Planning for Temporally Extended Goals as Propositional Satisfiability

Robert Mattmüller
Albert-Ludwigs-Universität
Institut für Informatik
Freiburg, Germany
mattmuel@informatik.uni-freiburg.de

Jussi Rintanen
National ICT Australia and
Australian National University
Canberra, Australia
Jussi.Rintanen@nicta.com.au

Abstract

Planning for temporally extended goals (TEGs) expressed as formulae of Linear-time Temporal Logic (LTL) is a proper generalization of classical planning, not only allowing to specify properties of a goal state but of the whole plan execution. Additionally, LTL formulae can be used to represent domain-specific control knowledge to speed up planning. In this paper we extend SAT-based planning for LTL goals (akin to bounded LTL model-checking in verification) to partially ordered plans, thus significantly increasing planning efficiency compared to purely sequential SAT planning. We consider a very relaxed notion of partial ordering and show how planning for LTL goals (without the next-time operator) can be translated into a SAT problem and solved very efficiently. The results extend the practical applicability of SAT-based planning to a wider class of planning problems. In addition, they could be applied to solving problems in bounded LTL model-checking more efficiently.

1 Introduction

In classical planning, a goal that an agent has to achieve is simply a property of a goal state to reach. TEGs are specifications not only stating desired properties of a final state but of a sequence of states, namely the execution of a plan satisfying the specification. Expressing these goals as formulae of an adequate temporal logic, one can impose more precise constraints on plans than one could with classical reachability goals. So for instance it is possible to specify maintenance goals (some property must be maintained indefinitely), goals stating how the agent should react to some environmental condition, and safety goals that impose a restriction on the agent's behavior not to change certain properties of the world state while trying to achieve a reachability goal.

In planning, temporal specifications are usually either regarded as extended goals or as a means of encoding domain-specific search control knowledge used to guide the planner. Different formalisms have been used, e. g. Temporal Action Logics (TAL) in TALplanner [Doherty and Kvarnström, 2001], and Metric Interval Temporal Logic (MITL) and Linear Temporal Logic (LTL) in TLPlan [Bacchus and Kabanza, 1996; 2000]. Recently, the Planning Domain Definition Language (PDDL) has been extended to express state trajectory constraints [Gerevini and Long, 2005]. Both TALplanner and TLPlan are forward-chaining planners, pruning the search space by progressing the temporal formula. Other approaches include compiling tasks including LTL goals into classical tasks [Cresswell and Coddington, 2004; Baier and McIlraith, 2006] before solving them by using a classical planner.

Bounded model-checking [Biere *et al.*, 1999; Latvala *et al.*, 2004], which is an extension of the planning as satisfiability approach [Kautz and Selman, 1992], can be viewed as a SAT-based technique for planning with TEGs. Since the efficiency of SAT-based planning techniques is often strongly dependent on the notion of partially ordered or parallel plans [Kautz and Selman, 1996; Rintanen *et al.*, 2006], extending the SAT-based LTL model-checking/planning approach to parallel plans may in some cases be very critical to obtain efficient planning. The contribution of this paper is an encoding of constraints that preserve the semantics of LTL formulae under *parallel* plans.

In Section 2 we give a formal description of the problem to be solved. In Section 3 we present the propositional encoding, in which we use the base encoding of planning as satisfiability [Kautz and Selman, 1992] reproduced in Section 3.1 and the translation of LTL formulae to propositional logic [Latvala *et al.*, 2004] reproduced in Section 3.2. Section 3.3 shows our adaption of the encoding of parallelism constraints given in [Rintanen *et al.*, 2006] to tasks with TEGs. Our experiments are described in Section 4.

2 Planning for Temporally Extended Goals as Propositional Satisfiability

2.1 Problem Description

Notation

Let A be a set of propositional variables and Φ a propositional or temporal formula. We write $Lit(A)$ as a shorthand for $A \cup \{\neg a \mid a \in A\}$ and $Lit(\Phi)$ instead of $Lit(var(\Phi))$, where $var(\Phi)$ is the set of variables in Φ. If Lit is a set of literals, we write $Lit^{\Phi} := Lit \cap Lit(\Phi)$. If Φ is a propositional formula, then a occurs positively (negatively) in Φ iff there is an occurrence of a in Φ nested within an even (odd) number of negation signs. A negative literal $\neg a$ occurs positively (negatively) in Φ iff a occurs negatively (positively) in Φ. A literal ℓ occurs in Φ if it occurs positively or negatively in Φ. We write $\mathrm{pos}(\ell, \Phi)$, $\mathrm{neg}(\ell, \Phi)$, and $\mathrm{occ}(\ell, \Phi)$, respectively.

Linear Temporal Logic

In order to specify TEGs, we have to choose a specification language. Here we use propositional LTL [Emerson, 1990] without next-time operator because it has a simple and well-defined semantics and it is sufficiently expressive for many qualitative TEGs.

The set of well-formed $\mathsf{LTL}_{-\mathbf{X}}$ formulae in negation normal form over a set A of propositional variables ($\mathsf{LTL}_{-\mathbf{X}}$ formulae for short) is inductively defined as follows: for all $a \in A$, a and $\neg a$ are $\mathsf{LTL}_{-\mathbf{X}}$ formulae. If φ, φ_1, and φ_2 are $\mathsf{LTL}_{-\mathbf{X}}$ formulae, so are $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\mathbf{F}\varphi$ ("eventually φ"), $\mathbf{G}\varphi$ ("always φ"), $\varphi_1 \mathbf{U} \varphi_2$ ("φ_1 until φ_2") and $\varphi_1 \mathbf{R} \varphi_2$ ("φ_1 releases φ_2").

An $\mathsf{LTL}_{-\mathbf{X}}$ formula φ over A is evaluated along an infinite path in a state space over A. Formally, a Kripke model $\mathfrak{M} = \langle Q, \rightarrow, L \rangle$ is a triple where Q is a set of states, $\rightarrow \subseteq Q \times Q$ is a reflexive binary relation over Q, the transition relation, and $L : Q \rightarrow 2^A$ is a function assigning to each state a propositional valuation of the variables in A. A path in $\mathfrak{M}$ is a function $\pi : \mathbb{N} \rightarrow Q$ such that for all $n \in \mathbb{N}$, $\pi(n) \rightarrow \pi(n+1)$. If π is a path in $\mathfrak{M}$ and $i \in \mathbb{N}$, then the ith suffix of π, $\pi^i : \mathbb{N} \rightarrow Q$, is defined as $\pi^i(j) := \pi(i+j)$ for all $j \in \mathbb{N}$.

For $k, b \in \mathbb{N}$, $k < b$, a path $\pi = u \cdot v^\omega$ consisting of a finite prefix $u = \pi(0), \ldots, \pi(k-1)$ and a loop $v = \pi(k), \ldots, \pi(b-1)$, repeated infinitely often, is called a (b,k)-loop. It is called a b-loop if it is a (b,k)-loop for some $k < b$.

The truth of an $\mathsf{LTL}_{-\mathbf{X}}$ formula φ along a path π, symbolically $\pi \models \varphi$, is now inductively defined as follows:

$$
\begin{aligned}
\pi \models a &:\Leftrightarrow a \in L(\pi(0)) \\
\pi \models \neg a &:\Leftrightarrow a \notin L(\pi(0)) \\
\pi \models \varphi_1 \wedge \varphi_2 &:\Leftrightarrow \pi \models \varphi_1 \text{ and } \pi \models \varphi_2 \\
\pi \models \varphi_1 \vee \varphi_2 &:\Leftrightarrow \pi \models \varphi_1 \text{ or } \pi \models \varphi_2 \\
\pi \models \mathbf{F}\varphi &:\Leftrightarrow \exists i \in \mathbb{N} : \pi^i \models \varphi \\
\pi \models \mathbf{G}\varphi &:\Leftrightarrow \forall i \in \mathbb{N} : \pi^i \models \varphi \\
\pi \models \varphi_1 \mathbf{U} \varphi_2 &:\Leftrightarrow \exists i \in \mathbb{N} : \pi^i \models \varphi_2 \text{ and} \\
&\qquad \forall j \in \{0, \ldots, i-1\} : \pi^j \models \varphi_1 \\
\pi \models \varphi_1 \mathbf{R} \varphi_2 &:\Leftrightarrow \forall i \in \mathbb{N} : \pi^i \models \varphi_2 \text{ or} \\
&\qquad \exists j \in \{0, \ldots, i-1\} : \pi^j \models \varphi_1.
\end{aligned}
$$

If a given path π is a b-loop, the first b states of π together with the value of k contain all the information needed to evaluate φ along π. In the following, all paths we will deal with are of that type.

Let $\bar{q} = \langle q_0, \ldots, q_b \rangle$ be a finite sequence of states such that $q_i \rightarrow q_{i+1}$ for all $i \in \{0, \ldots, b-1\}$ and that there is a $k \in \{0, \ldots, b-1\}$ with $q_b = q_k$. In order to be able to evaluate an $\mathsf{LTL}_{-\mathbf{X}}$ formula along $\bar{q}$, we consider an infinite unraveling of $\bar{q}$: If $q_b \in \{q_0, \ldots, q_{b-1}\}$, say $q_b = q_k$, let $\bar{q}_k^\infty : \mathbb{N} \rightarrow Q$, where $\bar{q}_k^\infty(i) = q_i$, if $i < b$, and $\bar{q}_k^\infty(i) = q_{[(i-b) \bmod (b-k)]+k}$, otherwise. Note that $\bar{q}_k^\infty$ is actually a path, i. e. consecutive states are related by $\rightarrow$. An $\mathsf{LTL}_{-\mathbf{X}}$ formula φ is valid along such a finite sequence $\bar{q}$, written as $\bar{q} \models \varphi$, iff there is a $k \in \{0, \ldots, b-1\}$ such that $q_b = q_k$ and $\bar{q}_k^\infty \models \varphi$.

We make sure that there is a $k \in \{0, \ldots, b-1\}$ such that $q_b = q_k$ by allowing idling in a final state (enforcing it if there is no other loop).

Planning

A planning task is a tuple $\mathcal{P} = \langle A, I, O, \varphi \rangle$, where A is a finite set of Boolean state variables, $I \in 2^A$ is the initial state, O is a finite set of operators, and φ is an $\mathsf{LTL}_{-\mathbf{X}}$ formula with variables in A. Operators have the form $o = \langle p, e, c \rangle$, where p is a propositional formula over A, the precondition of o, e is a finite set of literals over A, the unconditional effects of o, and c is a finite set of pairs $f \rhd d$, consisting of a propositional formula f and a finite set of literals d. These pairs are the conditional effects of o.

The set of all effects of o will be written as $[o]_\Diamond := e \cup \bigcup\{d \mid f \rhd d \in c\}$, the set of unconditional effects as $[o]_\Box := e$, and the set of active effects in a state q as $[o]_q := e \cup \bigcup\{d \mid f \rhd d \in c \text{ and } q \models f\}$ for a single operator o and $[S]_q := \bigcup_{o \in S} [o]_q$ for a set S of operators.

A set S of operators is applicable in a state q if $q \models p$ for all $o \in S$ and $[S]_q$ is consistent. We identify an operator o with the singleton set $\{o\}$, thus o is applicable in q if its precondition is satisfied and its active effects are consistent. For a set S of operators, possibly singleton for sequential plans or empty to model idling, and a state q such that S is applicable in q, the simultaneous application of S in q results in the state q' obtained from q by making the literals in $[S]_q$ true and leaving the other state variables unchanged. We then write $q \xrightarrow{S} q'$ (or $q \xrightarrow{o} q'$ if $S = \{o\}$). Let $\bar{S} = \langle S_0, \ldots, S_{b-1} \rangle$ be a sequence of sets of operators and q_0 a state such that $q_0 \xrightarrow{S_0} q_1 \xrightarrow{S_1} \ldots \xrightarrow{S_{b-1}} q_b$ is defined. Then the sequence of states $\mathit{exec}(q_0, \bar{S}) := \langle q_0, \ldots, q_b \rangle$ is called the execution of $\bar{S}$ in q_0. Finally, let $\bar{S} = \langle S_0, \ldots, S_{b-1} \rangle$ be a sequence of sets of operators, $\bar{\prec} = \langle \prec_0, \ldots, \prec_{b-1} \rangle$ a sequence of binary relations such that $\prec_t$ is a total ordering of S_t, say $o_{t,0} \prec_t \cdots \prec_t o_{t,|S_t|-1}$, for all $t \in \{0, \ldots, b-1\}$, and q_0 a state. Assume that $q_t \xrightarrow{o_{t,0}} q_t^1 \xrightarrow{o_{t,1}} q_t^2 \xrightarrow{o_{t,2}} \ldots \xrightarrow{o_{t,|S_t|-1}} q_{t+1}$ is defined for all $t \in \{0, \ldots, b-1\}$. Then the sequence of states $\mathit{exec}(q_0, \bar{S}, \bar{\prec}) := \langle q_0, q_0^1, q_0^2, \ldots, q_1, q_1^1, q_1^2, \ldots, q_2, \ldots, q_{b-1} \rangle$ is called a linearized execution of $\bar{S}$ in q_0.

A plan of length b for $\mathcal{P} = \langle A, I, O, \varphi \rangle$ is a tuple $\bar{S} = \langle S_0, \ldots, S_{b-1} \rangle$ with $S_t \subseteq O$ for all $t \in \{0, \ldots, b-1\}$ together with a sequence $\bar{\prec} = \langle \prec_0, \ldots, \prec_{b-1} \rangle$ such that (a) $\prec_t$ is a total ordering of S_t for all $t \in \{0, \ldots, b-1\}$ and (b) $\bar{q} = \mathit{exec}(I, \bar{S}, \bar{\prec})$ is defined and $\bar{q} \models \varphi$ in the Kripke model induced by $\mathcal{P}$.

2.2 Reduction to Satisfiability

Planning as satisfiability [Kautz and Selman, 1992] roughly works as follows: given a planning task $\mathcal{P} = \langle A, I, O, \varphi \rangle$, propositional formulae $\Phi_0, \Phi_1, \Phi_2, \ldots$ are generated such that there exists a plan of length b for $\mathcal{P}$ if Φ_b is satisfiable.

The Φ_b are evaluated by using a SAT solver. If Φ_b is unsatisfiable, the evaluation will proceed to Φ_{b+1}, otherwise a plan for $\mathcal{P}$ can be extracted from a satisfying valuation v for Φ_b.

2.3 Solution Quality

The quality of a parallel plan $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ can be measured with respect to its parallel plan length b or its sequential plan length $\sum_{t=0}^{b-1} |S_t|$. We will focus on finding plans with a low parallel plan length because the size of the largest propositional formula to be considered for a given task is roughly proportional to the *parallel* length of a plan corresponding to a satisfying valuation. As SAT solver running times grow exponentially in the formula size in the worst case, obtaining small formulae is particularly important.

3 Propositional Encoding

3.1 Base Encoding

We first give the base encoding $[\![\mathcal{P}]\!]^b_{base}$ of the transition system induced by a planning task $\mathcal{P} = \langle A, I, O, \varphi\rangle$ for a bound b on the plan length first proposed by [Kautz and Selman, 1992], omitting the reachability goal formula:

$$[\![\mathcal{P}]\!]^b_{base} = I_0 \wedge \bigwedge_{t=0}^{b-1} \mathcal{R}_t,$$

where $\mathcal{R}_t$ is the conjunction of precondition axioms $o_t \rightarrow p_t$, effect axioms $o_t \rightarrow \bigwedge e_{t+1}$, conditional effect axioms $\bigwedge_{f \rhd d \in c}((o_t \wedge f_t) \rightarrow \bigwedge d_{t+1})$ for all $o = \langle p, e, c\rangle \in O$, and positive and negative frame axioms $(\neg a_t \wedge a_{t+1}) \rightarrow \bigvee_{o \in O}(o_t \wedge (EPC_a(o))_t)$ and $(a_t \wedge \neg a_{t+1}) \rightarrow \bigvee_{o\in O}(o_t \wedge (EPC_{\neg a}(o))_t)$, respectively, for all $a \in A$. In the frame axioms, $EPC_\ell(\langle p, e, c\rangle)$ is defined as $\top$, if $\ell \in e$, and as $\bigvee\{f \mid f \rhd d \in c \text{ and } \ell \in d\}$, otherwise.

This encoding contains propositional variables a_t for all state variables $a \in A$ and time points $t \in \{0, \ldots, b\}$ as well as o_t for all operators $o \in O$ and time points $t \in \{0, \ldots, b-1\}$ with the intended semantics that a holds at time point t iff a_t is true, and that operator o is applied at time point t iff o_t is true. Where sets of variables or propositional formulae are indexed with some t, this actually denotes the sets or formulae with variables indexed correspondingly.

The following theorem states the correctness of the propositional translation. A proof can be found in [Rintanen *et al.*, 2006].

Theorem 1. *Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task. Then there exists a sequence of sets of operators $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ such that $exec(I, \bar{S})$ is defined iff the formula $[\![\mathcal{P}]\!]^b_{base}$ is satisfiable.* □

3.2 Temporally Extended Goals

In this subsection we reproduce the reduction of the bounded LTL/LTL$_{-\mathbf{X}}$ model-checking problem to propositional satisfiability given in [Latvala *et al.*, 2004], adding the constraint that a satisfying sequence of states *must* contain a loop: let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task and $b \in \mathbb{N}$. Then

$$[\![\mathcal{P}]\!]^b_{bmc} := looptoAx_b \wedge beforeAx_b \wedge uniqueAx_b \wedge inftyAx_b \wedge [\![\varphi]\!]^0_b, \quad \text{where}$$

$$looptoAx_b := \bigwedge_{t=0}^{b-1}\Big(l_t \rightarrow \bigwedge_{a\in A}(a_t \leftrightarrow a_b)\Big),$$

$$beforeAx_b := \neg bef_0 \wedge \bigwedge_{t=1}^{b-1}\Big(bef_t \leftrightarrow (bef_{t-1} \vee l_{t-1})\Big),$$

$$uniqueAx_b := \bigwedge_{t=0}^{b-1}\Big(bef_t \rightarrow \neg l_t\Big), inftyAx_b := \bigvee_{t=0}^{b-1} l_t,$$

with fresh auxiliary variables l_t, bef_t, $t \in \{0, \ldots, b-1\}$ (with the intended semantics that there is a loop from q_{b-1} back to q_t or back to some $q_{t'}, t' < t$, respectively). The recursive translation $[\![\varphi]\!]^0_b$ of φ is defined as

	$t < b$	$t = b$
$[\![a]\!]^t_b$	a_t	$\bigvee_{j=0}^{b-1}(l_j \wedge a_j)$
$[\![\neg a]\!]^t_b$	$\neg a_t$	$\bigvee_{j=0}^{b-1}(l_j \wedge \neg a_j)$
$[\![\varphi_1 \wedge \varphi_2]\!]^t_b$	$[\![\varphi_1]\!]^t_b \wedge [\![\varphi_2]\!]^t_b$	$\bigvee_{j=0}^{b-1}(l_j \wedge [\![\varphi_1 \wedge \varphi_2]\!]^j_b)$
$[\![\varphi_1 \vee \varphi_2]\!]^t_b$	$[\![\varphi_1]\!]^t_b \vee [\![\varphi_2]\!]^t_b$	$\bigvee_{j=0}^{b-1}(l_j \wedge [\![\varphi_1 \vee \varphi_2]\!]^j_b)$
$[\![\mathbf{F}\varphi]\!]^t_b$	$[\![\varphi]\!]^t_b \vee [\![\mathbf{F}\varphi]\!]^{t+1}_b$	$\bigvee_{j=0}^{b-1}(l_j \wedge \langle\!\langle \mathbf{F}\varphi\rangle\!\rangle^j_b)$
$[\![\mathbf{G}\varphi]\!]^t_b$	$[\![\varphi]\!]^t_b \wedge [\![\mathbf{G}\varphi]\!]^{t+1}_b$	$\bigvee_{j=0}^{b-1}(l_j \wedge \langle\!\langle \mathbf{G}\varphi\rangle\!\rangle^j_b)$
$[\![\varphi_1 \mathbf{U} \varphi_2]\!]^t_b$	$[\![\varphi_2]\!]^t_b \vee \big([\![\varphi_1]\!]^t_b \wedge [\![\varphi_1 \mathbf{U}\varphi_2]\!]^{t+1}_b\big)$	$\bigvee_{j=0}^{b-1}(l_j \wedge \langle\!\langle \varphi_1 \mathbf{U}\varphi_2\rangle\!\rangle^j_b)$
$[\![\varphi_1 \mathbf{R} \varphi_2]\!]^t_b$	$[\![\varphi_2]\!]^t_b \wedge \big([\![\varphi_1]\!]^t_b \vee$ $\vee [\![\varphi_1 \mathbf{R}\varphi_2]\!]^{t+1}_b\big)$	$\bigvee_{j=0}^{b-1}(l_j \wedge \langle\!\langle \varphi_1 \mathbf{R}\varphi_2\rangle\!\rangle^j_b)$
$\langle\!\langle \mathbf{F}\varphi\rangle\!\rangle^t_b$	$[\![\varphi]\!]^t_b \vee \langle\!\langle \mathbf{F}\varphi\rangle\!\rangle^{t+1}_b$	$\bot$
$\langle\!\langle \mathbf{G}\varphi\rangle\!\rangle^t_b$	$[\![\varphi]\!]^t_b \wedge \langle\!\langle \mathbf{G}\varphi\rangle\!\rangle^{t+1}_b$	$\top$
$\langle\!\langle \varphi_1 \mathbf{U}\varphi_2\rangle\!\rangle^t_b$	$[\![\varphi_2]\!]^t_b \vee \big([\![\varphi_1]\!]^t_b \wedge \langle\!\langle \varphi_1 \mathbf{U}\varphi_2\rangle\!\rangle^{t+1}_b\big)$	$\bot$
$\langle\!\langle \varphi_1 \mathbf{R}\varphi_2\rangle\!\rangle^t_b$	$[\![\varphi_2]\!]^t_b \wedge \big([\![\varphi_1]\!]^t_b \vee \langle\!\langle \varphi_1 \mathbf{R}\varphi_2\rangle\!\rangle^{t+1}_b\big)$	$\top$

Notice that the translation $[\![\cdot]\!]^t_b$ is closely related to the formula progression procedures used in [Bacchus and Kabanza, 2000; Doherty and Kvarnström, 2001].

The following theorem states the correctness and completeness of the propositional translation. A slightly different formulation as well as a proof can be found in [Latvala *et al.*, 2004].

Theorem 2. *Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task and $b \in \mathbb{N}$. Then there exists a sequence of sets of operators $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ such that $\bar{q} = exec(I, \bar{S})$ is defined and $\bar{q} \models \varphi$ iff the formula $[\![\mathcal{P}]\!]^b_{base} \wedge [\![\mathcal{P}]\!]^b_{bmc}$ is satisfiable.* □

3.3 Parallel Plans

In this section we present our main contribution, being constraints which guarantee that the meaning of LTL$_{-\mathbf{X}}$ formulae is preserved under parallel plans, and a propositional encoding of these constraints.

Theorem 2 tells us how to encode the requirement that there is a sequence $\bar{S}$ such that $exec(I, \bar{S}) \models \varphi$. But we have not yet made sure that $\bar{S}$ is in fact a plan, i. e. there is a sequence $\bar{\prec}$ of corresponding total orderings such that $exec(I, \bar{S}, \bar{\prec}) \models \varphi$. For such a sequence $\bar{\prec}$ to yield an admissible linearization, it must ensure that (a) $\bar{q} = exec(I, \bar{S}, \bar{\prec})$ is defined and if so, that (b) $\bar{q} \models \varphi$. In order to state how this can be achieved, we need some definitions. The first one and the subsequent lemma are due to [Lamport, 1983].

Definition 1. Let $\pi = q_0, q_1, q_2, \ldots$ and $\tilde{\pi} = \tilde{q}_0, \tilde{q}_1, \tilde{q}_2, \ldots$ be two infinite (finite) paths in a Kripke model $\langle Q, \rightarrow, L\rangle$. Then π and $\tilde{\pi}$ are called *stuttering equivalent*, $\pi \sim \tilde{\pi}$ for short, if there are two infinite (finite) sequences of natural numbers $0 = i_0 < i_1 < i_2 < \ldots (< i_n)$ and $0 = j_0 < j_1 < j_2 < \ldots (< j_n)$ such that for all $0 \leq k\ (< n)$: $L(q_{i_k}) = L(q_{i_k+1}) = \ldots = L(q_{i_{k+1}-1}) = L(\tilde{q}_{j_k}) = L(\tilde{q}_{j_k+1}) = \ldots = L(\tilde{q}_{j_{k+1}-1})$. A finite subsequence like $q_{i_k}, q_{i_k+1}, \ldots, q_{i_{k+1}-1}$ of π or $\tilde{\pi}$ consisting of identically labeled states is called a block.

Lemma 3. *Let φ be an $\mathsf{LTL}_{-\mathbf{X}}$ formula, $\mathfrak{M} = \langle Q, \rightarrow, L\rangle$ a Kripke model, where $L : Q \rightarrow 2^A$ for some $A \supseteq var(\varphi)$, and $\pi, \tilde{\pi}$ two infinite (finite) paths in $\mathfrak{M}$ with $\pi \sim \tilde{\pi}$. Then $\pi \models \varphi$ iff $\tilde{\pi} \models \varphi$.* □

The following definition, adapted from a similar one by [Rintanen *et al.*, 2006], is crucial for the rest of this section in that it describes under which circumstances an operator o may or may not be applied before an operator o' in a linearization $\prec$ of a set of operators S if one wants to ensure that for all time points t the application of S_t in q_t in the ordering given by $\prec_t$ is defined in the first place, and if so, that $exec(I, \bar{S}, \bar{\prec}) \models \varphi$.

Definition 2. Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task and $o = \langle p, e, c\rangle, o' = \langle p', e', c'\rangle \in O$. Then *$o$ affects o'* iff $o \neq o'$ and either (1.) there is a literal ℓ over A such that (a.) $\ell \in [o]_\Diamond$ and (b.) (i.) $\mathrm{neg}(\ell, p')$ or (ii.) $\mathrm{occ}(\ell, f')$ for some $f' \rhd d' \in c'$, or (2.) $[o']^\varphi_\Diamond \setminus [o]^\varphi_\Box \neq \emptyset$.

Here, $[o']^\varphi_\Diamond$ is the restriction of $[o']_\Diamond$, i. e. of the set of all conditional and unconditional effect literals of o', to those literals occurring (positively or negatively) in φ. Similarly, for $o = \langle p, e, c\rangle$, $[o]^\varphi_\Box = e \cap Lit(\varphi)$.

The cases correspond to different problems that can potentially arise if o is applied before o' in a linearization, namely to o falsifying the precondition of o' [(1.a.) + (1.b.i.)], o affecting the set of active effects of o' [(1.a.) + (1.b.ii.)], and o and o' putting at risk the stuttering equivalence of $exec(I, \bar{S})$ and $exec(I, \bar{S}, \bar{\prec})$ [(2.)]. The definition gives rise to a condition on the admissibility of a sequence $\bar{\prec}$ of total orderings.

Lemma 4. *Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task, $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ a sequence of sets of operators such that $exec(I, \bar{S}) \models \varphi$, and $\bar{\prec} = \langle \prec_0, \ldots, \prec_{b-1}\rangle$ a sequence such that $\prec_t$ is a total ordering of S_t for all $t \in \{0, \ldots, b-1\}$. If there are no operators $o, o' \in S_t$ such that o affects o' and $o \prec_t o'$ for any $t \in \{0, \ldots, b-1\}$, then $\bar{S}$ together with $\bar{\prec}$ is a plan for $\mathcal{P}$.*

Proof sketch. We have to show that $exec(I, \bar{S}, \bar{\prec})$ is defined and $exec(I, \bar{S}, \bar{\prec}) \models \varphi$. As argued in [Rintanen *et al.*, 2006], in order to show that $exec(I, \bar{S}, \bar{\prec})$ is defined, it suffices to show that no operator $o \in S_t$ can disable another operator $o' \in S_t$ by potentially falsifying its precondition or altering its set of active effects for any $t \in \{0, \ldots, b-1\}$. These two possibilities are ruled out by condition (1.a.) together with (1.b.i.) and (1.b.ii.) of Definition 2 respectively (see [Rintanen *et al.*, 2006] for a detailed proof of that claim). If $exec(I, \bar{S}, \bar{\prec})$ is defined, by using Lemma 3 it is sufficient to show that $exec(I, \bar{S}) \sim exec(I, \bar{S}, \bar{\prec})$ in the Kripke model $\mathfrak{M} = \langle Q, \rightarrow, L\rangle$, where Q and $\rightarrow$ form the state space induced by $\mathcal{P}$ and $L : Q \rightarrow 2^{var(\varphi)}$ is defined by $L(q) = q \cap var(\varphi)$. To see that this is true, consider a single time point $t \in \{0, \ldots, b-1\}$ first. Let $\prec_t$ be the total ordering of S_t and, say, $o_{t,0} \prec_t \cdots \prec_t o_{t,|S_t|-1}$. Then the subsequence of the linearized execution $exec(I, \bar{S}, \bar{\prec})$

$$\cdots \rightarrow q_t \xrightarrow{o_{t,0}} q_t^1 \xrightarrow{o_{t,1}} q_t^2 \xrightarrow{o_{t,2}} \ldots \xrightarrow{o_{t,|S_t|-1}} q_{t+1} \rightarrow \cdots$$

corresponds to the subsequence of the execution $exec(I, \bar{S})$

$$\cdots \rightarrow q_t \xrightarrow{S_t} q_{t+1} \rightarrow \cdots$$

Note that condition (2.) of Definition 2 together with the fact that no operator affects an operator applied later in $\prec_t$ makes sure that for $o \prec_t o'$, $[o']^\varphi_\Diamond \subseteq [o]^\varphi_\Box$ holds and thus all effects in $[S_t]_{q_t}$ relevant to φ, i. e. those effects concerning a variable occurring in φ, are effects of $o_{t,0}$. Therefore, the operators $o_{t,1}, \ldots, o_{t,|S_t|-1}$ do not have an *additional* effect on the labeling L. Thus, $L(q_t^1) = L(q_t^2) = \ldots = L(q_{t+1})$, and $q_t^1, q_t^2, \ldots, q_{t+1}$ form one block of the stuttering equivalence of $exec(I, \bar{S}, \bar{\prec})$ and $exec(I, \bar{S})$. The corresponding block in $exec(I, \bar{S})$ is the singleton $\{q_{t+1}\}$. The other blocks are constructed analogously. □

The next step is to find a propositional formula encoding the condition that at no time point t any two operators o, o' such that o affects o' and $o \prec_t o'$ can be applied simultaneously. For that purpose we define the notion of a disabling graph [Rintanen *et al.*, 2006] for a planning task $\mathcal{P}$.

Definition 3. Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task. A directed graph $\mathcal{G} = \langle O, E\rangle$, where $E \subseteq O \times O$, is a *disabling graph* for $\mathcal{P}$ if E contains all edges (o, o') such that (1.) there is a state q reachable from I with operators in O in which o and o' are simultaneously applicable and (2.) o affects o'.

Let S_t be a set of operators. If the subgraph $\mathcal{G}_t = \langle S_t, E_t\rangle$, $E_t = E \cap (S_t \times S_t)$, of a disabling graph $\mathcal{G} = \langle O, E\rangle$ for $\mathcal{P}$ induced by S_t is acyclic, then there is an ordering $\prec_t$ of S_t in which there are no two operators $o \prec_t o'$ with o affecting o'. In fact, $\prec_t$ can be an arbitrary topological ordering of $\langle S_t, E_t^{-1}\rangle$. As the strongly connected components (SCCs) of a directed graph form a directed acyclic graph, instead of ensuring the acyclicity of $\mathcal{G}_t$, it is sufficient to ensure the acyclicity of the subgraphs of $\mathcal{G}_t$ induced by the SCCs C_i of $\mathcal{G}$, i. e. of $\mathcal{G}_t^i = \langle S_t^i, E_t^i\rangle$, where $S_t^i = S_t \cap C_i$ and $E_t^i = E_t \cap (S_t^i \times S_t^i)$. This can be achieved as follows:

Let $\mathcal{G} = \langle O, E\rangle$ be a disabling graph, $\mathcal{C} = \{C_1, \ldots, C_m\}$ the set of SCCs of $\mathcal{G}$, and $\prec^i$ an arbitrary total ordering of $C_i = \{o_{i_1}, \ldots, o_{i_{|C_i|}}\}$ for all $i \in \{1, \ldots, m\}$, say $o_{i_1} \prec^i \cdots \prec^i o_{i_{|C_i|}}$. For $o^1, \ldots, o^n \in O$, $E, R \subseteq O$ and $\ell \in Lit(A)$, we define formulae stating that there are no $o_i, o_j \in \{o^1, \ldots, o^n\}, i < j$, such that $o_i \in E, o_j \in R$, and

o_i, o_j are applied simultaneously (intuitively, operators in E can disable operators in R wrt ℓ):

$$chain(o^1, \ldots, o^n; E; R; \ell) :=$$
$$\bigwedge\{o^i \rightarrow a^{j,\ell} \mid i < j, o^i \in E, o^j \in R, \{o^{i+1}, \ldots, o^{j-1}\} \cap R = \emptyset\}$$
$$\cup\{a^{i,\ell} \rightarrow a^{j,\ell} \mid i < j, \{o^i, o^j\} \subseteq R, \{o^{i+1}, \ldots, o^{j-1}\} \cap R = \emptyset\}$$
$$\cup\{a^{i,\ell} \rightarrow \neg o^i \mid o^i \in R\},$$

where the $a^{i,\ell}$ are fresh auxiliary variables. Now, the negation of (1) in Definition 2 translates to the conjunction of $chain$ formulae for all time points, SCCs, and literals in φ, if we use the following sets E_ℓ, R_ℓ:

$$E_\ell := \{o \in O \mid \bar{\ell} \in [o]_\Diamond\} \quad \text{and}$$
$$R_\ell := \{\langle p, e, c\rangle \in O \mid \text{pos}(\ell, p) \text{ or ex. } f \rhd d \in c \text{ s.t. occ}(\ell, f)\}.$$

Similarly, the negation of (2) in Definition 2 translates to the conjunction of $chain$ formulae with the sets $E_\ell^\sim, R_\ell^\sim$:

$$E_\ell^\sim := \{o \in O \mid \ell \notin [o]_\Box\} \text{ and } R_\ell^\sim := \{o \in O \mid \ell \in [o]_\Diamond\}.$$

So, the parallelism constraints can be encoded in the formula

$$[\![\mathcal{P}]\!]^b_{lin} := \bigwedge_{t=0}^{b-1} \bigwedge_{i=1}^{m} \Bigg[\bigwedge_{\ell \in Lit(A)} chain(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell; R_\ell; \ell)_t$$
$$\wedge \bigwedge_{\ell \in Lit(\varphi)} chain(o_{i_1}, \ldots, o_{i_{|C_i|}}; E_\ell^\sim; R_\ell^\sim; \tilde{\ell})_t \Bigg].$$

Remark 1. If the valuation corresponding to a sequence of sets of operators $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ satisfies $[\![\mathcal{P}]\!]^b_{lin}$, then for all time points t and SCCs C_i of the disabling graph $\mathcal{G} = \langle O, E\rangle$ used in the construction of $[\![\mathcal{P}]\!]^b_{lin}$, all subgraphs $\mathcal{G}_t^i = \langle S_t^i, E_t^i\rangle$ are acyclic.[1] As the SCCs form an acyclic graph, there is a total ordering $\prec_\mathcal{C}$ on $\mathcal{C}$ such that for all $i, j \in \{1, \ldots, m\}$ with $C_i \prec_\mathcal{C} C_j$, there are no $o \in C_i$ and $o' \in C_j$ such that $(o, o') \in E$. Since $\mathcal{G}_t^i$ is acyclic, there is a total ordering $\prec_t^i$ of S_t^i for each $t \in \{0, \ldots, b-1\}$ and $i \in \{1, \ldots, m\}$ (consistent with the ordering $\prec^i$ used in the construction of $[\![\mathcal{P}]\!]^b_{lin}$) such that there is no pair $o, o' \in C_i$ with $(o, o') \in E$ and $o \prec_t^i o'$. The relations $\prec_t^i$, $i \in \{1, \ldots, m\}$, and $\prec_\mathcal{C}$ can be combined lexicographically, resulting in an ordering $\prec_t$ of S_t. It follows that there is no $t \in \{0, \ldots, b-1\}$ and no $o, o' \in S_t$ such that o affects o' and $o \prec_t o'$.

The following theorem combines the conclusions from Theorem 2 and Lemma 4 with the above Remark.

Main Theorem. *Let $\mathcal{P} = \langle A, I, O, \varphi\rangle$ be a planning task and $b \in \mathbb{N}$. If $[\![\mathcal{P}]\!]^b_{base} \wedge [\![\mathcal{P}]\!]^b_{bmc} \wedge [\![\mathcal{P}]\!]^b_{lin}$ is satisfiable, then there is a plan of length b for $\mathcal{P}$.*

Proof sketch. From Theorem 2 we know that there exists a sequence of sets of operators $\bar{S} = \langle S_0, \ldots, S_{b-1}\rangle$ such that $\bar{q} = exec(I, \bar{S})$ is defined and $\bar{q} \models \varphi$. In order to be able to use Lemma 4, we still need a sequence $\bar{\prec} = \langle \prec_0, \ldots, \prec_{b-1}\rangle$ of corresponding total orderings such that there is no time point $t \in \{0, \ldots, b-1\}$ and no pair $o, o' \in S_t$ of operators with o affecting o' and $o \prec_t o'$. The orderings $\prec_t$ constructed in Remark 1 form such a sequence. Thus, the precondition of Lemma 4 is satisfied and it follows that $\bar{S}$ together with $\bar{\prec}$ is a plan for $\mathcal{P}$, and in particular that $exec(I, \bar{S}, \bar{\prec})$ is defined and $exec(I, \bar{S}, \bar{\prec}) \models \varphi$. $\Box$

[1] We will not formally prove this here. A similar proof can be found in [Rintanen *et al.*, 2006].

4 Experiments

4.1 Setting

We compared the cumulative SAT solver running times until the first satisfiable formula for (a.) the parallel encoding described in Section 3 and (b.) a sequential encoding derived from the parallel one by replacing the parallelism constraints by axioms demanding at most one operator per time point. The evaluation of the formulae corresponding to increasing plan lengths was performed sequentially. Additionally, we compared the (parallel) plan lengths of the resulting plans.

We used two types of planning tasks. First, we considered a simple hand-crafted logistics-like transportation task with three portables and trucks each. The goal was to find an *infinite* plan assuring that the portables are shipped back and forth between two locations indefinitely. The goal formula we used was $\varphi = \bigwedge_{j=1}^{2} \mathbf{GF}\big(\bigwedge_{i=1}^{3} \mathtt{at}(\mathtt{p}_i, \mathtt{d}_{ij})\big)$, where the $\mathtt{p}_i$ are portables and the $\mathtt{d}_{ij}$ are locations.

The other tasks were adapted from the 2006 International Planning Competition. We modified the qualitative preferences tasks from the rovers domain by turning the soft temporal constraints (preferences) into hard constraints and by ignoring the metric function. When changing soft into hard constraints, it turned out that it was necessary to drop some of them in order to keep the tasks solvable. This was done by drawing uniformly at random κ constraints for each task and only retaining the ones drawn.[2] The constraints were translated to $\mathsf{LTL}_{-\mathbf{X}}$ as explained in [Gerevini and Long, 2005]. Unlike the first task above, the rovers tasks lacked explicit nesting of temporal operators[3] and could, if solvable at all, be solved by plans always yielding *finite* executions, apart from infinite idling in a final state. The reachability goals specified in the problem definitions were required to hold in such a final state.

The SAT solver we used was SIEGE V. 4 [Ryan, 2004]. The experiments were run on a PC with 1.8 GHz AMD ATHLON 64 CPU, 768 MB RAM, and a LINUX operating system.

4.2 Results

Table 1 contains results from the logistics task and the modified rovers tasks for $\kappa = 3$.

The second and third columns show the size M of the largest SCC of the computed disabling graph compared to its overall number of nodes $|O|$. The fourth column shows the parallel plan lengths b_p obtained with our parallel encoding, compared to sequential plan lengths b_s in column five. Where an interval $(m, n]$ is given, the shortest sequential plan has length $m < b_\mathrm{s} \leq n$, but we could not precisely determine b_s because of SAT solver running times exceeding our time-out

[2] For $\kappa = 3$, the tasks r-03, r-05 and r-15 remained unsolvable.

[3] There is an implicit nesting of depth two to three in the temporal operators `sometime-after`, `sometime-before` and `at-most-once`.

Task	$\lvert O\rvert$	M	b_p	b_s	r_p	r_s	R_p
logist.	18	12	13	21	0.1	5.8	7
r-01	63	50	7	11	0.1	0.2	6
r-02	53	53	7	9	0.1	0.1	9
r-03	76	49	–	–	–	–	–
r-04	86	24	5	9	0.1	0.1	9
r-05	144	138	–	–	–	–	–
r-06	178	122	13	$(26, 37]$	0.1	1639.7	11
r-07	151	23	6	$(14, 20]$	0.1	1176.7	10
r-08	328	60	6	$(13, 41]$	0.1	3578.0	9
r-09	362	139	10	$(22, 39]$	0.1	2243.8	12
r-10	382	168	7	$(13, 54]$	0.1	> 1 h	13
r-11	436	29	10	$(21, 43]$	0.1	2713.1	12
r-12	366	98	6	$(14, 23]$	0.1	1881.3	10
r-13	749	156	7	$(15, 73]$	0.1	> 1 h	24
r-14	525	34	10	$(21, 35]$	0.1	2214.8	17
r-15	751	240	–	–	–	–	–
r-16	671	139	7	$(14, 61]$	0.3	> 1 h	21
r-17	1227	270	12	$(22, 135]$	0.4	> 1 h	68
r-18	1837	157	6	$(12, 71]$	0.2	> 1 h	155
r-19	2838	237	8	$(14, 108]$	1.0	> 1 h	265
r-20	3976	190	8	$(13, 130]$	2.5	> 1 h	573

Table 1: Results for logistics task and modified rovers tasks.

of 10 min. The values r_p and r_s are the cumulative SAT solver running times up to the first satisfiable formula in seconds in the parallel and sequential case, respectively. To obtain these values, we used a time-out of 120 sec for each single satisfiability test, proceeding to the next formula after time-out. R_p denotes the overall running time needed for parallel planning (PDDL parsing, encoding, SAT solving, decoding) in seconds. The discrepancy between SAT solver and overall running times arises because we used an unoptimized SML program to do the encoding. In particular, for the construction of the disabling graph and the computation of invariants, a speed-up by an order of magnitude appears to be possible.

5 Conclusion

We combined existing techniques for planning as satisfiability, bounded LTL model-checking, and parallel planning in order to obtain an efficient method of planning for TEGs. The use of a disabling-graph-based encoding of constraints ensuring the stuttering equivalence of a parallel and at least one corresponding sequential plan execution allowed us to extend parallel planning to planning for TEGs. Our experimental results show that, like in classical SAT-based planning [Kautz and Selman, 1996] and in Graphplan [Blum and Furst, 1995], admitting parallelism can noticeably speed up planning for TEGs. By using the non-sequential formula evaluation strategies given in [Rintanen *et al.*, 2006, Sections 5.2, 5.3], which are orthogonal to the contributions of the present paper, planner running times might be further reducible.

Acknowledgments

This work was partly supported by the DFG within SFB/TR 14 AVACS, by the EC under contract no. FP6-004250-CoSy, and by the BMBF through the DESIRE project. Jussi Rintanen's research was supported by National ICT Australia (NICTA) in connection with the DPOLP project. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

References

[Bacchus and Kabanza, 1996] F. Bacchus and F. Kabanza. Planning for Temporally Extended Goals. In *Proc. 13th AAAI'96*, pages 1215–1222, 1996.

[Bacchus and Kabanza, 2000] F. Bacchus and F. Kabanza. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artif. Intell.*, 116(1-2):123–191, 2000.

[Baier and McIlraith, 2006] J. Baier and S. McIlraith. Planning with First-Order Temporally Extended Goals Using Heuristic Search. In *Proc. 21st AAAI'06*, pages 788–795, 2006.

[Biere *et al.*, 1999] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic Model Checking Without BDDs. In *Proc. 5th TACAS'99*, pages 193–207, 1999.

[Blum and Furst, 1995] A. L. Blum and M. L. Furst. Fast Planning Through Planning Graph Analysis. In *Proc. 14th IJCAI'95*, pages 1636–1642, 1995.

[Cresswell and Coddington, 2004] S. Cresswell and A. M. Coddington. Compilation of LTL Goal Formulas into PDDL. In *Proc. 16th ECAI'04*, pages 985–986, 2004.

[Doherty and Kvarnström, 2001] P. Doherty and J. Kvarnström. TALplanner: A Temporal Logic Based Planner. *AI Magazine*, 22(3):95–102, 2001.

[Emerson, 1990] E. A. Emerson. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, pages 995–1072. MIT Press, Cambridge, MA, 1990.

[Gerevini and Long, 2005] A. Gerevini and D. Long. Plan Constraints and Preferences in PDDL3. Technical Report RT 2005-08-47, Dept. of Electronics for Automation, University of Brescia, Italy, August 2005.

[Kautz and Selman, 1992] H. Kautz and B. Selman. Planning as Satisfiability. In *Proc. 10th ECAI'92*, pages 359–363, 1992.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proc. 13th AAAI*, pages 1194–1201, 1996.

[Lamport, 1983] L. Lamport. What Good is Temporal Logic? In *Inform. Processing 83*, pages 657–668, 1983.

[Latvala *et al.*, 2004] T. Latvala, A. Biere, K. Heljanko, and T. Junttila. Simple Bounded LTL Model Checking. In *Proc. 5th FMCAD'04*, pages 186–200, 2004.

[Rintanen *et al.*, 2006] J. Rintanen, K. Heljanko, and I. Niemelä. Planning as Satisfiability: Parallel Plans and Algorithms for Plan Search. *Artif. Intell.*, 180(12–13):1031–1080, 2006.

[Ryan, 2004] L. Ryan. Efficient Algorithms for Clause-Learning SAT Solvers. Master's thesis, Simon Fraser University, 2004.