w}, \mathbf{b}, \mathbf{w}') \ \wedge \bigvee_{\sigma \in Act} \mathcal{H}_\sigma(\mathbf{b})$$

The first subformula of the above conjunction assures that for each action from Act the following condition is satisfied: if this action is to be performed in one of the components of the asynchronous parallel composition, then it has to be performed in all the components in which it appears. The second subformula requires that the vector $\mathbf{b}$ represents an action $\sigma \in Act$.

4 Experimental Results

Our experiments were performed on a computer equipped with Intel Xeon 2 GHz processor, 4GB of RAM and the operating system Ubuntu Linux Server with the kernel 3.5.0. We set the default limits of 1 GB of memory and 1800 seconds. Moreover, we used PicoSAT [12] in version 957 to test the satisfiability of the propositional formulae generated by the module BMC4ECTLS [9].

As the first benchmark we used the generic pipeline paradigm (GPP) introduced in [13]. The GPP consists of the Producer that is able to produce data, the Consumer that is able to receive data, and a chain of n intermediate Nodes that are able to receive, process, and send data. We assume that the following local states ProdReady, Ready_j and ConsReady are initial, respectively, for Producer, Node_j and Consumer. The global system is obtained as the parallel composition of the components, which are shown in Figure 4.

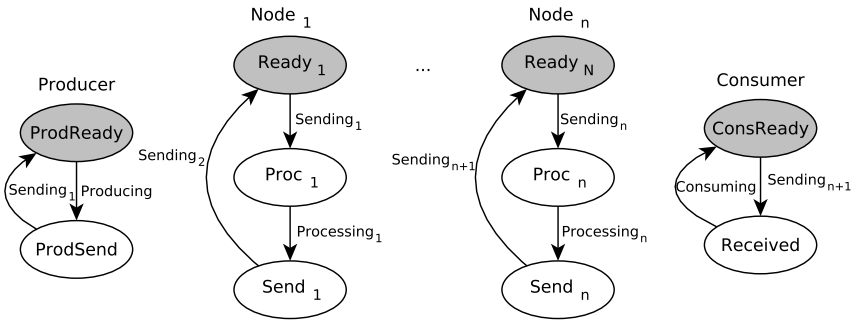

Fig. 1. The automata for the Sender, the Nodes and the Producer.

As the second benchmark we used the train controller system (TC) introduced in [14]. The TC consists of a controller, and n trains (for $n \geqslant 2$). Each train uses its own circular track for travelling in one direction. Eventually, each train has to pass through a tunnel, but because there is only one track in the tunnel, the trains arriving from each direction cannot use it simultaneously. There are signals on both sides of the tunnel, which can be either red or green. Each train notifies the controller when it request entry to the tunnel or when it leave the tunnel. The controller controls the colour of the displayed signal. The local state Away_j is initial for Train_j, and the local state Green is initial for Controller. The global system is obtained as the parallel composition of the components, which are shown in Figure 4.

As the third benchmark we used the dining philosophers problem [15, 16]. We have modelled this problem by means of a parallel composition of $2n + 1$ transition systems. The global system consists of n transition system each of which models a philosopher, together with n transition systems each of which models a fork, together with one transition system which models the lackey. The latter transition system is used to

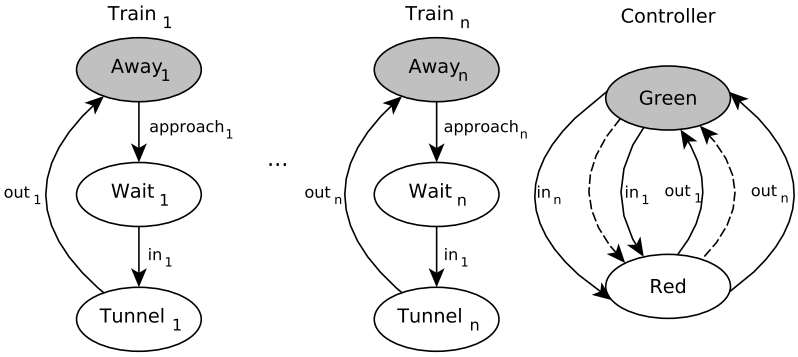

Fig. 2. The automata for the Trains and the Controller.

coordinate the philosophers' access to the dining-room. In fact, this automaton ensures that no deadlock is possible. The global system is obtained as the parallel composition of the components, which are shown in Figure 3.

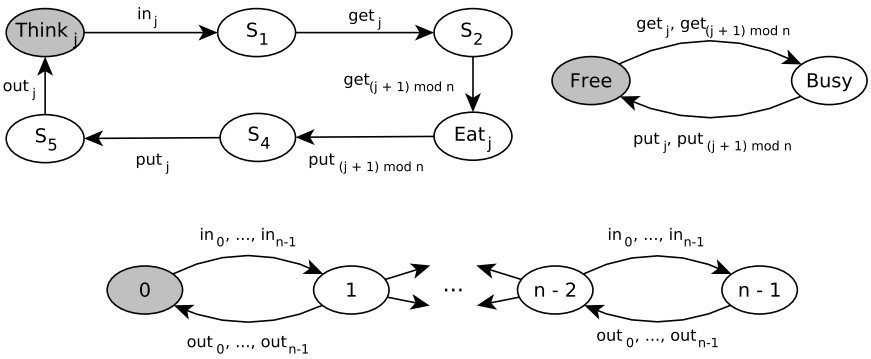

Fig. 3. The automata for the j-th Philosopher, the j-th Fork and the Lackey.

Let $\mathcal{I}(\mathbf{w})$ be a propositional formula such that for each valuation $V \in \{0,1\}^{SV}$, V satisfies $\mathcal{I}(\mathbf{w})$ iff $\mathbf{S}(w)$ is an initial state of the model. In order to compare experimental results for the three Boolean encodings of transition relation we have tested first (on a symbolic path of the length 1) the ECTL* formula $\varphi_0 = \mathbf{E}(\mathbf{true})$ that is valid in the models considered. The translation of the formula φ_0 results in the propositional formula:

– $\mathcal{I}(\mathbf{w}) \wedge \mathcal{T}(\mathbf{w}, \mathbf{w}')$ for the old encoding of the asynchronous parallel composition,

- $\mathcal{I}(\mathbf{w}) \wedge \mathcal{T}(\mathbf{w}, \mathbf{a}, \mathbf{w}')$ for the encoding of the synchronous parallel composition,
- $\mathcal{I}(\mathbf{w}) \wedge \mathcal{T}(\mathbf{w}, \mathbf{b}, \mathbf{w}')$ for the new encoding of the asynchronous parallel composition.

As we expected, the experimental results for the formula φ_0 (Table 1) show that the new encoding of the transition relation for the asynchronous parallel composition significantly influence the size (i.e. the number of clauses) of the resulting propositional formulae: for the GPP and 1800 nodes the number of clauses for the new encoding is 56 times less than for the old one; for the TC and 2800 trains the number of clauses for the new encoding is 25 times less than for the old one; for the DP and 1250 philosophers the number of clauses for the new encoding is 26 times less than for the old one.

Moreover, the results from Table 1 show that the encoding of the transition relation for the synchronous parallel composition results in propositional formulae that are shorter than formulae for the old encoding and significantly longer than formulae for the new encoding.

Encoding of the transition relation	(*Max) number of components	Number of variables	Number of clauses
Generic Pipeline Paradigm			
New	*105000	3727243	10131599
Sync	*28000	3689932	8130256
New	28000	989629	2688769
Old	*1800	3283218	9833442
Sync	1800	190792	426140
New	1800	63577	172637
Train Controller			
New	*87000	3829207	10617501
Sync	*29000	4262182	9800524
New	29000	1320132	3670282
Old	*2800	3991404	11949004
Sync	2800	346605	809243
New	2800	131235	365609
Dining Philosophers			
New	*2400	288650	805804
Sync	*2350	812694	1994856
New	2350	283150	790554
Old	*1150	3374711	10137875
Sync	1150	376831	927590
New	1150	138855	387680

Table 1. Results for the formula φ_0 generated by the SAT-based BMC translations

Although, in our opinion, testing the formula φ_0 for different systems allows to draw reliable conclusions about the presented encodings, we have also tested some for-

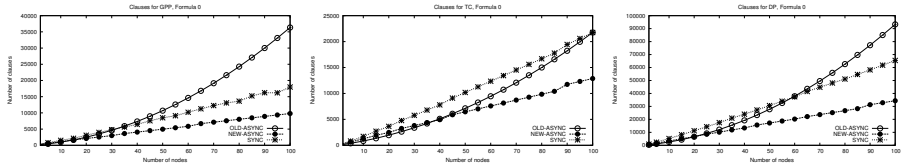

Fig. 4. Number of clauses for the formula φ_0.

mulae each of which expresses either a reachability or an extended reachability property (i.e. a formula of the form $\mathbf{E}(\mathbf{F}\psi_1 \wedge \ldots \wedge \mathbf{F}\psi_m)$, where $\psi_1, \ldots, \psi_m$ are propositional formulae). For all the tested systems we assume that for every local state s, $L(s) = \{s\}$.

For the GPP we have tested the following two formulae:

- $\varphi_1(n) = \mathbf{EF}(Received)$
- $\varphi_2(n) = \mathbf{EF}(Send_1 \wedge \ldots \wedge Send_n)$

Let us note that for the asynchronous parallel composition the witness for the formula $\varphi_1(n)$ has the length $2n + 2$, and the witness for the formula $\varphi_2(n)$ has the length $n^2 + 2n$, whereas for the synchronous parallel composition the witness for the formula $\varphi_1(n)$ has the length $2n + 2$, and the witness for the formula $\varphi_2(n)$ has the length $3n$. The experimental results for the tested formulae φ_1 and φ_2, are presented at the pictures 9 and 10 respectively.

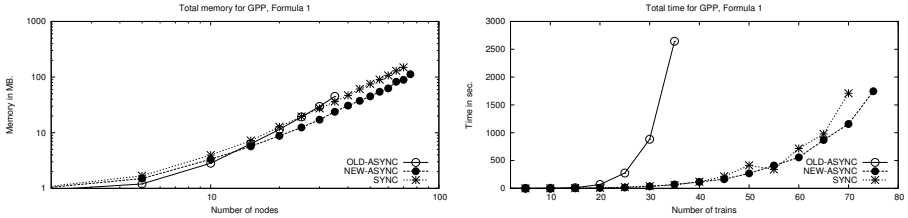

Fig. 5. Experimental results for GPP and the formula φ_1.

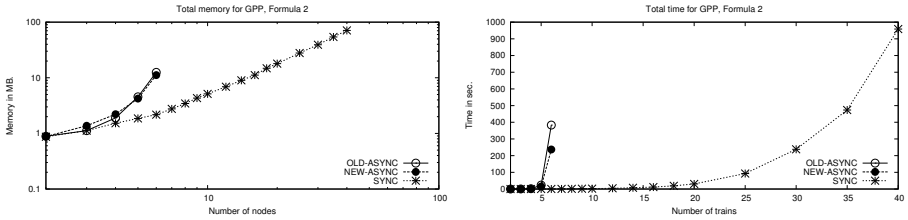

Fig. 6. Experimental results for GPP and the formula φ_2.

For the TC we have tested the following two formulae:

- $\varphi_3(n) = \mathbf{E}(\mathbf{F}(Tunnel_1) \wedge \mathbf{F}(Tunnel_n)$
- $\varphi_4(n) = \mathbf{E}(\mathbf{F}(Tunnel_1) \wedge \ldots \wedge \mathbf{F}(Tunnel_n))$

Let us note that for the asynchronous parallel composition the witness for the formula $\varphi_3(n)$ has the length 5, and the witness for the formula $\varphi_4(n)$ has the length $3(n-1)+2$, whereas for the synchronous parallel composition the witness for the formula $\varphi_4(n)$ has the length 4, and the witness for the formula $\varphi_4(n)$ has the length $2n$. The experimental results for the tested formulae φ_3 and φ_4, are presented at the pictures 7 and 8 respectively.

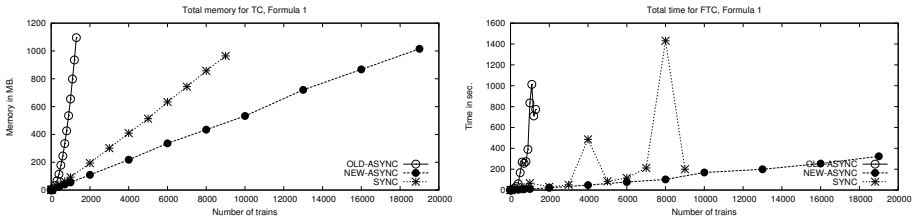

Fig. 7. Experimental results for TC and the formula φ_1.

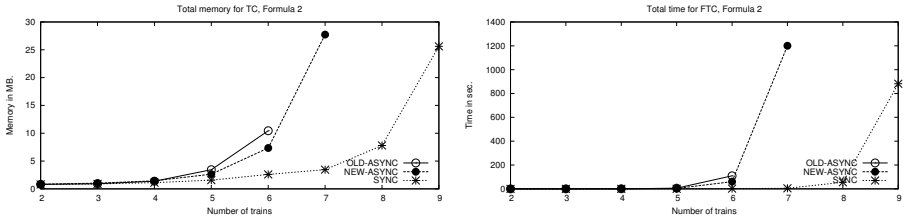

Fig. 8. Experimental results for TC and the formula φ_2.

For the DP we have tested the following two formulae:

- $\varphi_5(n) = \mathbf{E}(\mathbf{F}(Eat_0) \wedge \mathbf{F}(Eat_{n-1})$
- $\varphi_6(n) = \mathbf{E}(\mathbf{F}(Eat_0) \wedge \ldots \wedge \mathbf{F}(Eat_{n-1}))$

Let us note that for the asynchronous parallel composition the witness for the formula $\varphi_5(n)$ has the length 7, and the witness for the formula $\varphi_6(n)$ has the length $4n+1$, whereas for the synchronous parallel composition the witness for the formula $\varphi_5(n)$ has the length 6, and the witness for the formula $\varphi_6(n)$ has the length $n+3$ for $n > 6$ and 9 otherwise. The experimental results for the tested formulae φ_5 and φ_6, are presented at the pictures 7 and 8 respectively.

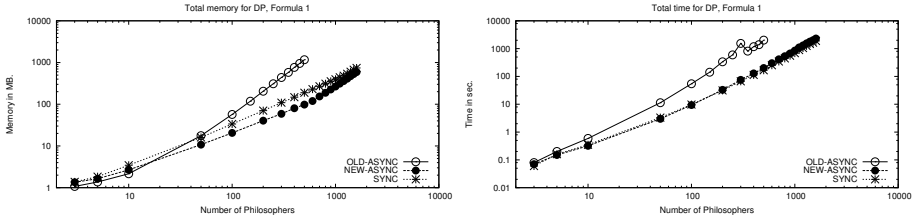

Fig. 9. Experimental results for DP and the formula φ_1.

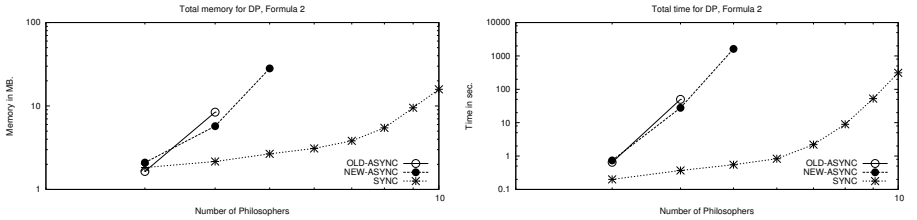

Fig. 10. Experimental results DP and for the formula φ_2.

5 Conclusions and Future Work

The experiments showed that our new Boolean encoding for asynchronous parallel composition is the most effective one while comparing to the other two considered. Moreover, the encoding of the transition relation for the synchronous parallel composition is nearly as effective as the new encoding for asynchronous parallel composition. The standard Boolean encoding used so far in the SAT-based verification modules of VerICS turned out to be the worst one. This is clearly seen for the formula $\varphi_0(n)$ that is verified on the path of the length 1. The length equal to 1 assures that the experimental results for the formula $\varphi_0(n)$ are affected by the Boolean encoding of the transition relation only.

The above conclusions, which are based on the experimental results presented in Section 4, are also supported by the fact that the length of the resulting Boolean formula for standard Boolean encoding for asynchronous parallel composition is $O(|Act| \cdot n)$, whereas the length of the resulting Boolean formula for our new Boolean encoding for asynchronous parallel composition is $O(|Act| \cdot \log |Act|)$.

Let us note that the experimental results for the formulae $\varphi_j(n)$, for $j = 1, \ldots, 6$, are affected not only by the Boolean encoding of the transition relation but also by the encoding of the translation of ECTL^* formulae to SAT. Nevertheless, the experimental results for these formulae confirm that the new encoding for asynchronous semantics significantly increases the efficiency of the SAT-based bounded model checking. It means that the effectiveness of the most of the SAT-based verification modules of VerICS could be significantly improved.

Moreover, the experimental results for the formulae $\varphi_2(n)$, $\varphi_4(n)$ and $\varphi_6(n)$ show that for formulae for which the length of the witness is significantly shorter for the

synchronous semantics, it is worth to apply the synchronous semantics instead of the asynchronous one.

We strongly believe that the choice of a SAT-solver will not change our conclusions. Nevertheless, in the nearest future we are going to repeat all the experiments for the other SAT-solvers.

References

1. Clarke, E.M.: The birth of model checking. In: 25 Years of Model Checking - History, Achievements, Perspectives. Volume 5000 of Lecture Notes in Computer Science., Springer (2008) 1–26
2. Clarke, E.M.: Model checking - my 27-year quest to overcome the state explosion problem. In: Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings. Volume 5330 of Lecture Notes in Computer Science., Springer (2008) 182
3. Clarke, E.M., Emerson, E.A., Sifakis, J.: Model checking: algorithmic verification and debugging. Communications of the ACM **52**(11) (2009) 74–84
4. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)
5. Zbrzezny, A.: Improvements in SAT-based reachability analysis for timed automata. Fundamenta Informaticae **60(1-4)** (2004) 417–434
6. Zbrzezny, A.: SAT-based reachability checking for timed automata with diagonal constraints. Fundamenta Informaticae **67(1-3)** (2005) 303–322
7. Kacprzak, M., Nabiałek, W., Niewiadomski, A., Penczek, W., Półrola, A., Szreter, M., Woźna, B., Zbrzezny, A.: VerICS 2007 - a model checker for knowledge and real-time. Fundamenta Informaticae **85(1-4)** (2008) 313–328
8. Zbrzezny, A.: Improving the translation from ECTL to SAT. Fundamenta Informaticae **85**(1-4) (2008) 513–531
9. Zbrzezny, A.: A new translation from ECTL* to SAT. Fundamenta Informaticae **120**(3-4) (2012) 377–397
10. Zbrzezny, A., Półrola, A.: SAT-based reachability checking for timed automata with discrete data. Fundamenta Informaticae **79(3-4)** (2007) 579–593
11. Męski, A., Penczek, W., Szreter, M., Woźna-Szcześniak, B., Zbrzezny, A.: BDD- versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: Algorithms and their performance. Autonomous Agents and Multi-Agent Systems (2013) to appear.
12. Biere, A.: Picosat essentials. Journal on Satisfiability, Boolean Modeling and Computation (JSAT) **4** (2008) 75–97
13. Peled, D.: All from one, one for all: On model checking using representatives. In: Proc. of the 5th Int. Conf. on Computer Aided Verification (CAV'93). Volume 697 of LNCS., Springer-Verlag (1993) 409–423
14. Hoek, W., Wooldridge, M.: Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. Studia Logica **75(1)** (2003) 125–157
15. Dijkstra, E.: Hierarchical ordering of sequential processes. Acta Inf. **1** (1971) 115–138
16. Hoare, C.: Communicating sequential processes. Prentice Hall (1985)