Symmetry Reduction for SAT Representations of Transition Systems

Jussi Rintanen
Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, 79110 Freiburg im Breisgau
Germany

Abstract

Symmetries are inherent in systems that consist of several interchangeable objects or components. When reasoning about such systems, big computational savings can be obtained if the presence of symmetries is recognized. In earlier work, symmetries in constraint satisfaction problems have been handled by introducing symmetry-breaking constraints. In reasoning about transition systems, notably in model-checking and reachability analysis in computer-aided verification, symmetries have been handled by symmetry reduction algorithms that eliminate redundant search caused by symmetries.

In this work, we investigate symmetry handling in a problem in the intersection of these two areas: handling symmetries in representations of transition systems in the propositional logic. The problem shows up in representations of AI planning as a satisfiability problem, and in recent approaches to model-checking that represent transition systems as propositional formulae. Symmetry-breaking constraints can be added to the propositional logic representation of transition sequences for removing all the symmetry at one point of time, but removing symmetry from the whole transition sequence is much more difficult, and has not been addressed in earlier work. We present a solution to the problem.

Introduction

Symmetry breaking has been extensively investigated in connection with problems represented in the propositional logic (Crawford *et al.* 1996; Joslin & Roy 1997). Symmetries divide the set of all solutions to equivalence classes: two symmetric solutions belong to the same equivalence class. The standard way of utilizing symmetries is by so-called symmetry-breaking constraints that eliminate from each equivalence class of symmetric solutions all but one. On some types of problems these constraints yield exponential speed-ups.

Earlier, symmetry reductions in explicit-state reachability analysis and in different forms of model-checking have been investigated. In explicit-state model-checking and reachability analysis, for example in Petri net reachability analysis, during the traversal of the state space restriction to only one of a class of symmetric successor states is made (Starke 1991). Similarly, model-checking algorithms for temporal logics have been modified to work on the quotient structure resulting from identification of states that are equivalent modulo symmetries (Emerson & Sistla 1996). These techniques solve the symmetry reduction problem for planning algorithms that use plain backward or forward search in the state space.

Satisfiability testing in the propositional logic has been used as a computational framework for solving a number of problems about transition systems. In planning by satisfiability (Kautz & Selman 1996) and its extensions (Rintanen 1999; Majercik & Littman 1999), sequences of actions/transitions are represented in the classical propositional logic or its quantified extensions. Similarly, bounded model-checking (Biere *et al.* 1999) represents transition sequences in the classical propositional logic.

None of the existing techniques for eliminating symmetry is directly applicable to representations of transition systems in the propositional logic. The symmetry reduction algorithms developed for model-checking take as input the current state, and compute one successor state from each class of successor states equivalent modulo symmetry. In principle, this kind of algorithms could be embedded in a satisfiability algorithm, but this would complicate the structure of such algorithms tremendously.

Also, symmetry-breaking constraints as used in satisfiability testing and constraint satisfaction do not directly generalize to symmetries in sequences of states. Conventional symmetry-breaking constraints can easily be applied to one time point at a time: just order the states for example lexicographically using standard techniques. However, separate lexicographic ordering of states in consecutive time points does not preserve the solvability of the problem instance. This is because the successors of the lexicographically first states are not necessarily lexicographically first: the transition relation does not preserve the lexicographic ordering imposed on the states by the symmetry-breaking constraints.

This paper addresses the following problem. Given a representation of finding transition sequences satisfying a certain property, expressed in the propositional logic, produce a new representation that allows only one sequence from each class of symmetrical transition sequences. This problem has not been solved in earlier work. Our framework is very general and can represent many different types of properties of

transition sequences needed in different applications, like reachability of a state satisfying a given propositional formula (AI planning (Kautz & Selman 1996)) and satisfaction of given formula in the linear temporal logic LTL (bounded model-checking (Biere *et al.* 1999)), and allows eliminating symmetries from the search space in all these cases.

We propose a new approach for eliminating symmetries in representations of transition sequences in the propositional logic. The idea is to not order the symmetric states at given time points, but to order the sequences of transitions. Transition sequences can be ordered lexicographically given an ordering on the transition relations.

In general, not all symmetry can be eliminated by ordering transition sequences, as transition relations that are not symmetric in some states still may produce successor states that are symmetric. However, in very many cases removing symmetries from transition sequences also removes all symmetry from the sequences of states these sequences generate.

The structure of the paper is as follows. First we exactly explain what is the problem we try to solve and why it cannot be solved by extensions of earlier approaches to symmetry elimination. Then we discuss how transition sequences can be ordered and what are the conditions under which two transition relations can be considered interchangeable in a given state, and how symmetry-breaking constraints are derived. The effect of adding the symmetry-breaking constraints on sizes of formulae is experimented with by using a simple and highly symmetric benchmark from AI planning. Finally, we will compare our results to earlier work on symmetries, and conclude the paper by discussing open problems raised by the work.

Symmetries in Transition Systems

Our transition systems consist of a set of states and a set of transition relations. Deterministic planning as well as transition systems in model-checking are compatible with this definition.

Definition 1 *A transition system $\langle S, I, T\rangle$ consists of a set S of states, a set T of transition relations $t \subseteq S \times S$,[1] and a set $I \subseteq S$ of initial states.*

We now define symmetries. Our definitions are similar or equivalent to those used earlier work on symmetries (Starke 1991; Emerson & Sistla 1996). A main difference to much of the earlier work is that the transition between two states has to be mapped to a symmetric transition. This labeling of the arcs in the transition graphs is not needed for the definition of symmetries in AI planning nor in many types of problems in computer-aided verification when symmetries are handled at the level of states, but in our case it is important for defining symmetries for transition sequences.

Definition 2 *For a transition system $P = \langle S, I, T\rangle$ and a property $E : S \to X$ (for some set X),* a symmetry group G of P *is a set of pairs $\langle \sigma, \tau\rangle$ such that*

[1]For classical (deterministic) planning the transition relations are partial functions $t : S \to S$.

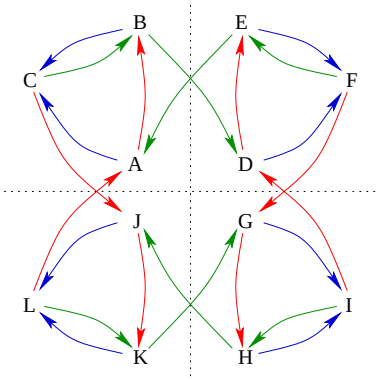

Figure 1: A symmetric transition system

- *$\sigma : S \to S$ is a permutation on S,*
- *$\tau : T \to T$ is a permutation on T,*
- *for all $\{s_1, s_2\} \subseteq S$ and $t \in T$, $(\sigma(s_1), \sigma(s_2)) \in \tau(t)$ if and only if $(s_1, s_2) \in t$,*
- *and $E(s) = E(\sigma(s))$ for all $s \in S$.*

Further, for every $\langle \sigma, \tau\rangle \in G$ there is $\langle \sigma^{-1}, \tau^{-1}\rangle \in G$ where f^{-1} is the inverse of the bijection f, and the pair $\langle 1_S, 1_T\rangle$ of identity functions $1_S(s) = s, s \in S$ and $1_T(t) = t, t \in T$ for states and transition relations also belong to G.

This is called a symmetry group because the set G fulfills the axioms of groups under composition of pairs of functions $\langle f, g\rangle \circ \langle f', g'\rangle = \langle f \circ f', g \circ g'\rangle$.

The function E formalizes the properties of the states that determine whether a given symmetry can be considered admissible. For example, when the objective of planning is to reach one of a set $S' \subseteq S$ of goal states, E is defined as $E(s) = 1$ for $s \in S'$ and $E(s) = 0$ for $s \in S \backslash S'$. This means that the relevant properties of the states, whether they are goal states or not, is invariant under the automorphism in question. For model-checking problems E could be used for partitioning the state space to sets of states that assign the same truth-value to all the state variables occurring in the formula to be model-checked.

A symmetry group G induces an equivalence relation $\equiv$ on S as $s \equiv s'$ if and only if there is $\langle \sigma, \tau\rangle \in G$ such that $s = \sigma(s')$. The relation partitions S into equivalence classes $[s] = \{s' \in S | s \equiv s'\}$.

Define $[X] = \{[s] | s \in X\}$ for sets X.

The equivalence relation induces a quotient transition system $\langle [S], [I], \{t_G | t \in T\}, E_G\rangle$ where $t_G = \{\langle [s], [s']\rangle | \langle s, s'\rangle \in t\}$ for all $t \in T$, and $E_G([s]) = E(s)$ for all $s \in S$.

Example 3 Figure 1 shows a transition system with symmetries. Let the function $E(x) = 1$ for $x \in \{B, E, H, K\}$ and $E(x) = 0$ otherwise. This graph has the following non-

trivial automorphisms (isomorphisms to itself.)

$$\begin{aligned}
\alpha_y &= \{\langle A,D\rangle,\langle B,E\rangle,\langle C,F\rangle,\langle D,A\rangle,\langle E,B\rangle,\langle F,C\rangle,\\
&\quad \langle G,J\rangle,\langle H,K\rangle,\langle I,L\rangle,\langle J,G\rangle,\langle K,H\rangle,\langle L,I\rangle\}\\
\alpha_x &= \{\langle A,J\rangle,\langle B,K\rangle,\langle C,L\rangle,\langle D,G\rangle,\langle E,H\rangle,\langle F,I\rangle,\\
&\quad \langle G,D\rangle,\langle H,E\rangle,\langle I,F\rangle,\langle J,A\rangle,\langle K,B\rangle,\langle L,C\rangle\}\\
\alpha_d &= \{\langle A,G\rangle,\langle B,H\rangle,\langle C,I\rangle,\langle D,J\rangle,\langle E,K\rangle,\langle F,L\rangle,\\
&\quad \langle G,A\rangle,\langle H,B\rangle,\langle I,C\rangle,\langle J,D\rangle,\langle K,E\rangle,\langle L,F\rangle\}
\end{aligned}$$

The automorphism α_y is mirroring horizontally, α_x is mirroring vertically, and α_d is mirroring diagonally. (In this example we do not present the mappings on edges separately.)

Hence there are automorphisms that map the state A to states D, G and J, the state B to E, H and K, and the state C to F, I and L, and all these automorphisms are invariant under the function E. Therefore the following equivalence classes of symmetric states are induced by the automorphisms.

$$\begin{aligned}
[A] &= \{A,D,G,J\}\\
[B] &= \{B,E,H,K\}\\
[C] &= \{C,F,I,L\}
\end{aligned}$$

The three automorphisms together with the identity permutation $\alpha_{id} : S \to S$ form a symmetry group of the transition system. Other symmetry groups are $\{\alpha_{id}\}$, $\{\alpha_{id},\alpha_x\}$, $\{\alpha_{id},\alpha_y\}$, and $\{\alpha_{id},\alpha_d\}$. ■

The standard approach for breaking symmetries is to consider the quotient transition system and represent each equivalence class $[s]$ by one of its elements, the *canonical state* s. When performing a transition from $[s]$ to $[s']$ where s' is the successor of s, the successor equivalence class $[s']$ is again represented by the canonical state s'' of $[s'] = [s'']$. All of a set of mutually symmetric states are mapped to the same canonical state. As the number of canonical states (and equivalence classes) is on highly symmetric problems a very small fraction of the number of all states, big speed-ups are obtained.

However, computing canonical states is non-trivial for big transition systems succinctly represented for example in terms of state variables. The successor s' of a canonical state s with respect to a transition is in general not a canonical state of $[s']$. Non-trivial computation is needed for computing the canonical states.

The problem is more pronounced in representations of transition systems in the propositional logic. For the first time point symmetry-breaking constraints can be used for choosing the first element s among the members of $[s]$, but the same constraints on the successor time point are usually inconsistent with the description of the transition from $[s]$ to $[s']$, because successors of the canonical elements are often not canonical elements of their equivalence classes.

Expressing the transition relation in a way that produces canonical elements from canonical elements would solve the symmetry problem. However, encoding transition relations in this way is very complicated as it would essentially mean that the encoding of the problem in the propositional logic would include a complete symmetry reduction algorithm, the expression of which even in a conventional programming language is often a complicated task. Consequently, this approach would result in impractically big formulae.

Symmetry Reduction with Transition Sequences

Our idea is to not consider equivalence classes of states directly, but instead equivalence classes of transition sequences. Expressing a lexicographic ordering on the transition sequences is much easier than representing the computation of canonical elements of equivalence classes of states, and yields a practically feasible approach for handling symmetries in representation of model-checking, planning and other computational problems in the propositional logic.

The basic insight is that for symmetry reduction that does not involve mapping states to the canonical states of their equivalence classes, it is sufficient to find automorphisms that map the current state to itself and under which two transitions enabled in the current state are interchangeable. If such an automorphism exists, we can safely ignore one of the transitions. For representing this in the propositional logic we needs to do the following.

1. Impose some ordering $<$ on the transition relations.
2. As a preprocessing step identify automorphisms that allow interchanging given two transition relations t and t'.
3. Synthesize a formula that tests for an automorphism that interchanges t and t', whether a given state is mapped to itself. If it is, the automorphism allows interchanging t and t' in that state.
4. Synthesize a formula that prevents using one of t and t' whenever t and t' may be interchanged in the current state. Choose t if $t < t'$ and otherwise choose t'.

In this section we formalize the first part of this idea, the conditions under which two transition relations can be viewed interchangeable, and show that it leads to symmetry reduction that preserves the existence of solutions to the model-checking or planning problem. Because finding arbitrary automorphisms is computationally very expensive, in the next section we discuss a practically relevant and computationally more tractable class of symmetries for which the derivation of symmetry-breaking constraints is much easier.

The formal definition of interchangeability is as follows.

Definition 4 *Let $s \in S$ be a state. Transition relations $t \in T$ and $t' \in T$ are* interchangeable in s *if $\langle s,s'\rangle \in t$ for some $s' \in S$ and there is $\langle \sigma,\tau\rangle \in G$ such that*

1. *$\sigma(s) = s$, and*
2. *$\tau(t) = t'$.*

The interchangeability condition allows ignoring one of two interchangeable transition relations: if there is a transition sequence starting with t and leading to a goal state or satisfying some formula to be model-checked, then there is a symmetric transition sequence starting with t' and having the same properties with respect to E, and it suffices to consider only one of t and t'.

Next we formally show that ignoring one of two interchangeable transitions never eliminates all the transition sequences having a property we might be looking for.

An ordering $<$ on transition relations induces an ordering on transition sequences for example lexicographically. Two

transition sequences $p = t_1 t_2 \cdots t_n$ and $p' = t'_1 t'_2 \cdots t'_n$ are ordered as $p < p'$ if $t_i < t'_i$ for some $i \in \{1, \ldots, n\}$ and $t_j = t'_j$ for all $j \in \{1, \ldots, i-1\}$.

Theorem 5 *Let $t_1, \ldots, t_n$ be a transition sequence starting in state s_0 and visiting the sequence of states $s_0, s_1, \ldots, s_n$. Let t'_i be interchangeable with t_i in state s_{i-1}. Then there is a sequence of transitions $t'_{i+1}, \ldots, t'_n$ that visits states $s'_i, \ldots, s'_n$ such that $E(s_j) = E(s'_j)$ for every $j \in \{i, \ldots, n\}$.*

Proof: Because t_i and t'_i are interchangeable, there is $\langle \sigma, \tau \rangle \in G$ such that $\sigma(s_{i-1}) = s_{i-1}$ and $\tau(t_i) = t'_i$. Define $s'_i, \ldots, s'_n$ by $s'_j = \sigma(s_j)$ and $t'_i, \ldots, t'_n$ by $t'_j = \tau(t_j)$ for all $j \in \{i, \ldots, n\}$. By definition of symmetry groups $E(s'_j) = E(\sigma(s_j)) = E(s_j)$ for all $j \in \{i, \ldots, n\}$. □

This result means that at any point of a transition sequence the next transition t can be replaced by t' if t and t' are interchangeable in the current state and $t' < t$. If at every time point the $<$-minimal transition is chosen among those that are interchangeable, from every equivalence class of symmetric transition sequences a lexicographically $<$-minimal transition sequence is obtained. This is the idea that we will implement in the following sections for encodings of transition systems in the propositional logic.

Symmetries from Schemata

Recognizing all symmetries in a transition system represented succinctly in terms of plan operators or some other high-level language is computationally very expensive. This is because sizes of the transition systems may be exponential in the size of their high-level descriptions.

However, there are important classes of symmetries that are easily recognized by syntactic inspection of high-level descriptions of transition systems. Indeed, an important reason for using high-level descriptions is that with them describing symmetric systems, with repetitive structure and multiple copies of some components, is more manageable. This has been recognized in earlier works on symmetries.

To avoid expensive computation needed for recognizing arbitrary symmetries, the use of special data types that induce symmetric transition systems has been proposed by Ip and Dill (1996). Interchangeability of objects has been pointed as a main source/consequence of symmetries in constraint satisfaction problems (Freuder 1991), and this has been taken advantage of for example by Joslin and Roy (1997) in computing symmetries for different types of constraint satisfaction problems, including planning, from schematic descriptions.

In this paper we use high-level descriptions of transition systems that are based on state variables, as widely used in computer-aided verification and AI planning. The set of possible transitions expressed in terms of operators acting on state variables is typically not presented by enumerating them. Instead, a schematic description is instantiated by some set of objects. This kind of schematic descriptions often directly lead to symmetric transition systems. The symmetries are induced by the interchangeability of the objects.

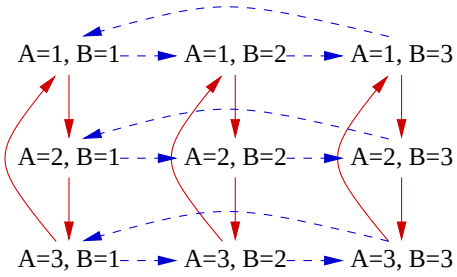

Figure 2: A transition system that is symmetric with respect to interchanging A and B.

Example 6 Consider the transition system in Figure 2. It is described by the schematic description $F(x), x \in X$ that is instantiated by members of the set $X = \{A, B\}$:

CASE x OF
1 ⇒ x := 2;
2 ⇒ x := 3;
3 ⇒ x := 1;

The transition relation $F(B)$ is depicted by dotted arrows, and the remaining arrows depict $F(A)$.

The transition system has two automorphisms (we assume that the function E for example assigns only the state $A = 3, B = 3$ a non-zero value): the pair of identity functions for the states and the transition relations, and another one consisting of the following permutation of states (we depict a state that assigns x and y respectively to A and B as xy)

$$\{\langle 11, 11\rangle, \langle 12, 21\rangle, \langle 13, 31\rangle, \langle 21, 12\rangle, \langle 22, 22\rangle, \langle 23, 32\rangle, \langle 31, 31\rangle, \langle 32, 23\rangle, \langle 33, 33\rangle\}$$

and the permutation $\{\langle F(A), F(B)\rangle, \langle F(B), F(A)\rangle\}$ of transition relations.

These two automorphisms together with the identity automorphism form a symmetry group of the transition system. The equivalence classes of symmetric states induced by the automorphisms are the following.

$$\begin{array}{lcl} [11] & = & \{11\} \\ [12] & = & \{12, 21\} \\ [13] & = & \{13, 31\} \\ [22] & = & \{22\} \\ [23] & = & \{23, 32\} \\ [33] & = & \{33\} \end{array}$$

In this example, the number of automorphisms is $n!$ when the cardinality of the set X is n, and the number of equivalence classes is $\sum_{i \geq 1}^{n+1} i$, which is $\mathcal{O}(n^2)$. Because the number of equivalence classes grows slowly with respect to the growth rate of the cardinalities of the state space (3^n), some of the equivalence classes have a very high cardinality, and the transition systems are highly symmetric. ■

The most basic automorphisms for problems represented schematically are those that correspond to interchanging some two constants $c_1, c_2 \in X$. All other automorphisms implicit in the schematic description can be obtained by composition from these.

For c_1 and c_2 to be interchangeable their only occurrence in the schematic description must be as a member of a set of objects used for instantiating the schemata. If one of these constants occurs in the schemata explicitly (without a matching occurrence of the other), the constants are not in general interchangeable. In Example 6 the variables A and B satisfy this property.

Representation of Transition Sequences in the Propositional Logic

In this section we briefly describe how planning and bounded model-checking can be represented in the propositional logic. The possible transitions of a transition system are represented by formulae $\rho(P, P')$ on the set $P \cup P'$ of propositions, with P describing the truth-values of the state variables in the predecessor state and P' describing the truth-values of the state variables in the successor state.

A deterministic transition relation t can be translated into a formula $\phi_t = c_t \wedge e_t$ where c_t is a formula over P (describing whether the transition can take place) and e_t is a conjunction of equivalences $\phi_p \leftrightarrow p'$, one for every $p \in P$, where ϕ_p is a formula over P that describes the condition under which p is true in the successor state. The truth-value of every state variable in the successor state is hence uniquely determined by the predecessor state. We also introduce a proposition t for every transition relation. Now the representation of one time step is

$$\rho(P, P') = (t_1 \vee \cdots \vee t_n) \wedge (t_1 \rightarrow \phi_{t_1}) \wedge \cdots \wedge (t_n \rightarrow \phi_{t_n})$$

with one proposition t for each transition.

A sequence of n transitions from a set of states described by ι and ending in a state satisfying Γ is represented by

$$\iota^0 \wedge \rho(P^0, P^1) \wedge \rho(P^1, P^2) \wedge \cdots \wedge \rho(P^{n-1}, P^n) \wedge \Gamma^n$$

with ι and Γ respectively expressed in terms of the propositions P^0 and P^n representing values of state variables respectively in time points 0 and n. The propositions t for transitions have time tags similarly.

It is also possible to allow several transitions in parallel. See for example (Kautz, McAllester, & Selman 1996). Parallelism is interpreted in the sense of interleaving semantics, that is, parallelity of t_1 and t_2 means that the transitions can take place in either order, first t_1 and then t_2, or vice versa. For this to be well-defined t_1 and t_2 may not interfere, that is, they may not undo changes done by the other nor enable or disable the other transition.

Derivation of Symmetry-Breaking Constraints

A main insight to the symmetries induced by a schematic description is that for two transition relations t and t' *there is at most one member* $\langle \sigma, \tau \rangle \in G$ *of the symmetry group that need to be considered*: the minimal substitution that makes t and t' equal. Intuitively, transition relations $F(a, b, c)$ and $F(q, b, c)$ are symmetric by interchanging a and q; interchanging anything else, like v and w for example, could not lead to finding further symmetry.

Based on Definition 4 we now derive symmetry-breaking constraints. Constants c_1 and c_2 occurring in a schematic description of a transition system are *interchangeable* if they do not explicitly occur in the schematic description of transition relation or in the (schematic) goal or whatever object is used in describing the function E.

Two transition relations t and t' are candidates for the interchangeability test of Definition 4 in a given state if one can be obtained from the other by exchanging c_1 and c_2 (or a higher number of constants.) We assume that the ordering $<$ of all the transition relations orders t before t', that is $t < t'$.

Let $\langle \sigma, \tau \rangle$ be an automorphism that corresponds to the substitution that interchanges c_1 with c_2. Hence $\tau(t) = t'$. Now testing the condition $\sigma(s) = s$ for a state s reduces to testing whether all pairs (v, v') of state variables that are obtained from each other by interchanging c_1 and c_2 have the same truth-value in s. If they have, then a conventional symmetry-breaking constraint should be applied to t and t'. All this can be expressed by the following formula.

$$((v_1 \leftrightarrow v_1') \wedge \cdots \wedge (v_m \leftrightarrow v_m')) \rightarrow (t' \rightarrow t) \qquad (1)$$

So, we extend the formulae $\rho(P^i, P^{i+1})$ of the preceding section by these constraints. No other changes are needed.

Example 7 Consider the problem of getting from city center to the university either by bus or by train, expressed in terms of state variables $V = \{$I=city, I=bus, I=train, I=uni, B=city, B=uni, T=city, T=uni$\}$. I, B and T denote the passenger, the bus, and the train, respectively, and their locations may be the university or the city. Additionally, the passenger may be in the bus or in the train. There are actions for boarding and driving the bus and the train. Boarding a bus or a train in the city center can be expressed by the following schematic formula with $x \in \{\mathrm{B}, \mathrm{T}\}$.

$$\begin{aligned} \text{board-x} \quad = \quad & (\text{I=city} \wedge \text{x=city}) \wedge (\top \leftrightarrow (\text{I=x})') \\ & \wedge (\bot \leftrightarrow (\text{I=city})') \wedge \\ & \textstyle\bigwedge_{p \in V \setminus \{\text{I=x,I=city}\}} (p \leftrightarrow p') \end{aligned}$$

We can identify equivalence classes of states by interchanging the truth-values of pairs of state variables obtained by interchanging B and T. Similarly, transition relations board-T and board-B turn out symmetric. The constraint for breaking the symmetry between boarding the bus and the train is the following.

$$\begin{aligned} &((\text{B=city} \leftrightarrow \text{T=city}) \wedge (\text{B=uni} \leftrightarrow \text{T=uni}) \wedge \\ &(\text{I=bus} \leftrightarrow \text{I=train})) \rightarrow (\text{board-B} \rightarrow \text{board-T}) \end{aligned}$$

■

Symmetry-Breaking for Parallel Transitions

Formula 1 is general enough for breaking symmetries also in the presence of parallel transitions. If there are n symmetric transitions that are enabled and could take place in parallel, there are $n + 1$ equivalence classes of symmetric sets of transition relations: subsets of cardinalities from 0 to n. Our symmetry-breaking constraint allows only one set of transition relations from each equivalence class.

Example 8 Consider switching off n lamps. The following formula is instantiated with $x \in \{1, \ldots, n\}$.

$$\text{switchoff(x)} \quad = \quad \text{on}(x) \wedge (\bot \leftrightarrow (\text{on}(x))')$$

The constraint on switching of every pair $\{i, j\} \subseteq \{1, \ldots, n\}, i < j$ of lamps is

$$(\text{on}(i) \leftrightarrow \text{on}(j)) \rightarrow (\text{switchoff}(i) \rightarrow \text{switchoff}(j)).$$

These constraints say that if lamp i is to be switched off, then also all lamps $j \in \{i+1, \ldots, n\}$ are. If initially all lamps are on, these constraints allow only one subset of each of the $n+1$ equivalence classes of symmetric sets of actions that are possible. ■

However, if there are dependencies between the transitions that are enabled at the same time point in the sense that some of them have mutually contradictory effects or cannot be fired simultaneously without violating the interleaving semantics of parallel transitions, the symmetry-breaking constraints may reduce the amount of parallelism possible. We illustrate this with an example.

Example 9 Consider a set of chess players $1, 2, 3$. Each potential player has two actions, choose White or choose Black. Each color can be chosen by at most one player, and no player can choose more than one color. In this example both players and colors are mutually interchangeable.

We order the possible actions 1W, 2W, 3W, 1B, 2B, 3B according to the players as 1W $>$ 2W $>$ 3W and 1B $>$ 2B $>$ 3B, and according to the colors as 1W $>$ 1B and 2W $>$ 2B and 3W $>$ 3B.

Assume that all potential players are ready to choose, and both colors are available. Now the symmetry-breaking constraints allow 1 to choose White. But all other simultaneous choices are blocked by the constraints. However, in the next time step Black is still available, and as the constraints order 2B before 3B, player 2 will choose Black.

Without these constraints it is possible for player 1 to choose White simultaneously with player 2 choosing Black. So our symmetry-breaking constraints do not allow all the potential parallelism. ■

This kind of dependencies between transitions can be handled by a simple extension to the symmetry-breaking constraints. A symmetry-breaking constraint on t and t' that prefers t to t' is ignored whenever t is blocked by a higher-priority transition t'' that according to the interleaving semantics cannot be parallel to t. This is achieved by weakening the constraint for t and t' with a new antecedent $\neg t''$ in the implication in Formula 1.

So let $t_1, \ldots, t_n$ be all the higher-priority transitions that interfere with t. Then the symmetry-breaking constraint that allows all parallelism that was possible in the original problem formulation without symmetry-breaking is as follows.

$$\begin{aligned}((v_1 \leftrightarrow v_1') \wedge \cdots \wedge (v_m \leftrightarrow v_m') \\ \wedge \neg t_1 \wedge \cdots \wedge \neg t_n) \quad \rightarrow \quad (t' \rightarrow t)\end{aligned} \qquad (2)$$

Theorem 10 *Let the transitions $X = \{t_1, \ldots, t_n\}$ be those that are fired in parallel at a given time point. Then there is a symmetric transition sequence $\tau(t_1), \ldots, \tau(t_n)$ for some $\langle \sigma, \tau \rangle \in G$ that satisfies the symmetry-breaking constraints.*

Proof: We give a proof sketch.

Assume that transitions $X' = \{\tau(t_1), \ldots, \tau(t_n)\}$ are fired in parallel in state $s \in S$ and that X' is the lexicographically first among sets of transitions that are symmetric to X.

We show that all symmetry-breaking constraints are satisfied. Let $(\phi \wedge t_1' \wedge \cdots \wedge t_m') \rightarrow (t' \rightarrow t)$ be any of the constraints. Here $t_1', \ldots, t_m'$ are the higher-priority transitions that interfere with t and ϕ is a conjunction of equivalences $v \leftrightarrow v'$. The constraint is obviously satisfied if $t' \notin X'$ or $t \in X'$ or ϕ is false. So we consider the case in which $t' \in X'$ and $t \notin X'$ and the formula ϕ is true, that is, t' and t are interchangeable and only the transition t' is fired.

It suffices to show that one of the transitions t_i' is in X'. Assume none of them is. Now we can produce a set X'' of transitions applicable in S and that is lexicographically better than X', thus contradicting our assumptions. Let $X'' = (X' \cup \{t\}) \setminus \{t'\}$. Because t and t' are interchangeable in s and t' is applicable in s, also t is applicable in s. Because none of the higher priority transitions is in X', there is nothing that would prevent firing t. And because $t < t'$ and X' and X'' agree on all other transitions, X'' is lexicographically better than X', thus contradicting our assumptions. □

Example: A Simple Planning Problem

We demonstrate all the stages of symmetry reduction by using a simple planning problem that has in recent years been a focus of interest of many planning researchers.

The schematic state variables are at-robot(R), at(B,R), free(H) and carry(B,H), that get instantiated as $R \in \{\text{roomA}, \text{roomB}\}$, $B \in \{\text{ball1}, \text{ball2}\}$ and $H \in \{\text{left}, \text{right}\}$.

The transitions (operators) are move(R,R') for moving the robot from one room to another ($R \neq R'$), pick(B,R,H) picking up a ball in a room, and drop(B,R,H) dropping a ball in a room.

In the initial state the state variables at-robot(roomA), free(left), free(right), at(ball1,roomA) and at(ball2,roomA) are true, and the rest are false. All states satisfying $\text{at}(\text{ball1}, \text{roomB}) \wedge \text{at}(\text{ball2}, \text{roomB})$ are goal states.

Because none of the possible values of B occur explicitly in the schematic definitions of transitions and all the balls are mentioned in the goal symmetrically, all possible values of B are interchangeable. Similarly for H. The rooms, however, are not interchangeable because the balls are required to be in roomB in the goal state.

Next we derive the symmetry-breaking constraints on the transitions for picking up the balls in roomA. Let us order the balls and the hands as ball1 $<$ ball2 and left $<$ right. From this we get the orderings

$$\begin{array}{lcl}\text{pick}(\text{ball1}, \text{roomA}, \text{left}) & < & \text{pick}(\text{ball1}, \text{roomA}, \text{right}) \\ \text{pick}(\text{ball2}, \text{roomA}, \text{left}) & < & \text{pick}(\text{ball2}, \text{roomA}, \text{right}) \\ \text{pick}(\text{ball1}, \text{roomA}, \text{left}) & < & \text{pick}(\text{ball2}, \text{roomA}, \text{left}) \\ \text{pick}(\text{ball1}, \text{roomA}, \text{right}) & < & \text{pick}(\text{ball2}, \text{roomA}, \text{right})\end{array}$$

on transitions. The symmetry-breaking constraint for the fourth pair of transitions is

$$
\begin{array}{l}
(((\mathrm{carry}(\mathrm{ball1},\mathrm{left}) \leftrightarrow \mathrm{carry}(\mathrm{ball2},\mathrm{left})) \\
\wedge(\mathrm{carry}(\mathrm{ball1},\mathrm{right}) \leftrightarrow \mathrm{carry}(\mathrm{ball2},\mathrm{right})) \\
\wedge(\mathrm{at}(\mathrm{ball1},\mathrm{roomA}) \leftrightarrow \mathrm{at}(\mathrm{ball2},\mathrm{roomA})) \\
\wedge(\mathrm{at}(\mathrm{ball1},\mathrm{roomB}) \leftrightarrow \mathrm{at}(\mathrm{ball2},\mathrm{roomB})) \\
\wedge\neg\mathrm{pick}(\mathrm{ball1},\mathrm{roomA},\mathrm{left})) \\
\rightarrow \\
(\mathrm{pick}(\mathrm{ball2},\mathrm{roomA},\mathrm{right}) \rightarrow \mathrm{pick}(\mathrm{ball1},\mathrm{roomA},\mathrm{right}))
\end{array}
$$

Impact on Sizes of SAT Encodings: Example

Given a transition system with T transitions with N occurrences of a constant in a state variable name, and a planning or model-checking problem with D time points, the size of the set of symmetry-breaking constraints is $\mathcal{O}(DNT^2)$. So the size of the set of constraints grows linearly in the length D of the transition sequences sought for, linearly in the number N of state variables, and quadratically in the number T of transitions. In general, the symmetry-breaking constraints may dominate the size of the encodings, but do not affect the asymptotic size of the encodings, because the axioms restricting the parallel firing of interfering transitions similarly have quadratic size. The symmetry-breaking constraints just bring the constant N on top of this. Extending the constraints with the additional antecedents for allowing maximal parallelism adds a further factor T and makes the size of the formulae cubic in the number of transitions in the worst case.

To see how the symmetry-breaking constraints affect the size of concrete problem encodings, we have implemented the symmetry reduction technique as a part of a SAT/QBF based planning system that takes as input in a standard planner input language, translates problem instances into formulae in the propositional logic, and outputs them in a format used by most implementations of satisfiability algorithms that work on formulae in CNF.

The implementation automatically recognizes objects that are symmetric with respect to the operators and the goal description, derives the symmetry-breaking constraints, and adds them to the encoding of transition relations in the propositional logic. Because translation of the constraints into CNF (as required by most of the publicly available satisfiability solvers) may increase their size considerably, we have also implemented a simplification procedure that eliminates redundancies in them: some of the equivalences in the antecedent of a constraint, or their negations, may be completely or partially entailed by the precondition of the lower-priority transition, which often allows simplifying the constraints a lot, in some cases eliminating some of them completely. On some of the example formulae discussed below this cuts the number of clauses to less than half.

The impact of symmetry reduction on a very wide range of computational problems is so well documented that we cannot claim to provide much new information on the topic. For illustrating the applicability of our technique we briefly discuss it in connection with a simple problem from AI planning, just to make concrete what effect adding the symmetry-breaking constraints can have on the size of the problem encodings and runtimes.

			without		with	
balls	truth	len	clauses	time	clauses	time
4	F	6	3564	0.0	3756	0.0
4	T	7	4154	0.0	4378	0.0
6	F	10	10314	1.6	11034	0.1
6	T	11	11342	0.2	12134	0.3
8	F	12	18884	32.4	31556	0.7
8	F	13	20454	663.2	34182	2.8
8	F	14	22024	>900	23816	0.9
8	T	15	23594	8.9	25514	2.4
10	F	18	39942	>900	43542	9.9
10	T	19	42158	141.4	45958	2.5
12	F	22	65316	>900	71652	5.7
12	T	23	68282	>900	74906	5.5
14	F	26	99394	>900	109586	35.7
14	T	27	103214	>900	113798	10.2
16	F	30	143424	>900	158784	7.6
16	T	31	148202	>900	164074	14.8
18	F	34	198654	>900	220686	35.5
18	T	35	204494	>900	227174	169.6
20	F	38	266332	>900	296732	108.1
20	T	39	273338	>900	304538	81.9

Table 1: Solution of the gripper problems with satisfiability algorithms. Symmetry-breaking for pairs of operators differing in only one parameter. The columns show the number of balls in the problem instance, truth-value of the formula, plan length, number of clauses, and runtime in seconds.

Some statistics on the formulae are given in Tables 1 and 2. The first table lists formulae with symmetry-breaking for pairs of operators that differ only in one parameter, and the second table in the general case. Having all constraints doubles the number of clauses related to symmetry-breaking and increases the runtimes. Another interesting phenomenon is the decrease in runtimes when leaving out the antecedents referring to interfering operators: solution lengths increase, but contrary to satisfiability planning and bounded model-checking typically, runtimes decrease.

The formulae are rather big, for example on the 39 step problem instance with 20 balls has 9756 propositions and 304538 clauses. Satisfiability problems this big can be very difficult to solve. The runtimes are for the ZChaff satisfiability solver (Moskewicz *et al.* 2001) and are given in seconds of CPU time (we also tried other solvers like relsat, satz and SATO that are slower on these problems, satz and SATO often very much slower.) The problem instances are from the infamous gripper domain, which involves transporting a number of balls from one room to another by a robot that can carry two balls at a time. All the balls are identical, and also the robot's two hands are identical. The problem is highly symmetric and easy to solve, but if symmetries are not recognized, proving that plans of certain length do not exist – and hence finding optimal plans – becomes combinatorially very difficult for general-purpose planning algorithms.

For each problem instance we give statistics on two (or more) formulae. The one with a highest number of time

			without		with	
balls	truth	len	clauses	time	clauses	time
4	F	6	3564	0.0	3900	0.0
4	T	7	4154	0.0	4546	0.0
6	F	10	10314	1.6	11634	0.2
6	T	11	11342	0.2	12794	0.2
8	F	12	18884	32.4	21764	2.5
8	F	13	20454	663.2	23574	3.4
8	F	14	22024	> 900	25384	2.7
8	T	15	23594	8.9	27194	3.4
10	F	18	39942	> 900	46782	4.8
10	T	19	42158	141.4	49378	8.3
12	F	22	65316	> 900	77460	53.8
12	T	23	68282	> 900	80978	49.2
14	F	26	99394	> 900	119050	48.1
14	T	27	103214	> 900	123626	40.7
16	F	30	143424	> 900	173184	55.7
16	T	31	148202	> 900	178954	74.5
18	F	34	198654	> 900	241494	194.4
18	T	35	204494	> 900	248594	87.0
20	F	38	266332	> 900	325612	179.2
20	T	39	273338	> 900	334178	140.2

Table 2: Solution of the gripper problems with satisfiability algorithms. Symmetry-breaking for all pairs of operators.

points is the one with a solution. This is the satisfiable formula (indicated by T in the "truth" column) and the satisfying assignment represents a plan. The preceding formula or formulae are unsatisfiable and show that no shorter plans exist. On the 8 ball problem the runtimes on proving unsatisfiability grow very fast when no symmetry-breaking constraints are used, and the satisfiability tester is not able to prove optimality of the solution length. In this case we also give runtimes on shorter unsatisfiable formulae.

Related Work

Practically all prior work on symmetries in transition systems is based on dividing states to equivalence classes modulo symmetry, because this allows the more powerful forms of symmetry reduction. A notable exception to this is by Godefroid (1999). He considers stateless model-checking of large software systems in which descriptions of states may be huge and impractical to handle directly, and breaks symmetries in transition sequences instead.

Most symmetry reduction algorithms are embedded in state space traversal algorithms: the successor states (or, for backward traversal, the predecessor states or sets of predecessor states) of a state are computed, each state is mapped to its equivalence class, and each equivalence class is further mapped to its canonical state. Because several successor states may belong to the same equivalence class, it is possible to obtain big reductions in the number of successor states (Starke 1991; Emerson & Sistla 1996). This kind of symmetry reduction has been widely used and is applicable to a wide range of model-checking and planning algorithms, but it is not suitable for satisfiability planning nor for bounded model-checking, as it is difficult to express the computation of the canonical states as propositional formulae.

Symmetry breaking has been considered for propositional satisfiability and constraint satisfaction problems (CSP) that do not represent transition systems, see for example (Crawford *et al.* 1996). Symmetries are viewed as inducing symmetry groups like with transition systems, but as time and change do not have to be considered, it is easier to derive symmetry-breaking constraints.

Joslin and Roy (1997) derive symmetries from a schematic problem representation like we do in this paper. They show how symmetry breaking predicates can be constructed for a representation of planning as a SAT/CSP problem. However, they do not discuss nor solve the problem we have solved in this paper, that is, symmetry breaking at all steps of the plan. It seems that Joslin and Roy break only the symmetries at the first time point.[2] For a certain set of problems they show that even this restricted form of symmetry breaking leads to big efficiency gains.

Some planning researchers (Fox & Long 1999; Guéré & Alami 2001; Fox & Long 2002) have published papers on symmetry reduction techniques that are less general and less powerful than earlier state-of-the-art symmetry reduction techniques.

Guere and Alami call the equivalence classes of symmetric states *shapes*, propose an algorithm that does planning by enumerating these equivalence classes, and show that on highly symmetric problems improvements in runtimes can be obtained.

Fox and Long introduce a notion of symmetries and use them for pruning search trees in the Graphplan algorithm (Fox & Long 1999). Their notion of symmetry does not view transition systems as symmetric, but only certain actions with respect to a single search state are considered either symmetric or asymmetric based on the symmetry of objects in that state, and symmetries initially present are lost forever by taking one action that treats two objects differently. This leads to a very weak form of symmetry reduction, which Fox and Long have later improved (Fox & Long 2002). This improved reduction technique is still strictly weaker than the state-of-the-art symmetry reduction techniques (Starke 1991; Emerson & Sistla 1996) which are directly applicable to the same types of algorithms for traversing state spaces.

Conclusions and Future Work

In this work we have presented a symmetry reduction technique that is applicable to representations of transition systems in the classical propositional logic á la satisfiability planning and bounded model-checking. The work is based on identifying symmetries in sequences of transitions, and from these symmetries we derive constraints on the applicability of transitions. The approach appears to provide a feasible symmetry reduction method. Because the technique operates at the level of transitions without explicitly looking at the successor states they produce, in some cases symmetries are not broken when it were possible.

[2] Alternatively, they introduce symmetry-breaking predicates for all time points, but this would in general not preserve solution existence as we have pointed out.

Some open problems remain. The symmetry reduction technique does not appear to be directly applicable to planning with partial observability. For example, planning without observability, sometimes called conformant planning, can be easily represented as quantified Boolean formulae (Rintanen 1999), but it is not clear how widely our symmetry reduction technique would be applicable in that context. The problem is that conformant planning does not work in the state space but in the belief space induced by the state space, and symmetry reduction should be defined in terms of the asymmetries in the belief state, not in the state space. More concretely, in a given state two transition relations should be considered interchangeable if beliefs concerning the related facts are the same, not just when the actual truth-values of the related facts are the same. One possibility would be to test interchangeability on the basis of the sequence of transitions taken so far.

The lexicographic ordering on transition sequences orders the temporally first transitions first, then the temporally the second, and so on. This roughly corresponds to how symmetry reduction is done in an algorithm that traverses the state space in the forward direction. It would also be possible to reverse the ordering, that is, order the last transition first, which would roughly correspond to symmetry reduction in a backward traversal of the state space. It is not clear what benefits or disadvantages reversing the ordering would have.

References

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In Cleaveland, W. R., ed., *Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of 5th International Conference, TACAS'99*, volume 1579 of *Lecture Notes in Computer Science*, 193–207. Springer-Verlag.

Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry-breaking predicates for search problems. In Aiello, L. C.; Doyle, J.; and Shapiro, S., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, 148–159. Morgan Kaufmann Publishers.

Emerson, E. A., and Sistla, A. P. 1996. Symmetry and model-checking. *Formal Methods in System Design: An International Journal* 9(1/2):105–131.

Fox, M., and Long, D. 1999. The detection and exploitation of symmetry in planning problems. In Dean, T., ed., *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 956–961. Morgan Kaufmann Publishers.

Fox, M., and Long, D. 2002. Extending the exploitation of symmetries in planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, 83–91. AAAI Press.

Freuder, E. C. 1991. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of the 9th National Conference on Artificial Intelligence*, 227–233.

Godefroid, P. 1999. Exploiting symmetry when model-checking software. In *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), October 5-8, 1999, Beijing, China*, 257–275. Kluwer Academic Publishers.

Guéré, E., and Alami, R. 2001. One action is enough to plan. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 439–444. Morgan Kaufmann Publishers.

Ip, C. N., and Dill, D. L. 1996. Better verification through symmetry. *Formal Methods in System Design* 9(1/2):41–75.

Joslin, D., and Roy, A. 1997. Exploiting symmetry in lifted CSPs. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, 197–202. Menlo Park: AAAI Press.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, 1194–1201. Menlo Park, California: AAAI Press.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. In Aiello, L. C.; Doyle, J.; and Shapiro, S., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, 374–385. Morgan Kaufmann Publishers.

Majercik, S. M., and Littman, M. L. 1999. Contingent planning under uncertainty via probabilistic satisfiability. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, 549–556. AAAI Press.

Moskewicz, M. W.; Madigan, C. F.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th ACM/IEEE Design Automation Conference (DAC'01)*, 530–535. ACM Press.

Rintanen, J. 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research* 10:323–352.

Starke, P. H. 1991. Reachability analysis of Petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis* 8(4/5):293–303.