The Common Fragment of CTL and LTL

Monika Maidl
Siemens Corporate Technology
Otto-Hahn-Ring 6, 81739 München, Germany
Monika.Maidl@mchp.siemens.de

Abstract

It is well-known that CTL *and* LTL *have incomparable expressive power. In this paper, we give an inductive definition of those* ACTL *formulas that can be expressed in* LTL. *In addition, we obtain a procedure to decide whether an* ACTL *formula lies in* LTL, *and show that this problem is* PSPACE *complete. By omitting path quantifiers, we get an inductive definition of the* LTL *formulas expressible in* ACTL. *We can show that the fragment defined by our logic represents exactly those* LTL *formulas the negation of which can be represented by a 1-weak Büchi automaton and that for this fragment, the representing automaton can be chosen to be of size linear in the size of the formula.*

1. Introduction

The propositional temporal logics CTL and LTL are the most commonly used specification logics for model checking. For complex and safety-critical systems, model checking appears as an attractive method of verifying the correctness of designs. Several model-checking tools have been developed: On the one hand, SMV and VIS [18, 1], using CTL as a specification logic; and on the other hand, SPIN, COSPAN [14, 9] using LTL; the pros and cons of each formalism with respect to model checking have been the subject of a lively debate [6, 23, 11, 22].

While LTL formulas describe a property of a computation sequence, in CTL every temporal operator has to be preceded by a path quantifier, and hence such a formula expresses a property of a computation tree. The two logics are relateable by considering them to be properties of systems: a system can be considered as a computation tree, but also a as a set of computation sequences.

The set of states that satisfy a CTL formula (and more general the formulas of the alternation-free μ-calculus) can be characterized as fixed point, and this allows efficient model-checked with symbolic methods, most prominently by using Binary Decision Diagrams (BDDs) [2]. The linear character of LTL offers a more natural semantics especially for open systems and assumption-commitment verification: A system satisfies an LTL property if all its computation sequences satisfy it. For checking that the property holds regardless of how the environment behaves, the behaviour of the environment can be added to the system, so that more computation sequences correspond to it, reflecting the possible environment steps. Assumption-commitment verification can be done by just restricting to the set of computation sequences that satisfy the assumption. So when dealing with open systems, it is advantageous to use properties that can be expressed simultaneously in CTL and LTL.

The relationship between linear- and branching-time temporal logics has been studied by several researchers. Clarke and Draghicescu in [3] gave a characterization of the CTL formulas that can be expressed in LTL. Kupferman and Vardi [13] solved the opposite problem of deciding whether an LTL formula can be specified in the alternation-free μ-calculus.

This paper is concerned with the formulas that can be expressed simultaneously in LTL and CTL, where we restrict ourselves to ACTL, the fragment of CTL that uses only universal path quantifiers. We present an inductive definition of those ACTL formulas that are expressible in LTL, which we call ACTL^{det}. Josko [10] also approached the problem and defined a fragment of ACTL having the property that all its formulas are equivalent to the universal quantification of an LTL formula. It is a proper subset of ACTL^{det}.

The proof that ACTL^{det} covers this fragment is built on a characterization of the fragment by a certain property of the tableau of an ACTL formula. So deciding whether an ACTL formula has an equivalent LTL formula can be reduced to deciding this property for the tableau of the ACTL formula. This gives a PSPACE lower bound, and it follows that the problem is PSPACE complete.

For an ACTL^{det} formula, an equivalent LTL formula is obtained by just omitting all path quantifiers. When doing so in the inductive definition of ACTL^{det}, one gets an inductive definition of a subset of LTL, which we call LTL^{det}. It follows immediately that LTL$^{\text{det}}$ characterizes all LTL formulas that are expressible in ACTL.

It turns out that LTL$^{\text{det}}$ characterizes yet another class of LTL formulas: Those LTL formulas the negation of which can be represented by a 1-weak Büchi automaton. 1-weak Büchi automata have an easier acceptance condition than general Büchi automata. Model checking of an LTL formula ϕ, as used in SPIN [9, 8] and COSPAN [14], is based on the construction of a Büchi automaton $A_{\neg\phi}$ representing $\neg\phi$. It is checked whether the product of the model with $A_{\neg\phi}$ has a fair path. The size of $A_{\neg\phi}$ influences the efficiency of model checking. While for general LTL formulas, $A_{\neg\phi}$ may have a size exponential in the size of ϕ, we can show that for LTL$^{\text{det}}$, $A_{\neg\phi}$ can be chosen to be linear in the size of ϕ. The easier acceptance condition also makes it possible to check for existence of fair paths more efficiently.

Using LTL formulas with a negation representable by a linear 1-weak automaton has another advantage: It makes it possible to reduce model checking of this logic to model checking of simple formulas of form $\mathsf{AG}\,p$ and $\mathsf{AG}\,\mathsf{AF}\,p$, where p is a state predicate. For these formulas, it is possible to apply special methods to encrease efficiency, see e. g. [16], which also contains approaches for verifying these formulas for systems with infinite state space.

2. Definitions

Temporal logics specify properties of a succession of (system-)states. In propositional temporal logics, properties of states are expressed by predicates over the system variables V, where for example for boolean variables v, v is a predicate, and for enumerative variables, equations between terms of the same sort are predicates. $Pred(V)$ denotes the predicates over V. Models for predicates are valuations for the variables in V; we call such valuations *states* over V, and denote by $States(V)$ the set of states over V. Possible successions of states, i.e. models for temporal logics, are defined by *Kripke structures* (Q, Q_0, R) over V, where $Q \subseteq States(V)$, $Q_0 \subset Q$ is the set of initial states and $R \subseteq Q \times Q$ is the transition relation.

Sometimes *Kripke structures with fairness* (Q, Q_0, R, F) have to be used to model a system, where the fairness constraint F is used to distinguish relevant paths from those presenting impossible behaviour. We use fairness constraints F that are sets of state predicates. A *path of a Kripke structure* (Q, Q_0, R, F) is a sequence π of states such that $(\pi(i), \pi(i+1)) \in R$ for all $i \geq 0$. A *fair path* is a path satisfying every predicate in F at infinitely many positions.

The formulas of linear time logic (LTL) [20] are inductively defined by using temporal operators: Predicates are in LTL and if ψ, ψ_1 and ψ_2 are LTL formulas, so are $\mathsf{X}\,\psi$ ("next"), $\psi_1\,\mathsf{U}\,\psi_2$ ("until"), $\psi_1\,\mathsf{W}\,\psi_2$ ("unless")[1], $\psi_1 \wedge \psi_2$ and $\neg\psi_1$.

Models of LTL formulas are sequences of states. A Kripke structure together with a state s_0 satisfies an LTL formula ϕ if all its paths starting in s_0 satisfy ϕ.

The branching time logic CTL [4] is defined by using the same temporal operators, but every temporal operator is prefixed by a path quantifier, either an existential quantifier (E) or a universal one (A), meaning that either at least one or all paths starting in a state satisfy the quantified formula. A model for a CTL formula is a Kripke structure together with a state s_0.

ACTL is the fragment of those CTL formulas that contain, when in negation normal form, only A as a quantifier.

In using Kripke structures as models of both LTL and CTL, the two logics become comparable. We use $\Leftrightarrow$ to denote equivalence between formulas. In particular, for an LTL formula ψ and a CTL formula ϕ, $\phi \Leftrightarrow \psi$ holds if for all Kripke structures $\mathcal{S}$ and states s of $\mathcal{S}$, $\mathcal{S}, s \models \phi$ iff $\mathcal{S}, s \models \psi$.

3. Linear-time restriction of CTL: ACTL$^{\text{det}}$

The following characterization of CTL formulas that can be expressed in the linear time formalism was given by Clarke and Draghicescu [3], where for a CTL formula ϕ, we denote the result of removing all path quantifiers from ϕ by ϕ^d.

Lemma 1 ([3]) *Let ϕ be a* CTL *formula. Then there is an* LTL *formula ψ such that ϕ and ψ are equivalent iff ϕ is equivalent to ϕ^d.*

It can be shown [3] that $\mathsf{AF}\,\mathsf{AG}\,p$ is not equivalent to $\mathsf{FG}\,p$, which implies that the ACTL formula $\mathsf{AF}\,\mathsf{AG}\,p$ cannot be expressed in LTL. The difference between $\mathsf{AF}\,\mathsf{AG}\,p$ and $\mathsf{FG}\,p$ can be described as follows: Satisfaction of $\mathsf{AF}\,\mathsf{AG}\,p$ forces all paths to have a state s such that all paths starting from s always satisfy p. In contrast to that, $\mathsf{FG}\,p$ is satisfied if on all paths a state s is reached such that all the following states on the path satisfy p, but there might be other paths starting from s which reach states satisfying $\neg p$.

Hence in the definition of ACTL$^{\text{det}}$, we must exclude formulas that contain a path quantifier forcing all paths starting at some state to have a certain property, while the same formula without path quantifiers requires only one path to have that property. We inductively define ACTL$^{\text{det}}$ formulas by using the clauses of the inductive definition of ACTL in restricted form:

Definition 1 (ACTL$^{\text{det}}$)
- $p \in Pred(V)$,

[1] This operator is not necessary in a logic that is closed under negation like LTL, but since later on we consider logics which do not have this property, we introduce this operator here. For technical reasons concerning ACTL$^{\text{det}}$, we do not use the operator V instead of W, which would be the exact dual of U.

- for ACTL$^{\det}$ formulas ϕ_1 and ϕ_2 and a predicate p: $\phi_1 \wedge \phi_2$, $\mathsf{AX}\,\phi_1$, $(p \wedge \phi_1) \vee (\neg p \wedge \phi_2)$, $\mathsf{A}\,(p \wedge \phi_1)\,\mathsf{U}\,(\neg p \wedge \phi_2)$, $\mathsf{A}\,(p \wedge \phi_1)\,\mathsf{W}\,(\neg p \wedge \phi_2)$.

Remark. For an ACTL$^{\det}$ formula ϕ, $\mathsf{A}\,\phi\,\mathsf{W}\,p$ can be expressed in ACTL$^{\det}$, since $\mathsf{A}\,\phi\,\mathsf{W}\,p \Leftrightarrow \mathsf{A}\,(\phi \wedge \neg p)\,\mathsf{W}\,p$. A special case is $\mathsf{AG}\,\phi$. Similarly, $\mathsf{A}\,\phi\,\mathsf{U}\,p$ can be expressed in ACTL$^{\det}$.

3.1. The tableau construction

In order to prove that ACTL$^{\det}$ is exactly the set of ACTL formulas equivalent to LTL formulas, we use a tableau construction for ACTL formulas. It differs from other tableau constructions, e.g. [15] and [17], in that not all possible states are generated, but only those which exactly characterize the formula that has to be satisfied at that point. It is very similar to the construction given by Gerth, Peled, Vardi and Wolper [7], except that we do not split state predicates.

The tableau construction is based on a splitting of formulas into a propositional part and a "next" part. We define an operator Φ that generates the splitting:

Definition 2

1. Exp_1 applies one expansion step to its arguments.

   $Exp_1 : 2^{2^{ACTL}} \longrightarrow 2^{2^{ACTL}}$ is defined by $Exp_1(\{Z_0,\ldots,Z_n\}) \stackrel{\text{def}}{=} \bigcup_{0\leq i\leq n} Exp_2(Z_i)$.

   The definition satisfies the following property:

   There is some $Z \in Exp_1(\{Z_0,\ldots,Z_n\})$ such that $\mathcal{S},s \models \bigwedge_{\psi\in Z}\psi$ iff $\mathcal{S},s \models \bigvee_i(\bigwedge_{\psi\in Z_i}\psi)$.

2. $Exp_2 : 2^{ACTL} \longrightarrow 2^{2^{ACTL}}$ is defined by $Exp_2(\{\phi_0,\ldots,\phi_n\}) \stackrel{\text{def}}{=} \{Z_0 \cup \cdots \cup Z_n \mid (Z_0,\ldots,Z_n) \in \prod_{0\leq j\leq n} Exp_3(\phi_j)\}$

   The definition has the following property:

   There is some $Z \in Exp_2(\{\phi_0,\ldots,\phi_n\})$ such that $\mathcal{S},s \models \bigwedge_{\psi\in Z}\psi$ iff $\mathcal{S},s \models \bigwedge_i \phi_i$.

3. $Exp_3(\phi)$ splits a single formula into a predicate part and a "next" part. For doing so, there might be several alternatives. So $Exp_2(\phi_0,\ldots,\phi_n)$ is the set of sets that are formed by choosing one alternative for each ϕ_i.

   - $Exp_3(p) = \{\{p\}\}$ for state predicates p,
   - $Exp_3(\mathsf{AX}\,\phi) = \{\{\mathsf{AX}\,\phi\}\}$,
   - $Exp_3(\psi_1 \wedge \psi_2) = \{\{\psi_1,\psi_2\}\}$,
   - $Exp_3(\psi_1 \vee \psi_2) = \{\{\psi_1\},\{\psi_2\}\}$,
   - $Exp_3(\mathsf{A}\,\psi_1\,\mathsf{W}\,\psi_2) = \{\{\psi_1, \mathsf{AX}\,\mathsf{A}\,\psi_1\,\mathsf{W}\,\psi_2\},\{\psi_2\}\}$,
   - $Exp_3(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) = \{\{\psi_1, \mathsf{AX}\,\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2\},\{\psi_2\}\}$.

   Obviously the following holds:

   $\mathcal{S},s \models \psi$ iff there is some element $Z \in Exp_3(\psi)$ such that $\mathcal{S},s \models \bigwedge_{\psi'\in Z}\psi'$.

4. - $\Phi(\emptyset) = \{\{\text{true}\}\}$, and
   - $\Phi(\phi_0,\ldots,\phi_n) = Exp_1^m(\{\{\phi_0,\ldots,\phi_n\}\})$, where m satisfies $Exp_1(Exp_1^m(\{\{\phi_0,\ldots,\phi_n\}\})) = Exp_1^m(\{\{\phi_0,\ldots,\phi_n\}\})$. Since the size of non-terminal formulas (i.e. those not of form p or $\mathsf{AX}\,\phi$) is decreasing in each expansion step, there is some $m \leq max\{|\phi_0|,\ldots,|\phi_n|\}$ having that property.

   An element $Y \cup X \in \Phi(\phi_0,\ldots,\phi_2)$, where Y is a set of state predicates and X is a set of formulas of the form $\mathsf{AX}\,\psi$, is represented as the pair $(\bigwedge Y, X)$.

The following lemma states that the result of transforming a formula ϕ into state predicates and "next" part by applying Φ is equivalent to ϕ.

Lemma 2 *Let ϕ be an* ACTL *formula.*

(a) Let $\mathcal{S}$ be a Kripke structure and let s be a state in $\mathcal{S}$. Then $\mathcal{S},s \models \phi$ iff there is some $(p,X) \in \Phi(\phi)$ such that $\mathcal{S},s \models p \wedge \bigwedge_{AX\psi\in X} AX(\psi)$.

(b) Let π be a path of states. Then $\pi \models \phi^d$ iff there is an element $(p,X) \in \Phi(\phi)$ such that $\pi \models p \wedge \bigwedge_{AX\psi\in X} X\psi^d$.

Definition 3 (Tableau T_ϕ)
The tableau $T_\phi = (Q_\phi, Q_{\phi_0}, R_\phi, F_\phi)$ for ϕ is defined as follows:

1. For a set of ACTL formulas X, let $\widehat{X} \stackrel{\text{def}}{=} \{\phi \mid \mathsf{AX}\,\phi \in X\}$.

2. The set of states Q_ϕ consists of elements of the form (p,X), where p is a satisfiable conjunction of predicates and X is a set of formulas of the form $\mathsf{AX}\,\phi'$.

   The states Q_ϕ of T_ϕ are defined inductively as follows:

   - If $(p,X) \in \Phi(\phi)$ and p is satisfiable, then (p,X) is in Q_ϕ;
   - if $(p,X) \in Q_\phi$, then all $(p',X') \in \Phi(\widehat{X})$ such that p' is satisfiable are in Q_ϕ.

3. The transition relation $R_\phi \subseteq Q_\phi \times Q_\phi$ of T_ϕ is:

$$R_\phi((p,X),(p',X')) \text{ iff } (p',X') \in \Phi(\hat{X}).$$

4. The initial states are $Q_{\phi_0} \stackrel{\text{def}}{=} \{(p,X) \mid (p,X) \in \Phi(\phi)\}$.

5. Let $liveness(\phi) \in 2^{ACTL}$ be the set of all subformulas of ϕ of form $\mathsf{A}\,\phi_1\,\mathsf{U}\,\phi_2$.

Assume that the states of T_ϕ are numbered, and let $n(p,X)$ be the number of state (p,X). T_ϕ can be considered as a Kripke structure over $\{pos_\phi\}$ when considering a state (p,X) to be a valuation of $\{pos_\phi\}$ by $(p,X)(pos_\phi) \stackrel{\text{def}}{=} n(p,X)$. (Remember that we defined the states of a Kripke structure over a set of variables V to be valuations for V.)

- $prom(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2) \stackrel{\text{def}}{=} \{(p,X) \in Q_\phi \mid \mathsf{AX}\,\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2 \in X\}$,
- $ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2) \stackrel{\text{def}}{=} \{(p,X) \in Q_\phi \mid \exists (q,Y) \in \Phi(\psi_2)$ s.t. $Y \subseteq X$ and $p \Rightarrow q\}$.

The fairness constraint F_ϕ of T_ϕ is $F_\phi \stackrel{\text{def}}{=} \{\bigwedge_{(p,X)\in prom(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)\setminus ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)} pos_\phi \neq n(p,X) \mid \mathsf{A}\,\phi_1 \,\mathsf{U}\, \phi_2 \in liveness(\phi)\}$.

Example The tableau for $\phi = \mathsf{AG}\,(p \vee \mathsf{AX}\,(\mathsf{A}\,p_1 \,\mathsf{U}\, p_2))$ is shown in Figure 1 on the following page. The initial states are 0 and 1, and the fairness condition is $\{pos_\phi \neq 1 \wedge pos_\phi \neq 4 \wedge pos_\phi \neq 5\}$.

We can characterize satisfaction of an ACTL formula ϕ by the *simulation order* (see [19, 5]) between a Kripke structure and the tableau T_ϕ. This use of the simulation order was introduced by Long [15].

Let $\mathcal{S} = (Q, Q_0, R, F)$ be a Kripke structure and $T_\phi = (Q_\phi, Q_{0\phi}, R_\phi, F_\phi)$ the tableau for an ACTL formula ϕ. A relation $\preccurlyeq \subseteq Q \times Q_\phi$ is a *simulation* of $\mathcal{S}$ by T_ϕ if the following holds: For states s and (p,X), $s \preccurlyeq s'$ implies that:

- $s \models p$, and
- for every fair path π in $\mathcal{S}$ starting at s there is a fair path π' starting at (p,X) in T_ϕ such that $\pi(n) \preccurlyeq \pi'(n)$ for all $n \geq 0$.

$\mathcal{S}, s \preccurlyeq T_\phi, (p,X)$ holds if there is a simulation $\preccurlyeq$ of $\mathcal{S}$ by T_ϕ such that $s \preccurlyeq (p,X)$. $\mathcal{S} \preccurlyeq T_\phi$ holds if for all $s \in Q_0$ there is a state $(p,X) \in Q_{0\phi}$ such that $\mathcal{S}, s \preccurlyeq T_\phi, (p,X)$.

Lemma 3 *Let $\mathcal{S} = (Q, Q_0, R, F)$ be a Kripke structure over V and ϕ be an* ACTL *formula, then $\mathcal{S}, s \models \phi$ for all initial states s iff $\mathcal{S} \preccurlyeq T_\phi$.*

The proof of Long [15] for this statement, which was conducted for a different tableau construction, can be easily adopted to ours.

Next we define the containment relation between a Kripke structure and a tableau T_ϕ, which plays an analogous role for an LTL formula ϕ^d as the simulation relation for an ACTL formula ϕ.

Let $\mathcal{S} = (Q, Q_0, R, F)$ be a Kripke structure and ϕ an ACTL formula. Let π_S be a path in $\mathcal{S}$ and π_ϕ be a path in T_ϕ. Then $\pi_S \sim \pi_\phi$ if for all i it holds that $\pi_S(i) \models (\pi_\phi(i))_0$.

$\mathcal{S} \subseteq T_\phi$ ($\mathcal{S}$ is *contained* in T_ϕ) if for all fair paths π_S in $\mathcal{S}$ starting in an initial state there is a fair path π_ϕ in T_ϕ starting in an initial state such that $\pi_S \sim \pi_\phi$.

Lemma 4 *Let $\mathcal{S}$ be a Kripke structure and let ϕ be an* ACTL *formula, then $\mathcal{S}, s \models \phi^d$ for all initial states s iff $\mathcal{S} \subseteq T_\phi$.*

The following characterization follows directly from Lemma 3 and Lemma 4.

Lemma 5 *For an* ACTL *formula ϕ, $\phi \Leftrightarrow \phi^d$ iff for any Kripke structure $\mathcal{S}$ it is the case that $\mathcal{S} \subseteq T_\phi$ iff $\mathcal{S} \preccurlyeq T_\phi$.*

3.2. Characterization of ACTL$^{\text{det}}$ by deterministic tableaux

Lemma 5 implies that an ACTL formula ϕ has an equivalent LTL formula if the notions $\mathcal{S} \subseteq T_\phi$ and $\mathcal{S} \preccurlyeq T_\phi$ collapse. We are looking for a property of tableaux that guarantees this. It suffices to require that if for a node (p,X) of T_ϕ, $s \models p$ and s' is a successor of s, then $s' \models p'$ holds for at most one successor (p',X') of (p,X). This can be guaranteed by requiring that in the tableau, the propositional parts of the successors of a node are mutually exclusive; such a tableau we call deterministic.

There are cases where a tableau is not deterministic but can easily be turned into a deterministic one while not changing its meaning. Consider the formula $\mathsf{A}\,p \,\mathsf{U}\, q$. Its tableau has two states: $(p, \{\mathsf{AX}\,\mathsf{A}\,p \,\mathsf{U}\, q\})$ and $(q, \{\})$, and it is not deterministic if $p \wedge q$ is satisfiable. But since $\mathsf{AX}\,\mathsf{A}\,p \,\mathsf{U}\, q \Rightarrow$ true, changing the first state to $(p \wedge \neg q, \{\mathsf{AX}\,\mathsf{A}\,p \,\mathsf{U}\, q\})$ does not change the meaning of the tableau. So if a state has two successors (p_1, X_1) and (p_2, X_2) such that $p_1 \wedge p_2$ is satisfiable and $\bigwedge_{\rho \in X_2} \rho \Rightarrow \bigwedge_{\rho \in X_1} \rho$, we want to replace (p_2, X_2) by $(p_2 \wedge \neg p_1, X_2)$.

This is however not allowed in all cases. Consider the formula $\mathsf{AF}\,\mathsf{AG}\,p$: The states are (true, $\{\mathsf{AX}\,(\mathsf{AF}\,\mathsf{AG}\,p)\}$) and $(p, \{\mathsf{AX}\,\mathsf{AG}\,p\})$. Since $\mathsf{AX}\,\mathsf{AG}\,p \Rightarrow \mathsf{AX}\,\mathsf{AF}\,\mathsf{AG}\,p$ holds, the propositional part of the second state would be changed to $p \wedge \neg$true. So the only possible path with satisfiable propositional parts would be $(p, \{\mathsf{AX}\,\mathsf{AG}\,p\})^\omega$, which is not fair, and hence this modification would change the meaning of $T_{\mathsf{AF}\,\mathsf{AG}\,p}$. So replacement should only be applied if fulfilling states are not restricted.

In the following, for a state (p,X) we use X instead of $\bigwedge_{\rho \in X} \rho$ when it is clear from the context that a formula is required and not a set.

Definition 4

1. We give an algorithm for making T_ϕ deterministic if possible; the algorithm constructs a structure T_ϕ^{simp}. For uniformity reasons, we define the algorithm for $\mathsf{AX}\,\phi$ instead of ϕ. $T_{\mathsf{AX}\,\phi}$ has initial state (true, $\{\mathsf{AX}\,\phi\}$)

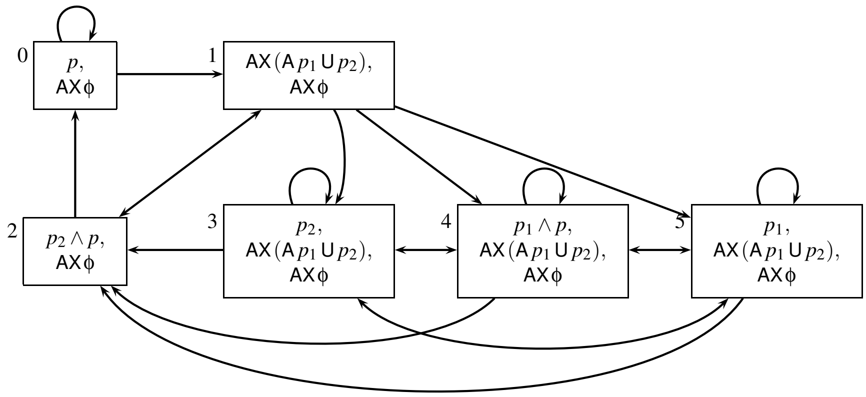

Figure 1. Tableau of an ACTL **formula**

and the successors of $(\text{true}, \{\mathsf{AX}\phi\})$ are the initial states of $T_{\mathsf{AX}\phi}$.

(i) Normalization:
If for two successors (p_1,X_1) and (p_2,X_2) of a state, $X_1 \Leftrightarrow X_2$ holds (note that CTL is decidable), then they are replaced by $(p_1 \wedge \neg p_2, X_1 \cup X_2)$, $(p_2 \wedge \neg p_1, X_1 \cup X_2)$ and $(p_1 \wedge p_2, , X_1 \cup X_2)$. (We cannot use X_1 or X_2 instead of the union since then we could lose fulfilling states.)

(ii) The algorithm proceeds as follows: For every next-part X occurring in the normalized $T_{\mathsf{AX}\phi}$, new successors $S(X)$ are constructed. (p',X') is a successor of (p,X) in $T^{simp}_{\mathsf{AX}\phi}$ iff $(p',X') \in S(X)$.

In the construction of $S(X)$, we use the following notion: The states (p_1,X_1) and (p_2,X_2) are a *prom-ful-pair* if for some subformula $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$ of ϕ:

- $(p_1,X_1) \in \mathit{prom}(\mathsf{AX}\,\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) \setminus \mathit{ful}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$,
- $(p_2,X_2) \in \mathit{ful}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$, and
- there is a loop for (p_1,X_1) in the normalized tableau consisting of states (p,X) lying in $\mathit{prom}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) \setminus \mathit{ful}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$.

Initially, $S(X)$ is the the set of successors in the normalized tableau of a node with next-part X. For any pair (p_1,X_1) and $(p_2,X_2) \in S(X)$, perform the following step:

If $p_1 \wedge p_2$ is satisfiable and $X_2 \Rightarrow X_1$ holds, and the pair is not a prom-ful-pair, then (p_2,X_2) is replaced by $(p_2 \wedge \neg p_1, X_2)$ in $S(X)$, and analogously if $X_1 \Rightarrow X_2$ holds. Otherwise, $S(X)$ is not changed. Since the normalization of step (i) was applied, the result of the step is uniquely determined.

Note that next-states are not changed in the construction of $S(X)$, so for any state (p,X) in the normalized $T_{\mathsf{AX}\phi}$, $S(X)$ is well-defined. Furthermore, membership in $\mathit{prom}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$ and $\mathit{ful}(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$ is not affected.

(iii) If for some state (p,X), p is not satisfiable or (p,X) has no successors, remove (p,X) and iteratively all states that have no successors. Afterwards, remove all states that are not reachable from some initial state.

(iv) If for two elements (p_1,X_1) and $(p_2,X_2) \in S(X)$, $p_1 \wedge p_2$ is satisfiable, $X_2 \Rightarrow X_1$ holds and it is not a prom-ful-pair any more (i. e. some states of the relevant loop for (p_1,X_1) have been removed in step (iii)), replace (p_2,X_2) by $(p_2 \wedge \neg p_1, X_2)$ in $S(X)$ or remove it in case $p_2 \wedge \neg p_1$ is not satisfiable.

2. A tableau T_ϕ is called *deterministic* if for all states (p_1,X_1) and (p_2,X_2) in $T^{simp}_{\mathsf{AX}\phi}$ that are both successors of the same state, $p_1 \wedge p_2$ is unsatisfiable.

The order on pairs of states used in (ii) of the construction of $T^{simp}_{\mathsf{AX}\phi}$ does not matter in the following sense:

Lemma 6 *(a) Let (p,X) be a state of a tableau. Independent of the chosen order on pairs of successors, $S(X)$ has the following property: $\bigvee_{(q,Y)\in\Phi(\widehat{X})}(q \wedge Y) \Leftrightarrow \bigvee_{(q,Y)\in S(X)}(q \wedge Y)$.*

(b) If T and T' are $T^{simp}_{AX\phi}$ constructed with respect to different orderings on pairs in step (ii), then T is deterministic iff T' is deterministic.

Example

$\mathsf{AG}\,\mathsf{AF}\,p$: $T_{\mathsf{AG}\,\mathsf{AF}\,p}$ has states $(\text{true}, \{\mathsf{AX}\,\mathsf{AG}\,\mathsf{AF}\,p, \mathsf{AX}\,\mathsf{AF}\,p\})$ and $(p, \{\mathsf{AX}\,\mathsf{AG}\,\mathsf{AF}\,p\})$, each one having the two states as successors. The normalized tableau has states $(\neg p, \{\mathsf{AX}\,\mathsf{AG}\,\mathsf{AF}\,p, \mathsf{AX}\,\mathsf{AF}\,p\})$ and $(p, \{\mathsf{AX}\,\mathsf{AG}\,\mathsf{AF}\,p, \mathsf{AX}\,\mathsf{AF}\,p\})$, each having both states as successors. This is already deterministic.

$\mathsf{AF\,AG}\,p$: $T_{\mathsf{AF\,AG}\,p}$ has states $(\text{true}, \{\mathsf{AX\,AF\,AG}\,p\})$ and $(p, \{\mathsf{AX\,AG}\,p\})$, the first having itself and the second as successors, the second having only itself as successor. The normalizing step changes nothing, since $\mathsf{AX\,AF\,AG}\,p$ does not imply $\mathsf{AX\,AG}\,p$. The successors are not changed either: Although $\mathsf{AX\,AG}\,p$ implies $\mathsf{AX\,AF\,AG}\,p$, the states $(\text{true}, \{\mathsf{AX\,AF\,AG}\,p\})$ and $(p, \{\mathsf{AX\,AG}\,p\})$ are a prom-ful-pair. So $T_{\mathsf{AF\,AG}\,p}$ is not deterministic.

We have to show that the operation $(\_)^{simp}$ does not change the meaning of a tableau as expressed by the following lemma. The appendix contains a proof sketch.

Lemma 7 *Let ϕ be an* ACTL *formula. For all Kripke structures $\mathcal{S}$ and states s of $\mathcal{S}$, there is some initial state (p,X) of T_ϕ such that $\mathcal{S},s \preccurlyeq T_\phi,(p,X)$ if and only if there is some initial state (p',X') of T_ϕ^{simp} such that $\mathcal{S},s \preccurlyeq T_\phi^{simp},(p',X')$ holds.*

The following lemma follows easily from the definition of deterministic tableau.

Lemma 8 *If T_ϕ is deterministic, then for any structure $\mathcal{S}$ holds that $\mathcal{S} \subseteq T_\phi^{simp}$ iff $\mathcal{S} \preccurlyeq T_\phi^{simp}$.*

The notion of deterministic tableau characterizes the ACTL formulas expressible in LTL:

Theorem 1 *Let $\phi \in$ ACTL. Then $\phi \Leftrightarrow \phi^d$ iff T_ϕ is deterministic.*

Proof: It suffices to show the direction from left to right, the other direction follows from Lemma 8 and Lemma 5. Assume that T_ϕ is not deterministic. Then $T_{\mathsf{AX}\phi}$ is not deterministic as well. There are two cases to consider.

Case 1: There is a state (p,X) in $T_{\mathsf{AX}\phi}^{simp}$ having two successors (p_1,X_1) and (p_2,X_2) such that

(i) $p_1 \wedge p_2$ is satisfiable,

(ii) $X_1 \not\Rightarrow X_2$ and $X_2 \not\Rightarrow X_1$, and

(iii) (p,X) is reachable from some initial state in $T_{\mathsf{AX}\phi}^{simp}$.

By (ii), there exist Kripke structures $\mathcal{S}_1$ and $\mathcal{S}_2$ and states s_i in $\mathcal{S}_i$ such that $\mathcal{S}_1,s_1 \models \bigwedge_{\mathsf{AX}\psi\in X_1}\psi \wedge \neg(\bigwedge_{\mathsf{AX}\psi\in X_2}\psi)$ and $\mathcal{S}_2,s_2 \models \bigwedge_{\mathsf{AX}\psi\in X_2}\psi \wedge \neg(\bigwedge_{\mathsf{AX}\psi\in X_1}\psi)$. By (iii), we can assume that there is a path π' in $T_{\mathsf{AX}\phi}^{simp}$ from an initial state to (p,X) such that all states except (p,X) have no successors (p_1',X_1') and (p_2',X_2') such that $p_1' \wedge p_2'$ is satisfiable. Let π be a path of length $|\pi'|$ consisting of states such that $\pi \sim \pi'$, which exists since the propositional part of all states in $T_{\mathsf{AX}\phi}^{simp}$ is satisfiable, and let $\tilde{s}$ be a state satisfying $p_1 \wedge p_2$. Let $\mathcal{S}'$ be the Kripke structure formed by concatenating $\mathcal{S}_1$ with the initial state s_1 and $\mathcal{S}_2$ with initial state s_2 to $\pi\tilde{s}$, and let $\pi(0)$ is the initial state of $\mathcal{S}'$.

We show that $\mathcal{S}' \not\preccurlyeq T_{\mathsf{AX}\phi}^{simp}$ but $\mathcal{S}' \subseteq T_{\mathsf{AX}\phi}^{simp}$, which contradicts Lemma 5. By construction, we have that $\mathcal{S}_2,s_2 \models \neg\bigwedge_{\mathsf{AX}\psi\in X_1}\psi$ and $\mathcal{S}_1,s_1 \models \neg\bigwedge_{\mathsf{AX}\psi\in X_2}\psi$, hence $\mathcal{S}',\tilde{s} \not\models p_1 \wedge X_1$ and $\mathcal{S}',\tilde{s} \not\models p_2 \wedge X_2$. By Lemma 3, $\mathcal{S},s \preccurlyeq T_{\mathsf{AX}\phi}^{simp},(p,X)$ implies that $\mathcal{S},s \models p \wedge X$. It follows that there is no simulation $\preccurlyeq$ of $\mathcal{S}'$ by $T_{\mathsf{AX}\phi}^{simp}$ such that $\tilde{s} \preccurlyeq (p_1,X_1)$ and the same holds for (p_2,X_2). As π' is uniquely determined by π, it follows that $\mathcal{S}' \not\preccurlyeq T_{\mathsf{AX}\phi}^{simp}$. On the other hand, $\mathcal{S}' \subseteq T_{\mathsf{AX}\phi}^{simp}$ holds, as by construction $\mathcal{S}_i,s_i \models \bigwedge_{\mathsf{AX}\rho\in X_i}\rho$ for $i=1,2$, and Lemma 4 implies that there are fair paths π_i' in $T_{\bigwedge_{\mathsf{AX}\rho\in X_i}\rho}^{simp}$ such that $\pi_i \sim \pi_i'$.

Case 2: There is a state (p,X) in $T_{\mathsf{AX}\phi}^{simp}$ having two successors (p_1,X_1) and (p_2,X_2) such that

(i) $p_1 \wedge p_2$ is satisfiable,

(ii) $X_2 \Rightarrow X_1$ and $X_1 \not\Rightarrow X_2$,

(iii) (p_1,X_1) is in $prom(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) \setminus ful(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$, while $(p_2,X_2) \in ful(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$, and (p_1,X_1) has a loop with states in $prom(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) \setminus ful(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$ for some subformula $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$ of ϕ,

iv) (p,X) is reachable from some initial state in $T_{\mathsf{AX}\phi}^{simp}$.

We can assume that the loop for (p_1,X_1) consists of states (p,X) such that for all other successors (p',X') of the predecessor of (p,X) in the loop, (p',X') and (p,X) are not a prom-ful-pair for which $p \wedge X \Rightarrow p' \wedge X'$ holds. Otherwise, this pair can be chosen.

Let π' be a path in T_ϕ^{simp} starting at an initial state and leading to (p,X), and let π a path of states such that $\pi \sim \pi'$. We can assume that there is no other path π'' in T_ϕ^{simp} such that $\pi \sim \pi''$. By (iii), there is a loop σ for (p_1,X_1) with states (p,X) in $prom(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2) \setminus ful(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)$. If some state $\sigma(i)$ has another successor (p',X') besides $\sigma(i+1)$, then either $\sigma(i)_1 \wedge \neg X'$ is satisfiable, or $\sigma(i)_0 \wedge p'$ is not satisfiable: Otherwise, $\sigma(i) \Rightarrow p' \wedge X'$ holds and $\sigma(i)_0 \wedge p'$ is satisfiable, and it follows that (p',X') and $\sigma(i+1)$ form a prom-ful-pair, contradicting the assumption on the loop.

We define a Kripke structure $\mathcal{S}'$ as follows. Let s_i be a state satisfying $\sigma(i)$. $\mathcal{S}'$ is defined by attaching $(s_0 s_1 \ldots s_{|\sigma|-1})^\omega$ to π, and by attaching certain Kripke structures to the states s_i, i. e. by letting a state s of a Kripke structure be a successor of s_i: For all $0 < i < |\sigma|$, let for every successor (p,X) of $\sigma(i-1)$ such that $p \wedge \sigma(i)_0$ is satisfiable (and hence as shown above, $\sigma(i)_1 \not\Rightarrow X$), $\mathcal{S}^i_{(p,X)}$ be a Kripke structure with a state $s^i_{(p,X)}$ such that $\mathcal{S}^i_{(p,X)}, s^i_{(p,X)} \models \bigwedge_{\mathsf{AX}\rho\in\sigma(i)_1}\rho \wedge \neg\bigwedge_{\mathsf{AX}\rho\in X}\rho$, and let $s^i_{(p,X)}$ be a successor of s_i. For the first occurrence of s_0, i. e. the successor of $\pi(|\pi|-1)$, let $\mathcal{S}^0_{(p,X)}$ and $s^0_{(p,X)}$ be defined for all successors (p,X) of $\pi'(|\pi'|-1)$ such that $\sigma(0)_0 \wedge p$ is satisfiable, and let $s^0_{(p,X)}$ be a successor of the first occurrence of s_0; condition (i) guarantees that $\mathcal{S}^0_{(p,X)}$ is defined for at least one successor (p,X) of $\pi'(|\pi'|-1)$, namely (p_2,X_2). For the other occurrences of s_0 in the loop, the Kripke structure with respect to $\sigma(|\sigma|-1)$ are attached. The fairness requirements are those of $\mathcal{S}$, and in addition the path $\pi(s_0 s_1 \dots s_{|\sigma|-1})^\omega$ is unfair: This can be easily obtained.

Again $\mathcal{S}' \not\preccurlyeq T^{simp}_{\mathsf{AX}\phi}$: Let $\widehat{\pi}$ be a fair path of $\mathcal{S}^0_{(p_2,X_2)}$ starting at $s^0_{(p_2,X_2)}$. For the fair path $\tilde{\pi} = \pi s_0 \widehat{\pi}$ in $\mathcal{S}'$ there is no fair path $\tilde{\pi}'$ in $T^{simp}_{\mathsf{AX}\phi}$ such that $\tilde{\pi}(n) \preccurlyeq \tilde{\pi}'(n)$ for all n; the only path $\tilde{\pi}'$ such that $\tilde{\pi}(n) \preccurlyeq \tilde{\pi}'(n)_0$ is $\pi'\sigma^\omega$, which is not fair. It is easy to see that $\mathcal{S} \subseteq T^{simp}_{\mathsf{AX}\phi}$ holds. □

The last step now consists in showing that ACTL$^{\text{det}}$ does exactly describe those formulas that have a deterministic tableau.

Lemma 9 *If ϕ is a formula of* ACTL *and T_ϕ is deterministic, then there is a formula ϕ' in* ACTL^{det} *such that $\phi \Leftrightarrow \phi'$.*

Proof: By induction on the structure of ψ, for every subformula ψ of ϕ we define an ACTL$^{\text{det}}$ formula ψ^{det} such that $X[\psi := \psi^{\text{det}}] \Leftrightarrow X$ for all states (p,X) in $T_{\mathsf{AX}\phi}$, where $X[\psi := \psi']$ denotes the result of replacing all occurrences of $\mathsf{AX}\,\psi$ in X by $\mathsf{AX}\,\psi'$. This implies that there is an ACTL$^{\text{det}}$ formula ϕ' such that $\mathsf{AX}\,\phi' \Leftrightarrow \mathsf{AX}\,\phi$, and hence $\phi \Leftrightarrow \phi'$. The cases of predicates, $\psi_1 \wedge \psi_2$ and $\mathsf{AX}\,\psi$ are clear or follow directly from the induction hypothesis. The cases $\psi_1 \vee \psi_2$ and $\mathsf{A}\,\psi_1\,\mathsf{W}\,\psi_2$ can be handled similarly to the case $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$.

Case $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$: Let $\Phi(\psi_1) = \{(s_0,U_0),\dots,(s_l,U_l)\}$ and $\Phi(\psi_2) = \{(t_1,V_1),\dots,(t_m,V_m)\}$. We define $(\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)^{\text{det}} =$

$$\mathsf{A}\Big(\big((s_0 \wedge \neg s_1 \wedge \dots \wedge \neg s_l \wedge U_0^{\text{det}}) \vee \cdots \vee (\neg s_0 \wedge \dots \wedge \neg s_{l-1} \wedge s_l \wedge U_l^{\text{det}})\big) \wedge (\neg t_0 \wedge \dots \wedge \neg t_m)\Big)$$
$$\mathsf{U}\Big(\big((t_0 \wedge \neg t_1 \wedge \dots \wedge \neg t_m \wedge V_0^{\text{det}}) \vee \cdots \vee (\neg t_0 \wedge \dots \wedge \neg t_{m-1} \wedge t_m \wedge V_m^{\text{det}})\big) \wedge (t_0 \vee \dots \vee t_m)\Big),$$

where U^{det} is the result of replacing all elements $\mathsf{AX}\,\rho$ of U by $\mathsf{AX}\,(\rho^{\text{det}})$. This is well-defined because U_i and V_j contain only formulas $\mathsf{AX}\,\rho$ such that ρ is a subformula of $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$. Let (p,X) be a state of $T_{\mathsf{AX}\phi}$ such that X contains $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$. We have to show that $X[\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2 := (\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)^{\text{det}}] \Leftrightarrow X$. Using the fact $T_{\mathsf{AX}\phi}$ is deterministic, it can be shown that the propositional parts of successors of $(p, X[\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2 := (\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)^{\text{det}}])$ are pointwise equivalent to the propositional parts of successors of (p,X). This implies that the tableau for $X[\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2 := (\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)^{\text{det}}]$ is isomorphic to the tableau for X, i. e. the states can be matched bijectively so that propositional parts are equivalent, fairness requirements are not changed and successors match. It follows that $X[\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2 := (\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2)^{\text{det}}] \Leftrightarrow X$. □

The converse is obvious from the definition of ACTL$^{\text{det}}$:

Lemma 10 *If for an* ACTL *formula ϕ there is a formula ϕ' in* ACTL^{det} *such that $\phi \Leftrightarrow \phi'$, then T_ϕ is deterministic.*

Theorem 1, Lemma 9 and Lemma 10 imply the main result:

Theorem 2 *Let ϕ be a* ACTL *formula. Then there exists an* LTL *formula ψ which is equivalent to ϕ iff ϕ can be expressed in* ACTL^{det}.

3.3. Decision procedure for ACTL$^{\text{det}}$

Using the notion of deterministic tableau, we obtain a procedure to decide whether an ACTL formula ψ can be expressed in LTL: Namely, by checking whether T_ψ is deterministic.

Theorem 3 *Deciding whether an* ACTL *formula ψ can be expressed in* LTL *is* PSPACE *complete.*

Proof: The lower bound follows by a reduction from ACTL satisfiability, which can be shown to be PSPACE complete analogously to LTL satisfiability ([21]) by using the tableau construction. Let p be a boolean variable that does not occur in ψ. It is easy to see that ψ is not satisfiability iff $\psi \wedge \mathsf{AF}\,\mathsf{AG}\,p$ is expressible in LTL.

We describe a nondeterministic polynomial space algorithm for deciding whether T_ψ is not deterministic. Assume that ψ is in negation normal form. We represent a state of the normalized tableau in form $P \cup Q \cup X$, where P is a set of propositional subformulas, X is a set of subformulas of form $\mathsf{A}\,\psi_1\,\mathsf{U}\,\psi_2$ or $\mathsf{A}\,\psi_1\,\mathsf{W}\,\psi_2$ of ψ such that $P \cup X$ is a state of $T_{\mathsf{AX}\phi}$, and Q is subset of the propositional subformulas of ψ; hence such a state can be encoded in twice the size of ψ. The formula represented by such a state is $\bigwedge P \wedge \bigwedge_{q\in Q} \neg q \wedge \bigwedge X$; the elements of Q stand for the formulas added during the construction of $T^{sim}_{\mathsf{AX}\psi}$, more precisely for an element of the disjunctive normal form of the (conjunction of the) added predicates.

To check whether T_ψ is not deterministic, the algorithm has to guess a path in $T^{simp}_{\mathsf{AX}\psi}$ from $\{\mathsf{AX}\,\psi\}$ to a state that has two successors $P_1 \cup Q_1 \cup X_1$ and $P_2 \cup Q_2 \cup X_2$ such that $\bigwedge P_1 \wedge \bigwedge_{q\in Q_1} \neg q \wedge \bigwedge P_2 \wedge \bigwedge_{q\in Q_2} \neg q$ is satisfiable. For any pair of successive states $P \cup Q \cup X$ and $P' \cup Q' \cup X'$ along the path (and for the last state with respect to $P_1 \cup Q_1 \cup X_1$ and $P_2 \cup Q_2 \cup X_2$), it has to be checked that the successor relation holds and that $\bigwedge P' \wedge \bigwedge q \in Q' \neg q$ is satisfiable. For the former, it has to be checked that for each other successor $P'' \cup X''$ of $P \cup X$ (successor in T_ψ) for which $\bigwedge X' \Rightarrow \bigwedge X''$ holds (this can be decided in PSPACE) and for which $P'' \cup X''$ and $P' \cup X'$ is not a prom-ful-pair, Q' contains an element of P''. By a similar argument, it follows that there is a PSPACE algorithm to check whether a state $P \cup Q \cup X$ has a loop in $T^{simp}_{\mathsf{AX}\psi}$. By Savitch's Theorem, the claim follows. □

4. Universal-branching restriction of LTL: $\mathrm{LTL}^{\mathrm{det}}$

We define $\mathrm{LTL}^{\mathrm{det}}$ inductively like $\mathrm{ACTL}^{\mathrm{det}}$ except that all path quantifiers are removed.

Definition 5 ($\mathrm{LTL}^{\mathrm{det}}$)
- $p \in Pred(V)$,
- for $\mathrm{LTL}^{\mathrm{det}}$ formulas ϕ_1 and ϕ_2 and a predicate p: $\phi_1 \wedge \phi_2$, $\mathsf{X}\phi_1$, $(p \wedge \phi_1) \vee (\neg p \wedge \phi_2)$, $(p \wedge \phi_1)\,\mathsf{U}\,(\neg p \wedge \phi_2)$, $(p \wedge \phi_1)\,\mathsf{W}\,(\neg p \wedge \phi_2)$.

Analogously to $\mathrm{ACTL}^{\mathrm{det}}$, $\mathrm{LTL}^{\mathrm{det}}$ represents all LTL formulas that are expressible in ACTL:

Corollary 1 *Let ψ be an* LTL *formula. There exists an* ACTL *formula ϕ which is equivalent to ψ iff ψ can be expressed in* LTL^{det}.

A *Büchi automaton* $A = (Q, Q_0, R, L, F)$ over the alphabet $\Sigma = States(V)$ consists of a finite set of states Q, a set $Q_0 \subseteq Q$ of initial states, a transition relation $R \subseteq Q \times Q$, a labelling $L : Q \longrightarrow 2^{\Sigma}$ and a set $F \subseteq Q$ of accepting states. A *run* of A on an ω-word $\zeta = \zeta(0)\zeta(1)\cdots \in \Sigma^{\omega}$ is a sequence $\sigma = \sigma(0)\sigma(1)\ldots$ such that $\sigma(0) \in Q_0$, for all $i \geq 0$, $(\sigma(i), \sigma(i+1)) \in R$ and $\zeta(i) \in L(\sigma(i))$. The run is called *successful* if $Inf(\sigma) \cap F \neq \emptyset$, where $Inf(\sigma) \stackrel{\mathrm{def}}{=} \{q \in Q \mid \sigma(n) = q$ for infinitely many $n\}$. The automaton A *accepts* ζ if there is a successful run of A on ζ.

A Büchi automaton *represents* an LTL formula ψ over V if it accepts exactly the paths ζ over $States(V)$ that satisfy ψ, i. e. all models of ψ.

Theorem 4 ([24, 25]) *An* LTL *formula ψ can be represented by a Büchi automaton the size of which is at most exponential in the size of ψ.*

A *1-weak Büchi automaton* is a weak Büchi automaton such that there is a partial ordering $\leq$ on the state set Q such that every q and q' for which $(q, q') \in R$, we have $q \leq q'$. It is useful to identify the accepting states $q \in F$ such that for all sequences ζ starting in a state s such that $s \in L(q)$ there is a successful run on ζ starting in q. Acc_0 consists of those accepting states, while Acc_1 consists of all other accepting states. So we denote a 1-weak Büchi automaton (Q, Q_0, R, F) alternatively by $(Q, Q_0, R, Acc_0, Acc_1)$.

We can do better for $\mathrm{LTL}^{\mathrm{det}}$ formulas than for general LTL formulas:

Theorem 5 *Let ϕ be an* LTL^{det} *formula. There is a 1-weak Büchi automaton A_{ϕ}^{Neg} representing $\neg\phi$ such that the number of states of A_{ϕ}^{Neg} is linear in the size of ϕ.*

The construction is explained in the appendix.

Example Let us consider the formula $\phi = \mathsf{G}\,(p \vee \neg p \wedge \mathsf{X}\,(p_1 \wedge \neg p_2\,\mathsf{U}\,p_2))$; the corresponding automaton A_{ϕ}^{Neg} (after removal of unsatisfiable states) is shown in Figure 2 on the next page, where $Acc_0 = \{(\neg p_1 \wedge \neg p_2, 1)\}$ and $Acc_1 = \{(p_1 \wedge \neg p_2, 4)\}$.

A formula that can be represented by an automaton of linear size can be model checked in time linear in the formula and the model. Moreover, as explained above, the structure of 1-weak Büchi automata makes them more efficiently checkable than general Büchi automata. Hence the following corollary can be seen as a formalization of the presumption formulated in [12] that LTL formulas expressible in CTL can be checked (at least) as efficiently as CTL formulas even when using the automaton-based approach to model checking:

Corollary 2 *For every* LTL *formula ψ expressible in* ACTL *there is a 1-weak Büchi automaton representing $\neg\psi$ which has size linear in ψ.*

The easier acceptance condition makes 1-weak Büchi automata attractive for verification, and hence a characterization of the LTL formulas the negation of which can be represented in this way is of interest. It turns out that $\mathrm{LTL}^{\mathrm{det}}$ provides such a characterization.

Lemma 11 *If ϕ is an* LTL *formula such that $\neg\phi$ has a recognizing Büchi automaton which is 1-weak, then there is a* ACTL *formula ϕ' that is equivalent to ϕ.*

For a proof sketch, see the appendix. Corollary 2 and Lemma 11 imply:

Theorem 6 *An* LTL *formula ϕ is expressible in* ACTL *iff there is a recognizing Büchi automaton for $\neg\phi$ that is 1-weak.*

5. Concluding Remarks

We have to leave open the problem whether an CTL formula ψ that is expressible in LTL(and hence by [3], $\psi \Leftrightarrow A\psi^d$ holds), is already expressible in ACTL. It this held, it would immediately follow that $\mathrm{ACTL}^{\mathrm{det}}$ characterizes the CTL formulas expressible in LTL and $\mathrm{LTL}^{\mathrm{det}}$ characterizes the LTL formulas that are expressible in CTL.

Acknowledgments

We would like to thank anonymous referees for spotting an error in an earlier draft, Orna Kupferman for pointing us to the complexity theoretic issues, Roderick Bloem for helpful discussions, especially on the topic of 1-weak Büchi automata, and Markus Kaltenbach for valuable comments.

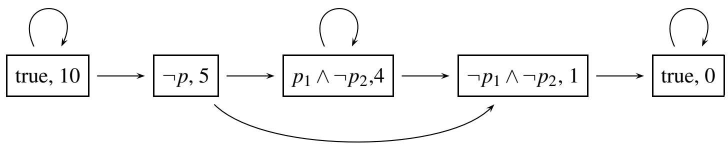

Figure 2. Simplified automaton representing $\neg \mathsf{G}\,(p \vee \neg p \wedge \mathsf{X}\,(p_1 \wedge \neg p_2 \,\mathsf{U}\, p_2))$

References

[1] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. Cheng, S. Edwards, S. Khatri, T. Kukimoto, A. Prado, S. Quadeer, R. Ranjan, S. Sarwary, T. Shiple, G. Swamy, and T. Villa. VIS: a system for verification and synthesis. In *Proc. 8th International. Conference on Computer Aided Verification*, LNCS 1102, pages 428–432, 1996.

[2] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5th Symposium on Logic in Computer Science*, 1990.

[3] E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Proc. Workshop on Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 428–437, 1988.

[4] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. IBM Workshop on Logics of Programs*, LNCS 131, pages 52–71, 1981.

[5] E. M. Clarke, O. Grumberg, and M. C. Browne. Reasoning about networks with many identical finite-state processes. In *Proc. 5th ACM Symposium on Principles of Distributed Computing*, pages 240–248, Calgary, Alberta, August 1986.

[6] E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time strikes back. In *Proc. 20th ACM Symposium on Principles of Programming Languages*, pages 84–96, New Orleans, January 1985.

[7] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *IFIP/WG 6.1*, 1995.

[8] G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.

[9] G. J. Holzmann. An overview of the SPIN model checker. Technical Report BRICS Autumn School on Verification, BRICS Notes Series NS-96-6, BRICS, October 1996.

[10] B. Josko. Modular specification and verification of reactive systems. Habilitationsschrift, Fachbereich Informatik, Carl von Ossietzky Universität Oldenburg, 1993.

[11] O. Kupferman and M. Vardi. Module checking. In *Proc. 8th International Conference on Computer Aided Verification*, LNCS 1102, pages 75–86, 1996.

[12] O. Kupferman and M. Y. Vardi. Relating linear and branching model checking. In *PROCOMET*, 1996.

[13] O. Kupferman and M. Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *Proc. 13th Symposium on Logic in Computer Science*, 1998.

[14] R. P. Kurshan. *Computer-Aided Verification of Coordinating Processes*. Princeton Series in Computer Science, 1994.

[15] D. E. Long. *Model Checking, Abstraction and Compositional Verification*. PhD thesis, CMU, July 1993.

[16] M. Maidl. *System Verification Based on Model Checking*. PhD thesis, Ludwig-Maximilians-Universität München, 2000.

[17] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Specification*. Springer-Verlag, 1992.

[18] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Boston,MA, 1993.

[19] R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd International Joint Conference on Artificial Intelligence*, pages 481–489, September 1971.

[20] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposion on Foundations of Computer Science*, pages 46–57, 1977.

[21] A. P. Sistla and E. M. Clarke. The complexity of propositional temporal logic. In *Proc. 14th ACM Symposium on Theory of Computing*, pages 159–167, 1982.

[22] M. Vardi. Linear vs. branching time: A complexity-theoretic perspective. In *Proc. 13th Symposium Logic in Computer Science*, 1998.

[23] M. Y. Vardi. On the complexity of modular model checking. In *Proc. 10th Symposium on Logic in Computer Science*, pages 101–111, June 1995.

[24] M. Y. Vardi and P. Wolper. Reasoning about infinite computations paths. In *Proceedings of the 24th IEEE symposium on foundation of computer science*, pages 185–194, Tuscan, 1983.

[25] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

6. Appendix

Proof of Lemma 7: Lemma 6 (a) implies that $\preccurlyeq$ is also a simulation of $\mathcal{S}$ by $T^{simp}_{\mathsf{AX}\phi}$. So for a fair path π of states starting at an initial state there is a path π' in $T^{simp}_{\mathsf{AX}\phi}$ starting in an initial state such that $\pi(n) \preccurlyeq \pi'(n)$ for all $n \geq 0$. In order to show that π' is fair, we use the following: If some state in $ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)$ cannot be entered at some k although $\pi(k) \models \psi_2$ holds, i.e. $\pi(k) \not\models p \wedge X$ for all $(p, X) \in ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)$, then by Lemma 6 (a) there is some $(p', X') \in prom(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2) \setminus ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)$ such that $\pi(k) \models p' \wedge X'$. Since $X' \Rightarrow \mathsf{AXA}\,\psi_1 \,\mathsf{U}\, \psi_2$, there is some $k' > k$ such that $\pi(k') \models \psi_2$. The claim follows from this since eventually $(p, X) \in ful(\mathsf{A}\,\psi_1 \,\mathsf{U}\, \psi_2)$ is the second node

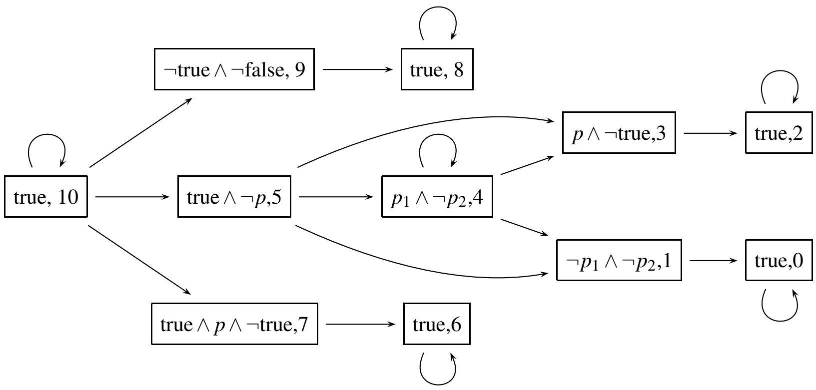

Figure 3. Automaton representing $\neg \mathsf{G}\,(p \vee \neg p \wedge \mathsf{X}\,(p_1 \wedge \neg p_2 \,\mathsf{U}\, p_2))$

of a prom-ful-pair and hence the propositional part of (p,X) was not strengthened by the algorithm. $\square$

Proof of Theorem 5: By induction on ϕ we proof that there is a 1-weak Büchi automaton $A^{Neg}_{\phi} = (Q, Q_0, R, Acc_0, Acc_1)$ of linear size that represents $\neg\phi$. States of A^{Neg}_{ϕ} are of form (p,i) for some state predicate p and $i \in \mathbb{N}$. The labelling set of states of a state (p,i) is the set of states satisfying p. The partial order on states is given by $(p,i) \leq (p',i')$ if $i \geq i'$. We use the integer-component of a state also as numbering and therefore choose a total order.

Case q: We define $Q = \{(\neg q, 1), (\text{true}, 0)\}$, $Q_0 = (\neg q, 1)$, $R = \{((\neg q,1),(\text{true},0)),((\text{true},0),(\text{true},0))\}$, $Acc_0 = \{(\neg q, 1)\}$ and $Acc_1 = \emptyset$.

Case $\mathsf{X}\phi$: Let N be the number of states of T_ϕ and $A^{Neg}_{\phi} = (Q', Q'_0, R', Acc'_0, Acc'_1)$. Then $Q = (\text{true}, N) \cup Q'$, $Q_0 = \{(\text{true}, N)\}$, $R = R' \cup \{((\text{true},N),(q,i)) \mid (q,i) \in Q_0\}$, $Acc_0 = Acc'_0$ and $Acc_1 = Acc'_1$.

Case $\phi_1 \wedge \phi_2$: Pointwise union of $A^{Neg}_{\phi_1}$ and $A^{Neg}_{\phi_2}$.

Case $(p \wedge \phi_1) \vee (\neg p \wedge \phi_2)$: As $(p \wedge \phi_1) \vee (\neg p \wedge \phi_2) \Leftrightarrow (p \vee \phi_2) \wedge (\neg p \vee \phi_1)$, we can use the construction for case $\phi_1 \wedge \phi_2$.

Case $(p \wedge \phi_1) \,\mathsf{U}\, (\neg p \wedge \phi_2)$: Let $A^{Neg}_{\phi_1} = (Q^1, {Q^1}_0, R_1, Acc^1_0, Acc^1_1)$ and $A^{Neg}_{\phi_2} = (Q^2, {Q^2}_0, R_2, Acc^2_0, Acc^2_1)$. Let N_1 be the size of Q_1 and assume that all second components in states and accepting states of $A^{Neg}_{\phi_2}$ are shifted by N_1. Let N be the number of states in Q^1 and Q^2 together.

- $Q = \{(p,N)\} \cup \{(p \wedge q, i) \mid (q,i) \in Q^1_0\} \cup \{(\neg p \wedge q, i) \mid (q,i) \in Q^2_0\} \cup (Q^1 \setminus Q^1_0) \cup (Q^2 \setminus Q^2_0)$,
- $Q_0 = \{(p,N)\} \cup \{(p \wedge q, i) \mid (q,i) \in Q^1_0\} \cup \{(\neg p \wedge q, i) \mid (q,i) \in Q^2_0\}$,
- $R = \{((p,N),(p,N))\} \cup \{((p,N),(p \wedge q, i)) \mid (q,i) \in Q^1_0\} \cup \{((p,N),(\neg p \wedge q, i)) \mid (q,i) \in Q^2_0\} \cup \{((p \wedge q, i), (q',i')) \mid ((q,i),(q',i')) \in R^1\} \cup \{((\neg p \wedge q, i)(q',i')) \mid ((q,i),(q',i')) \in R^2\} \cup R^1 \cup R^2$,
- $Acc_0 = Acc^1_0 \cup Acc^2_0$, $Acc_1 = \{(p,N)\} \cup Acc^1_1 \cup Acc^2_1$.

The claim follows by using the fact that $\neg(p \wedge \phi_1) \,\mathsf{U}\, (\neg p \wedge \phi_2) \Leftrightarrow p \,\mathsf{W}\, ((p \wedge \neg\phi_1) \vee (\neg p \wedge \neg\phi_2))$.

Case $(p \wedge \phi_1) \,\mathsf{W}\, (\neg p \wedge \phi_2)$: Analogous to the case $(p \wedge \phi_1) \,\mathsf{U}\, (\neg p \wedge \phi_2)$, except that Acc_1 is just $Acc^1_1 \cup Acc^2_1$. $\square$

As an example consider Figure 3, which displays the automaton A^{Neg}_{ϕ} for $\phi = \mathsf{G}\,(p \vee \neg p \wedge \mathsf{X}\,(p_1 \wedge \neg p_2 \,\mathsf{U}\, p_2))$; thereby, $Acc_0 = \{(\neg p_1 \wedge \neg p_2, 1), (p \wedge \neg true, 3), (\text{true} \wedge p \wedge \neg\text{true}, 7), (\neg\text{true} \wedge \neg\text{false}, 9)\}$ and $Acc_1 = \{(p_1 \wedge \neg p_2, 4)\}$.

Proof of Lemma 11: Let A be a representing 1-weak Büchi automaton for $\neg\phi$. By induction on the number n of states of A reachable from a state q and different from q, we assign to q an LTL formula $F(q)$ the negation of which is expressible in LTL^{det}, such that for a sequence π of states, $\pi \models F(q)$ iff A with initial state q accepts π. For a state q, let $Pred(q) = \bigvee\{pred(s) \mid s \in L(q)\}$, where $pred(s)$ is the conjunction of all atomic propositions that are satisfied by s.

Case $n = 0$: q has no successors except from itself. If q has no successors, then $F(q)$ is false, and if it is its own successor, then $F(q)$ is $\mathsf{G}\,(Pred(q))$.

Case $n+1$: Let $q_0, \ldots, q_n$ be the successors of q different from q. If q is its own successor and an accepting state, then let $F(q) = Pred(q) \wedge \mathsf{X}\,(Pred(q)) \,\mathsf{W}\, (\bigvee_i F(q_i))$. Otherwise, if q is not accepting, then let $F(q) = Pred(q) \wedge \mathsf{X}\,(Pred(q)) \,\mathsf{U}\, (\bigvee_i F(q_i))$. If q is not its own successor, then let $F(q) = Pred(q) \wedge \mathsf{X}\,(\bigvee_i F(q_i))$.

Let $q_1, \ldots, q_m$ be the initial states of A. We define ϕ' to be $\bigwedge_i \neg F(q_i)$. By construction, $\pi \models \neg\phi'$ iff π is accepted by A, which implies that $\phi' \Leftrightarrow \phi$. $\square$