Partial Order and Symmetry Reductions for Discrete Time

Dragan Bošnački

Dept. of Computer Science, Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: dragan@win.tue.nl

Abstract. Symmetry and partial order reductions are among the most effective techniques for alleviating the state space explosion problem in model checking of untimed systems. In this paper we show how these two techniques can be combined for verification of discrete-time systems. In particular, we work with the discrete-time model used in DT Spin, an extension of the model checker Spin which features partial order reduction for discrete (integer) time. Based on the theoretical results presented in the paper, we combined DT Spin with SymmSpin, an extension of Spin with symmetry reduction, with promising preliminary results.

1 Introduction

Reductions based on *symmetry* and *partial order* reductions are among the most successful techniques to tackle the state space explosion problem in model checking of untimed systems. In this paper we show how these two techniques can be combined for efficient verification of timed systems.

This work was mainly instigated by practical questions that arose during the development of two extensions of the model checker Spin [18]. The first one, *DT Spin* [7], extends Spin with discrete time and features partial order reduction for discrete time. The second extension, *SymmSpin* [8], upgrades Spin with several heuristics for symmetry reduction. Because their implementations are orthogonal, DT Spin and SymmSpin can be easily combined into de facto a new tool which extends Spin with discrete time and which features partial order and symmetry reductions.

In the time model used in DT Spin, time is divided into slices indexed by natural numbers that can be seen as readings of a fictitious global digital clock that ticks at the end of each slice. The actions happening in the same slice are assigned the same clock value, so the elapsed time between actions is measured in ticks. In this model time passes only if no other action in the system is possible.

Since concurrency is modeled by interleaving, all the actions happening in one run (execution sequence) of the system are totally ordered and thus two actions happening in the same slice are not considered necessarily simultaneous. Instead, they are considered to be ordered, and their ordering inside one slice is determined by their ordering in the run. The properties that depend only on the ordering of actions (not necessarily only within one time slice) are called *qualitative*, while those depending on the elapsed time between actions are called *quantitative*. In this paper we are interested in the verification of qualitative properties, although the model contains explicitly given time constraints.

It is easy to show [7] that the semantics of systems with such a discrete-time model can be described by means of classical transition systems for untimed models. To this end the transitions systems feature a special transition ("tick") which implements the passage of time.[1] Therefore, although the discussion in the sequel is in the context of untimed systems, all results can be seamlessly extended to discrete time.[2]

Symmetry reduction exploits the inherent symmetry of the model which is present in many systems like: mutual exclusion algorithms, cache coherence, bus communication protocols, etc. After observing that the symmetry in the description of the model results in a symmetric state space, the key idea is to partition the state space into equivalence classes of (symmetric) states. Then, the state space exploration can be performed in the usually smaller quotient state space that consists only of (representatives of the) equivalence classes.

The problem of finding canonical, i.e., unique, representatives of equivalence classes is also known as the *orbit problem*. The orbit problem is equivalent to the graph isomorphism problem[10], for which no polynomial algorithm is known. As a result, often with symmetry reduction the verification time can become critical. On the other hand, finding multiple (non-canonical) representatives usually boils down to sorting algorithms [13, 8]. An obvious drawback of the multiple representatives is that they provide less state space reduction compared to the canonical representatives. However, often in practice it turns out that, with an acceptable increasing of the state space, the verification time can be improved significantly by using multiple representatives [21, 8].

From the point of view of the symmetry reduction theory the special timed ("tick") transitions can be treated as the ordinary (untimed) transitions as long as the generated state space is symmetric. The symmetry of the state space can be achieved along the lines of [21] by imposing simple syntactic constraints on the usage of the variables (including timers) in the discrete-time models.

It is always an advantage if one can combine symmetry with other reduction techniques which are orthogonal to it, i.e., which exploit different aspects of the system for reduction of the state space. One such technique, is partial order reduction [16, 25, 23].

Partial order reduction exploits the independence of the checked property from the execution order of the system actions. More specifically, two actions a, b are allowed to be permuted precisely when, if for all sequences v, w of actions:

[1] Due to lack of space we do not discuss in this paper the advantages and disadvantages of discrete versus dense time, or more general, the issue region- versus zone automata [2]. See, for instance, [6, 4, 3, 9]

[2] Actually, we conjecture that in analogous way we can derive similar results for verification of dense time systems based on region automata [2] along the lines of [6].

if $vabw$ (where juxtaposition denotes concatenation) is an accepted behavior, then $vbaw$ is an accepted behavior as well. In a sense, instead of checking all the execution sequences, the desired property is checked only on representative sequences, which results in significant savings in space and time. Thus, the corner stone of the independence relation is the confluence condition as given in Fig. 1a. The confluence requires that from each state s of the state space the permutations of two independent actions a and b will lead to the same state s'. The actual reduction of the state space is realized during the state space exploration by limiting the search from a given state s to only a subset of the actions that are executable in s. In [7] it is shown how the timed ("tick") transitions can be fitted into the independence relation.

The problem of combining symmetry based on canonical representatives with partial order reduction was solved in [14]. In fact, most of this paper can be seen as a continuation of [14] which deals also with multiple representatives. Following [14] we derive our result in the more general setting of bisimulation preserving reductions. As symmetry reduction is just a special case of a bisimulation preserving reduction, all results are valid for symmetry too.

The central idea in the reconciling of symmetry and partial order reduction is to use a new notion of independence which is weaker than the standard one. As mentioned above, in the usual definition of independence we insist on confluence, i.e., we require that the two paths obtained by permuting the independent actions a and b meet in the same state s'. Instead, in the new definition we relax the confluence condition by allowing the permutations to lead to *bisimilar states* s'_1 and s'_2, as represented in Fig. 1b.

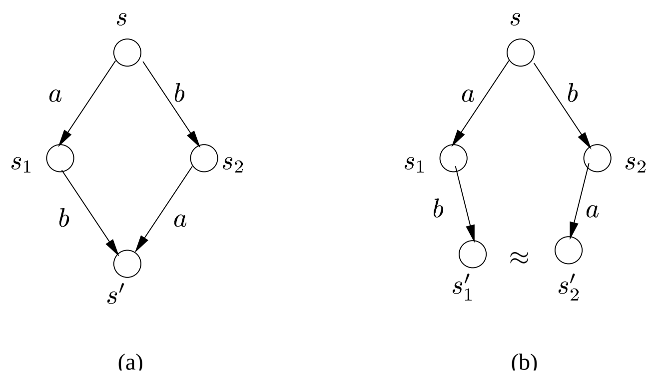

Fig. 1. Confluence of independent actions.

It turns out that almost all property preservation results, like absence of deadlock, safety, and liveness (LTL and CTL^* without the next operator), that can be found in the literature can be reused with a straightforward adaptation.

(Due to space limitation in this paper we concentrate only on next-free LTL, which is used in DT Spin and SymmSpin.)

Related work other than [14]. The usefulness of multiple representatives (without partial order reduction) is discussed in several papers (e.g. [21], [10]).

To the best of our knowledge only [17] tackles the problem of combining symmetry with multiple representatives and partial order reduction. The main difference compared to our work is that [17] works only with safety properties, while we also show preservation of liveness properties given in the next-free versions of LTL (and CTL^*). Also the multiple representatives used in [17] are of a special kind. [3] As [17] targets software model checking, it deals with symmetry based on transitions instead of states. We believe that our concept of weak independence leads to a more seamless reuse of the results that exists in the literature. This is because the latter are mostly based on symmetry of states.

Recently we learned that the concept of *symmetric independence* was introduced in [20]. It turns out that the concept is a special case of our weak independence. However, symmetric independence is used in [20] in a different context of combining partial order reduction with so-called heap symmetries for software model checking. Also, [20] assumes only canonical representatives.

We could not find in the literature any other attempt to combine partial order reduction with symmetry in the context of timed systems.

Paper layout. The next section provides some basic definitions and terminology used in the paper. Section 3.1 recalls some of the main concepts and results from [14]. The theoretical contributions of the paper are contained in Sections 3.2 and 4. In Section 3.2 we introduce the definition of weak independence and give the property preservation results that can be derived from it. In Section 4 we show that the combination of partial order and bisimulation preserving reductions preserves next-free LTL formulae which are invariant under the preserved bisimulation. Some preliminary experimental results with the combination of DT Spin and SymmSpin are reported in Section 5. The last section concludes the paper and provides some directions for future work.

2 Preliminaries

In the paper we assume temporal logics defined in the standard way. For the definitions of CTL^* and LTL, as well as their variants without the next operator CTL^*-$X$and LTL-X, we refer the reader to, for instance, [12].

We represent the state space of the system which is checked as a *labeled transition system* formally defined as follows:

Definition 1. *Let Π be a set of atomic propositions. A* labeled transition system *(LTS) is a 5-tuple $T = (S, R, L, A, \hat{s})$, where*

[3] For the sake of truth it should be said that the derivation of the results can be done for the general case.

- S *is a finite set of* states,
- $R \subseteq S \times A \times S$ *is a* transition relation *(we write $s \xrightarrow{a} s' \in R$ for $(s, a, s') \in R$),*
- $L : S \to 2^{\Pi}$ *is a* labeling function *which associates with each state a set of atomic propositions that are true in the state,*
- A *is a finite set of actions,*
- $\hat{s}$ *is the* initial state*,*

Unless stated differently, we fix T to be $(S, R, L, A, \hat{s})$ for the rest of the paper.

An action a is *enabled* in a state $s \in S$ iff $s \xrightarrow{a} s' \in R$ for some $s' \in S$. We denote the set of enabled actions in a given state s with $en_T(s)$. Similarly, for given $s \in S$ and $a \in A$ we define $a_T(s) = \{s' \mid s \xrightarrow{a} s' \in R\}$. An *execution sequence* or *path* is a finite or infinite sequence of subsequent transitions, i.e., for $s_i \in S$, $a_i \in A$, the sequence $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \ldots$ is an execution sequence in T iff $s_i \xrightarrow{a_i} s_{i+1} \in R$ for all $i \geq 0$. A finite execution sequence $c = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$,$n \geq 1$ is a *cycle* iff the start and end states coincide, i.e. $s_0 = s_n$. A state s is *reachable* iff there exists a finite execution sequence that starts at $\hat{s}$ and ends in s.

Next, we define bisimulation between two LTSs:

Definition 2. *Given two LTSs $T_1 = (S_1, R_1, L_1, A, \hat{s}_1)$ and $T_2 = (S_2, R_2, L_2, A, \hat{s}_2)$, an equivalence relation $\equiv \subseteq S_1 \times S_2$ is called a* bisimulation *between T_1 and T_2 iff the following conditions hold:*

- $\hat{s}_1 \equiv \hat{s}_2$*;*
- *If $s \equiv s'$, then:*
  - $L_1(s) = L_2(s')$*;*
  - *Given an arbitrary transition $s \xrightarrow{a} s_1 \in R_1$, there exists $s_2 \in S_2$ such that $s' \xrightarrow{a} s_2 \in R_2$ and $s_1 \equiv s_2$;*
  - *The symmetric condition holds: Given an arbitrary transition $s' \xrightarrow{a} s_2 \in R_2$, there exists $s_1 \in S_1$ such that $s \xrightarrow{a} s_1 \in R_1$ and $s_1 \equiv s_2$;*

We say that T_1 and T_2 are bisimilar *iff there exists a bisimulation between T_1 and T_2.*

3 Property Preserving Reductions

3.1 Bisimulation Preserving Reduction

In this section we recall some definitions and results from [14]. Our starting point is the idea to perform the model checking in an abstract state space, which is usually much smaller than the original one. To this end, the original state set S is partitioned into equivalence classes. The abstract state space consists of (not necessarily unique) representatives of these classes, chosen by a function h, with transitions between them as defined below.

Definition 3. *Given a function $h : S \to S$ on LTS $T = (S, R, L, A, \hat{s})$, we define the corresponding* abstract LTS *T_h to be $(S_h, R_h, L_h, A, h(\hat{s}))$, where*

- $S_h = h(S)$, *the set of representatives,*
- $r_1 \xrightarrow{a} r_2 \in R_h$ *iff there exists* $s \in S$ *such that* $r_1 \xrightarrow{a} s \in R$ *and* $h(s) = r_2$.
- *for all* $r \in S_h, L_h(r) = L(r)$.

In order to preserve the properties of interest that hold in T also in T_h we need to impose some additional constraints on the function h. In particular, we require that the equivalence class induced by the partitioning is a bisimulation.

Definition 4. *For a given LTS T, a function $h : S \to S$ is a* selection function *iff there exists a bisimulation $\equiv \subseteq S \times S$ between T and T such that for all $s \in S$, $s \equiv h(s)$.*

Intuitively, the function h picks one or more representatives for each equivalence class of S induced by $\equiv$. It should be emphasized that in the definition of selection function in [14] h is required to satisfy the additional property

$$s \equiv s' \text{ implies that } h(s) = h(s')$$

which in the sequel we call *canonicity requirement*. Obviously, with this requirement each equivalence class has a canonical (unique) representative. In what follows we assume that h is a selection function without the canonicity requirement, which implies multiple representatives per equivalence class.

We say that h *preserves* the bisimulation relation $\equiv$. The following result is implied by the definitions given above:

Lemma 1 ([14]). *Given an LTS $T = (S, R, L, A, \hat{s})$ and a selection function h, T and T_h are bisimilar.*

Lemma 1 is actually Lemma 8 from [14]. The proof of that lemma in [14] is valid in our setting too because it does not use the canonicity requirement for the representatives. (Due to lack of space the proofs are omitted in this version of the paper.) Taking into account the well known preservation properties of bisimulation, the main consequence of Lemma 1 is the following result [14]:

Theorem 1. *Given a LTS T, let ϕ be a formula in CTL^*-X over the set of atomic propositions Π of T. The formula ϕ is satisfied by T iff it is satisfied by $h(T)$. (See [12] for a definition of the satisfaction of an CTL^*-X formula by an LTS.)*

The result above implies that we can do the model checking in the reduced state space. Notice that the property ϕ is in fact not an arbitrary CTL^*-X formula. The set of preserved formulae is limited by the choice of the set of atomic propositions Π and the bisimulation relation. Thus, there is an implicit requirement that ϕ is invariant under the bisimulation $\equiv$ which is preserved by the reduction. More precisely, if $prop(\phi) \subseteq \Pi$ is the set of atomic propositions in ϕ, we have that for all $s, s' \in S$, if $s \equiv s'$, then $L(s) \cap prop(\phi) = L(s') \cap prop(\phi)$. (The last condition is obviously trivially satisfied because we have $L(s) = L(s')$ from the definition of T_h.) As LTL is contained in CTL^*, Theorem 1, with the above invariance remark, is valid for LTL too.

3.2 Partial Order Reduction

This section and the next one contain the main theoretical contributions of the paper. In the current section we define a (partial-order) reduced LTS along the lines of [1]. After the definition of the reduced LTS modulo a reduction function r, we list some conditions on this function, which ensure preservation of particular properties. Intuitively, during the construction of the reduced LTS in each new state the function r selects only a subset of the enabled transitions in that state.

Definition 5 ((Partial Order) Reduction). *For any so-called* reduction *function* $r : S \to 2^R$, *we define the (partial-order)* reduction *of* T *with respect to* r *as the smallest LTS* $T^r = (S^r, R^r, L^r, A, \hat{s})$ *satisfying the following conditions:*

- $S^r \subseteq S$, $R^r \subseteq R$, *and* $L^r = L \cap (S^r \times 2^{\Pi})$;
- *for every* $s \in S^r$, *if* $s \xrightarrow{a} s'$ *is in* $r(s)$, *then* $s \xrightarrow{a} s'$ *is in* R^r.

Note that these requirements imply that, for every $s \in S^r$ *and* $a \in A$, *if* $s \xrightarrow{a} s' \in R^r$, *then also* $s \xrightarrow{a} s' \in R$. *For every* $s \in S$ *we define* $act(r(s)) = \{a \mid \exists s'.s \xrightarrow{a} s' \in r(s)\}$.

It may be clear that not all reductions preserve all properties of interest. Thus, depending on the properties that a reduction must preserve, we have to define additional restrictions on r. To this end, we need to formally capture the notion of independence introduced earlier. Actions occurring in different processes may still influence each other, for example, when they access global variables in the specification. The following notion of independence defines the absence of such mutual influence. Intuitively, two actions are independent iff, in every state where they are both enabled, (1) the execution of one action cannot disable the other and (2) the result of executing both actions is always the same or similar state.

Definition 6 (Independence). *Given an LTS* $T = (S, R, L, A, \hat{s})$ *and a relation* $\approx \subseteq S \times S$, *an irreflexive and symmetric relation* $I \subseteq A \times A$ *is a* $\approx$-independence relation *on actions iff for each pair of actions* $(a, b) \in I$ *(called* independent actions*) it must hold that for each* $s \in S$, *if* $\{a, b\} \subseteq en_T(s)$ *then*

- *for each state state* $s' \in a_T(s)$ *we have that* $b \in en_T(s')$.
- *there exists a path* $s \xrightarrow{a} s_1 \xrightarrow{b} s'$ *in* T *iff there exists a path* $s \xrightarrow{b} s_2 \xrightarrow{a} s''$ *in* T *and* $s' \approx s''$.

In what follows we assume that $\approx$ is either the identity or a bisimulation relation. The first version of $\approx$-independence, obtained by taking $s' \approx s''$ iff $s' = s''$, corresponds to the usual definition of independence that can be found in the literature. Therefore, throughout the paper we refer to it simply as *(standard) independence*. The second version, when $\approx$ is a bisimulation relation, we call *weak independence*. We introduce the latter in order to combine bisimulation preserving reductions with partial order reduction.

In order to reuse as seamlessly as possible the existing results about partial order reduction in the sequel we assume that T is $\approx$-*deterministic*, i.e., for any $s \in S$ and $a \in A$ it holds that if $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ are in R, then $s \approx s'$. This is not a serious limitation because the symmetry reduction (or more precisely, symmetry equivalence), which is our main target, satisfies this constraint.[4]

The following conditions C0 to C3 that constrain the choice of $r(T)$ are sufficient in order to guarantee preservation of LTL forumulae.

C0 (emptiness) $r(s) = \emptyset$ iff $en_T(s) = \emptyset$.

C1 (persistency) For any $s \in S$ and a finite execution sequence $s = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$ with $a_i \notin r(s)$ for all i $(0 \leq i < n)$, action a_{n-1} is $\approx$-independent of all actions in $act(r(s))$.

C2 (cycle proviso) For any *cycle* $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n = s_0$ there is an i with $0 \leq i < n$ such that $r(s_i) = en_T(s_i)$.

For any LTL formula ϕ over the set of atomic propositions Π of a given LTS T, $prop(\phi)$ is the set of atomic propositions in ϕ.

Definition 7 (Invisibility). *An action $a \in A$ is ϕ-*invisible *in state $s \in S$ iff $a \notin en_T(s)$ or, for all $\pi \in prop(\phi)$ and $s' \in S$ such that $s \xrightarrow{a} s'$, $\pi \in L(s) \Leftrightarrow \pi \in L(s')$. Action a is* globally *ϕ-invisible iff it is ϕ-invisible for all $s \in S$.*

Informally, a globally ϕ-invisible action cannot change the truth value of the formula ϕ.

C3 (invisibility) For any state $s \in S$, all actions in $act(r(s))$ are globally ϕ-invisible or $r(s) = en_T(s)$.

Theorem 2 (Next-time-free LTL preservation). *Let r be a reduction function for LTS T satisfying C0, C1, C2, and C3; let ϕ be a next-time-free LTL formula. T satisfies ϕ iff the reduced LTS T^r satisfies ϕ.*

The proof of (variants of) Theorem 2 for standard independence can be found in [23, 19, 26, 11]. The proof of [11] can be straightforwardly adjusted for weak independence too. A similar remark to the one at the end of Section 3.1 for the preservation of CTL^*-$X$and the local properties above can also be given for LTL-X: there is an implicit requirement that ϕ is invariant under $\approx$.

In a similar way one can adjust the partial order reduction algorithm of [15] in order to preserve CTL^*.

[4] Nondeterminism and partial order reduction can be reconciled too with certain modifications of the criteria presented below [25].

4 Combining Bisimulation Preserving Reductions with Partial Order Reduction

In this section we combine the bisimulation preserving reduction with partial order reduction in two ways. In the first approach, which we introduce only for the sake of the proof and which we call *sequentially combined reduction*, we first apply the bisimulation preserving reduction in order to obtain the abstract LTS T_h. After that we apply partial order reduction on T_h. We use for $\approx$ in the definition of r the bisimulation which is preserved by h, i.e., we assume weak independence. Intuitively, the second more efficient approach corresponds to applying both partial order and bisimulation reduction simultaneously while generating (on-the-fly) the (reduced) state space. So, we refer to this approach as *simultaneously combined reduction*. We assume the same weak independence relation as in the first approach. Actually, it turns out that we can reuse the independence relation of the original LTS T, as a weak independence relation for the reduction in the reduced state space in both approaches. The correctness of the first combination follows straightforwardly from the preservation results presented in the previous section. Thus, we establish the correctness of the second approach by showing that both approaches generate the same reduced state space, provided that the same selection and reduction functions are used in both cases.

Definition 8. *Given a LTS T and a selection function h we define the* sequentially combined reduction $(T_h)^r$ *to be the LTS obtained such that partial order reduction, as defined in Def. 5, is applied on the abstract LTS T_h.*

In order to preserve properties we substitute in r for $\approx$ the bisimulation $\equiv$ which is preserved by h. The preservation of the LTL-Xproperties which hold in the original LTS T is implied by the preservation results from the previous section. In particular, the preservation from T to T_h is guaranteed by Theorem 1. The preservation in the second phase, between T_h and $(T_h)^r$, follows directly from Theorem 2. Hence, we can rephrase the theorem(s) from the previous section for the sequentially combined reduction $(T_h)^r$.

Theorem 3 (Next-time-free LTL preservation). *Given an LTS T with a selection function h which preserves the bisimulation $\equiv$, let r be a reduction function which satisfies C0, C1, C2, and C3, and which is based on $\equiv$-independence. Let ϕ be a LTL-Xformula. T satisfies ϕ iff the reduced LTS $(T_h)^r$ satisfies ϕ.*

A generic algorithm for partial order algorithm is given in fig Fig. 2. Applied on T_h it constructs the sequentially combined reduction $(T_h)^r$. (It is trivial to write a similar algorithm which generates T_h from T according to Def. 3.)

The following definition paves the way to a more efficient algorithm for performing combined reduction:

Definition 9 (Simultaneously Combined Reduction). *Let T be a LTS for which we are given a selection function h, which preserves the bisimulation $\equiv$,*

```
states := unexpanded := {ŝ}; transitions = ∅
while unexpanded ≠ ∅ do
    remove a state s from unexpanded;
    for each transition s -a-> s' in r(s) do
        if s' ∉ states then
            add s' to states and unexpanded;
            add s -a-> s' to transitions
```

Fig. 2. An algorithm for Sequentially Combined Reduction

and a reduction function r. We define the simultaneously combined reduction *for T with respect to r and h as the smallest LTS $T_h^r = (S_h^r, R_h^r, L_h^r, A, h(\hat{s}))$ satisfying the following conditions:*

- *$S_h^r \subseteq S$, $R_h^r \subseteq R$, and $L_h^r = L \cap (S_h^r \times 2^{\Pi})$;*
- *for every $s \in S_h^r$, if $s \xrightarrow{a} s'$ is in $r(s)$, then $s \xrightarrow{a} h(s')$ is in R_h^r.*

The algorithm in Fig. 3 is an implementation of the above definition.

```
states := unexpanded := {h(ŝ)}; transitions = ∅
while unexpanded ≠ ∅ do
    remove a state s from unexpanded;
    for each transition s -a-> s' in r(s) do
        if h(s') ∉ states then
            add h(s') to states and unexpanded;
            add s -a-> h(s') to transitions
```

Fig. 3. An algorithm for Simultaneously Combined Reduction

The following theorem states the equivalence of the two reduction combinations, if they are featuring the same selection and reduction functions:

Theorem 4. *For a LTS T with a selection function h which preserves the bisimulation $\equiv$ it holds that $(T_h)^r = T_h^r$, provided that the functions r in both reductions are defined with $\equiv$-independence and satisfy the same subset of the conditions C0 to C3.*

The theorem can be proved in a similar way as Theorem 19 in [14]. After observing that with the conditions of the theorem the functions r are the same, we show that for each run of the algorithm in Fig. 2 there exists a corresponding run of the algorithm in Fig. 3. To this end we execute lock-step both runs and show that the invariant holds that the variables *states*, *transitions*, and *unexpanded* are the same in both executions.

The following observation is important from practical point of view because it solves in an elegant way the requirement that the same weak independence has to be used in both approaches:

Lemma 2. *Let I be an independence relation for a LTS $T = (S, R, L, A, \hat{s})$ and h a bisimulation preserving selection function for T which preserves the bisimulation $\equiv$. In this case, I is a weak (i.e., $\equiv$) independence relation for the corresponding abstract LTS T_h.*

Again, this can be proved like the analogous Lemma 10 from [14] which holds for unique representatives.

In conclusion, Theorem 4 leads us directly to an algorithm for checking properties in a reduced LTS which is obtained by combining partial order and bisimulation reduction (thus, also symmetry) with multiple representatives. Moreover, this can be done on-the-fly and by Lemma 2 reusing the independence relation of the original LTS T. Such an algorithm can be obtained by adjusting in a straightforward way the algorithm in Fig. 3 to various well known algorithms for verification of safety and liveness properties.

As it was mentioned in the introduction, although the above results were derived in the context of untimed systems, they can be directly extended to the discrete-time model used in DT Spin.

5 Experimental Results

In this section we report some preliminary experimental results evaluating the combination of techniques presented in the previous sections. Experiments were performed with a combination of DT Spin and SymmSpin, extensions of the model checker Spin with discrete time (featuring partial order reduction) and symmetry, respectively.

The obtained reductions were of several orders of magnitude in the number of states. Also, due to the significantly smaller number of states that had to be explored, in all the cases the verification with symmetry reduction was faster than the one without it. In the verification experiments we used the combination of DT Spin and SymmSpin with and without the partial order reduction (POR) option. There was an obvious synergy between the symmetry and the partial order reductions. The two reduction techniques are orthogonal because they exploit different features of the concurrent systems, therefore, their cumulative effect can be used to obtain more efficient verification. Also the profit in verification time from the non-canonical (multiple) representatives was clearly observed.

Table 1 contains some results for the well known Fischer's mutual exclusion protocol [24]. All experiments were performed on a 1 GHz Pentium III machine with 256 MB running Red Hat Linux 7.0. Verification times (in the rows labeled with "t") are given in seconds ($s.x$); the number of states (in the rows labeled with "s") is given directly or in millions (say, 9.1M); *o.m.* stands for out of memory, and *o.t.* denotes out of time (more than 10000 seconds); +POR and -POR mean with and without POR, respectively. The abbreviations "can." and "mult." refer to the heuristics for symmetry reduction "pc-segmented" and "pc-sorted" from [8] which use canonical and multiple (non-canonical) representatives, respectively. Similarly, "no sym." means verification without symmetry reduction. N denotes the number of competing processes.

Table 1. Results for Fischer's mutual exclusion protocol.

N		5		6		7		8		9		10		11	
		+POR	-POR	+POR	-POR	+POR	-POR	+POR	-POR	+POR	-POR	+POR	-POR	+POR	-POR
no	s	503898	2.52M	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.	o.m.
sym.	t	3.2	26.9	—	—	—	—	—	—	—	—	—	—	—	—
	s	8949	32630	23414	115066	55065	362827	120541	1.04M	245467	2.78M	476558	o.m.	o.t.	o.m.
can.	t	0.2	0.8	0.8	4.6	3.7	29.0	25.7	202.5	213.1	1620.7	1980.6	—	—	—
	s	11394	45052	32781	189011	85426	726729	208970	2.6M	482569	o.m.	1.08M	o.m.	2.3M	o.m.
mult.	t	0.1	0.7	0.5	3.8	1.1	18.8	3.2	87.6	8.9	—	23.5	—	59.7	—

The results given in Table 1 are for the values of the two time constants in the protocol as in [24], i.e., 1 and 2. (Notice that for these values the Fischer protocol is correct.) If the values of the time constants are increased, also the state space increases. However, the reduction percentage due to POR and symmetry (combined as well as separately) stays virtually the same as for the basic values (1 and 2). With erroneous versions of Fischer's protocol (obtained by swapping the values of the time constants) reductions due to symmetry (and/or POR) are still observed. They are much more modest though – they are mostly around 50%, for the values of $N \leq 11$.

6 Conclusion and Future Work

In this paper we showed that partial order and symmetry reductions can be used together for verification of discrete time systems. The central idea behind the compatibility was to introduce a new kind of independence which is weaker than the independence usually used in the literature. In particular, we weakened the confluence condition by requiring that the permutations of independent actions lead to bisimilar states. The theoretical results were used as a background for extending the model checker Spin with a combination of discrete time with partial order and symmetry reductions. Preliminary experimental results show that significant improvements in both memory and time can be achieved by employing together these two techniques.

An obvious task for the future is to try to confirm in practice the results of the paper on more case studies. Another straightforward direction might be to combine symmetry with partial order techniques for dense time based on zone automata [2], (e.g. [5]). In that context it should be trivial to adjust the symmetry heuristics from [8] for difference bound matrices [2] and related data structures used in other verification tools (e.g. [22, 24]).

The implementations of DT Spin and SymmSpin are described in [7] and [8], respectively. DT Spin is available from

`http://www.win.tue.nl/~dragan/DTSpin,`

while SymmSpin can be obtained from

`http://www.win.tue.nl/~lhol/SymmSpin.`

Acknowledgments. The author would like to thank Mark Scheffer for his comments on an early draft of this paper and for the contribution to the proof of Theorem 2.

References

1. R. Alur, R.K. Brayton, T.A. Henzinger, S. Qadeer, S.K. Rajamani, Partial-Order Reduction in Symbolic State-Space Exploration, Computer Aided Verification, CAV '97, LNCS 1254, pp. 340–351, Springer, 1997.
2. R. Alur, D.L. Dill, *A Theory of Timed Automata*, Theoretical Computer Science, 126, pp.183-235, 1994.
3. R. Alur, R. P. Kurshan, *Timing Analysis in Cospan*, Hybrid Systems III, LNCS 1066, pp.220-231, Springer, 1996.
4. E. Asarin, M. Bozga, A. Kerbat, O. Maler, A. Pnueli, A. Rasse, *Data Structures for the Verification of Timed Automata*, In Proc. of the Intl. Workshop on Hybrid and Real-Time Systems, 1997
5. J. Bengtsson, B. Jonsson, J. Lilius, W. Yi, *Partial Order Reductions for Timed Systems*, CONCUR'98, LNCS 1466, pp. 485-501, 1998.
6. D. Bošnački, *Partial Order Reduction for Region Automata* (abstract), Proceedings of the 11th Nordic Workshop on Programming Theory NWPT'99, Uppsala, Sweden, October 1999
7. D. Bošnački, D. Dams, *Integrating Real Time in Spin: a Prototype Implementation*, FORTE/PSTV'98, Kluwer, pp. 423-439, 1998.
8. D. Bošnački, D. Dams, L. Holenderski, *Symmetric Spin*, 7th Int. SPIN Workshop on Model Checking of Software SPIN 2000, pp. 1-19, LNCS 1885, Springer, 2000.
9. M. Bozga, O. Maler, A. Pnueli, S. Yovine, *Some Progress in the Symbolic Verification of Timed Automata*, CAV'97,1997.
10. E.M. Clarke, R. Enders, T. Filkorn, S. Jha, Exploiting Symmetry in Temporal Logic Model Checking, *Formal Methods in System Design*, Vol. 19, 77–104, 1996.
11. E.M. Clarke, Jr., O. Grumberg, D.A. Peled, *Model Checking*, The MIT Press, 2000.
12. E.A. Emerson, *Temporal and Modal Logic*, in J. van Leeuwen (ed.), Formal Models and Semantics, pp. 995–1072, Elsevier, 1990.
13. E.A. Emerson, A.P. Sistla, *Symmetry and model checking*, Proc. of CAV'93 (Computer Aided Verification), LNCS 697, pp. 463–478, Springer, 1993.
14. E.A. Emerson, S. Jha, D. Peled, *Combining partial order and symmetry reductions*, in Ed Brinksma (ed.), Proc. of TACAS'97 (Tools and Algorithms for the Construction and Analysis of Systems), LNCS 1217, pp. 19–34, Springer, 1997.

15. R. Gerth, R. Kuiper, D. Peled, W. Penczek, *A Partial Order Reduction Approach to Branching Time Logic Model Checking*, Proc. of Third Israel Symposium on Theory on Computing and Systems, pp. 130-139, Tel Aviv, Israel, IEEE, 1995.
16. P. Godefroid, *Partial Order Methods for the Verification of Concurrent Systems: An Approach to the State Space Explosion*, LNCS 1032, Springer, 1996.
17. P. Godefroid, Exploiting Symmetry when Model-checking Software, *Proc. of FORTE/PSTV'99 (Formal Methods for Protocol Engineering and Distributed Systems)*, 257–275, Beijing, Oct. 1999.
18. G. J. Holzmann, *Design and Validation of Communication Protocols*, Prentice Hall, 1991. Also: `http://netlib.bell-labs.com/netlib/spin/whatispin.html`.
19. G. Holzmann, D. Peled, *An Improvement in Formal Verification*, FORTE 1994, Bern, Switzerland, 1994.
20. R. Iosif, *Symmetry Reduction Criteria for Software Model Checking*, to appear in the Proc. of SPIN 2002, LNCS, Springer, 2002.
21. C.N. Ip, D.L. Dill, Better verification through symmetry. *Formal Methods in System Design*, Vol. 9, pp. 41–75, 1996.
22. K.G. Larsen, P. Pettersson, W. Yi, *UPPAAL: Status & Developments*, Computer Aided Verification CAV 97, LNCS 1254, pp.456-459, Springer-Verlag, 1997.
23. D. Peled, *Combining Partial Order Reductions with On-the-Fly Model Checking*, Computer Aided Verification 1994, LCNS 818, pp. 377–390, 1994.
24. S. Tripakis, C. Courcoubetis, *Extending Promela and Spin for Real Time*, TACAS '96, LCNS 1055, Springer Verlag, 1996.
25. A. Valmari, The State Explosion Problem. Lectures on Petri Nets I: Basic Models, LNCS 1491, pp. 429-528, Springer, 1998.
26. A. Valmari, *A Stubborn Attack on State Explosion*, Formal Methods in System Design, Vol. 1, pp. 297–322", 1992.