An Action Computation Tree Logic With *Unless* Operator

Robert Meolic, Tatjana Kapus, and Zmago Brezočnik

University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Smetanova ulica 17, SI-2000 Maribor, Slovenia
{meolic,kapus,brezocnik}@uni-mb.si

Abstract. This paper is about action computation tree logic (ACTL), a propositional branching-time temporal logic very suitable for specifying properties of concurrent systems described with processes. A new variant of ACTL is introduced, which is based on temporal operators *until* and *unless*, whereas all other temporal operators are derived from them. A fixed point characterisation usable for global model checking with the ability of witnesses and counterexamples generation is shown. The relationship of the new ACTL with CTL and the classical ACTL is discussed.

Keywords: CTL, ACTL, Model checking, Unless, Weak until

1 Introduction

Many temporal logics have been developed for expressing properties of CCS-like processes. In 1985, *M. Hennessy* and *R. Milner* introduced a simple modal logic called *process logic* (nowadays known as HML — Hennessy-Milner logic) and proved that it can provide a characterisation of strong equivalence [1]. In 1989, *C. Stirling* proposed a variant of HML, which is adequate with respect to observational equivalence [2]. In 1990, *R. De Nicola* and *F. Vaandrager* defined HMLU, another variant of HML, which includes indexed until operators and is adequate with respect to branching bisimulation equivalence [3]. In the same year, the same authors also introduced action computation tree logic (ACTL), an action-based branching time logic, which is interpreted over transition-labelled structures and is capable of expressing properties about sequences of executed actions [4, 5]. ACTL has all the nice characteristics of CTL [6], a popular temporal logic interpreted over Kripke structures, including the feasibility of efficient model checking. In the definition of ACTL, a distinction was made between internal actions and visible actions. The given semantics includes indexed temporal operators **X** and **U**, which allowed a derivation of others. In 1993, *R. De Nicola et al.* gave some additional notes on ACTL and proposed it as a general framework for verifying properties in process algebrae [7]. They also showed how to exploit a CTL model checker for ACTL model checking.

ACTL is successfully integrated in some verification tools. In [8], *M. Ferrero* and *M. Cusinato* describe an effective BDD-based ACTL model checker *Severo*. The proposed approach slightly differs from the one of De Nicola et al. since invisible actions are treated equally to visible actions. In [9], *R. Mateescu* reports on successful verification of a bounded retransmission protocol by using ACTL and Lotos. Only a fragment of ACTL is used. ACTL was implemented in functional programming language XTL and the verification was performed by checking given properties with the XTL prototype model checker, which is a part of the *CADP toolbox* [10]. In [11], *A. Bouali*, *S. Gnesi*, and *S. Larosa* introduce verification environment *JACK*, which includes the ACTL model checker AMC encoded by *G. Ferro*. AMC is capable of generating counterexamples but unfortunately, it is also reported to be pretty limited in size of verification problems due to an explicit representation of the state space. As *A. Fantechi et al.* report in [12], this work has been later extended with SAM, a symbolic model checker for μ-ACTL [13], which is an extension of ACTL with fixed point operators. The JACK environment was successfully used in a couple of projects, e.g. in verification of a railway signalling system design [14, 15]. There are two other ACTL approaches known to us. In [16], *C. A. Middelburg* describes a first-order version of ACTL* for reasoning about telecommunication services and features described with SDL. In [17], *P. Asirelli* and *S. Gnesi* report about using an ACTL model checker implemented in Prolog in a deductive database management system *Gedblog*.

The model checkers used in projects involving ACTL are of very different types. Severo and AMC are special-purpose ACTL model checkers written from the scratch. In other approaches, ACTL formulae are transformed either into another type of temporal logic or into special programming languages. In the first case, the modal μ-calculus [18, 19] is of considerable importance because of its expressiveness. In the second case, interpreters are used to carry out verification. An example of such an approach is SAM, integrated in the JACK environment, where ACTL formulae are transformed into programming language BSP.

This paper introduces a new, enriched variant of ACTL, which is based on temporal operators *until* (**U**) and *unless* (**W**), whereas all other temporal operators are derived from them. We call it *ACTL with unless operator*. In the literature, temporal operator **W** is also known as *weak until* and has not been introduced in ACTL. All formulae of ACTL can be rendered by the ACTL with *unless* operator whereas the opposite is not true.

The paper is further organised as follows. In Section 2 we give the syntax and semantics of the ACTL with *unless* operator. The proposed semantics covers finite and infinite fullpaths. We also give useful abbreviations and BNF-style syntactic rules for implementation of a comprehensible formulae parser. The section is concluded with a comparison between CTL and ACTL with *unless* operator. Section 3 is on the implementation aspects of model checking using ACTL with *unless* operator. Two fixed point characterisations of operators are given. In the second one, the formulae are transformed to contain only operators **EU**, **EG**, **AW**, and **AF**. This enables an effective generation of witnesses and counterexamples. In discussion in Section 4, we describe the relationship between the ACTL with *unless* operator and the ACTL studied earlier. We give three interesting properties which cannot be expressed with the ACTL proposed in [4], but they can be expressed using the ACTL with *unless* operator. In the conclusion we give some remarks and list directions for further work.

2 Action Computation Tree Logic with *unless* operator

The ACTL with *unless* operator is a propositional branching time temporal logic interpreted over *labelled transition systems* (LTSs). In this section, we introduce and discuss this new variant of action computation tree logic. For simplicity, we refer to it as ACTL.

Definition 1 (Labelled Transition System): A labelled transition system (LTS) is a 4-tuple $L = (\mathcal{S}, \mathcal{A}_\tau, \delta, s_0)$ where:

– $\mathcal{S}$ is a non-empty set of states;
– $\mathcal{A}_\tau$ is a finite, non-empty set of actions containing visible actions and silent action τ not visible to an external observer;
– $\delta \subseteq \mathcal{S} \times \mathcal{A}_\tau \times \mathcal{S}$ is the transition relation;
– s_0 is the initial state. ■

Set of actions $\mathcal{A}_\tau$ will be called an alphabet of the LTS. In comparison to [4], silent action τ is here defined to be in the alphabet. An element $(p, \alpha, q) \in \delta$ is called an α-*transition* or shortly a *transition* from state p to state q. τ-transition is called *internal transition.* If there exists an α-*transition* from a given state, we say that in this state the LTS can *perform* α-transition or that it can *perform* action α. A state is called a *deadlocked state* iff there are no transitions from that state. The set of all deadlocked states in $\mathcal{S}$ will be denoted with $\mathcal{S}_{dead}$. Let $L = (\mathcal{S}, \mathcal{A}_\tau, \delta, s_0)$ be an LTS. A sequence of transitions (p_0, a_1, p_1), $(p_1, a_2, p_2), \ldots$ where $\forall i \geq 0 \,.\, (p_i, a_{i+1}, p_{i+1}) \in \delta$ is called a *path* in L. Moreover, p_i and a_i are called the i-th state and the i-th action on this path, respectively, and the transition ending in the i-th state is called the i-th transition on this path. We will also use notations $st(\pi, i)$ and $act(\pi, i)$ for identification of particular states and transitions on paths (Fig. 1):

– $st(\pi, 0)$ is the first state on the path π,
– $st(\pi, i)$ is a state reached after the i-th transition on the path π $(i \geq 1)$,
– $act(\pi, i)$ is an action executed during the i-th transition on the path π $(i \geq 1)$.

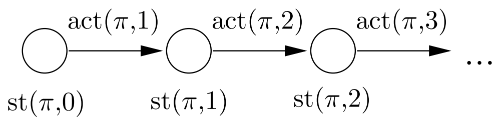

Fig. 1. A path π in an LTS

A sequence of transitions starting and ending in the same state is called a *cycle.* If a path is infinite or ends in a deadlocked state, it is called an *infinite fullpath* or a *finite fullpath*, respectively. The *empty fullpath* is a finite fullpath with one state and no transitions. The number of transitions in finite fullpath π will be denoted with $len(\pi)$.

An ACTL formula may include constants *true* and *false*, *action variables* $\alpha \in \mathcal{A}_\tau$, standard Boolean operators $\neg, \wedge, \vee, \Longrightarrow, \Leftrightarrow$, *path quantifiers* **E** ("there exists a path") and **A** ("for all paths"), and *temporal operators* **U** ("until"), **W** ("unless" or "weak until"), **X** ("for the next transition"), **F** ("for some transition in the future"), and **G** ("for all transitions in the future"). The formal definition of ACTL syntax and semantics includes constant *true*, Boolean operators $\neg$ and $\wedge$, and temporal operators **U** and **W**. Other Boolean and temporal operators are derived from them, whereas constant $\mathit{false} = \neg \mathit{true}$.

Definition 2 (ACTL syntax): Let α be a visible action or silent action τ. Then, χ, φ, and γ are *action formula*, *state formula* (also called *ACTL formula*), and *path formula*, respectively, iff they meet the following syntactic rules:

$$\begin{aligned}
\chi &::= \mathit{true} \mid \alpha \mid \neg\chi \mid \chi \wedge \chi' \\
\varphi &::= \mathit{true} \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \mathbf{E}\,\gamma \mid \mathbf{A}\,\gamma \\
\gamma &::= [\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] \mid [\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi'] \quad \blacksquare
\end{aligned}$$

Definition 3 (ACTL semantics): Let $L = (\mathcal{S}, \mathcal{A}_\tau, \delta, s_0)$ be an LTS and $\alpha \in \mathcal{A}_\tau$. Satisfaction of action formula χ by an action $a \in \mathcal{A}_\tau$ (written $a \models \chi$), state formula φ by a state $p \in \mathcal{S}$ (written $p \models \varphi$), path formula γ by a finite fullpath π (written $\pi \models \gamma$), and path formula γ by an infinite fullpath σ (written $\sigma \models \gamma$) in an LTS L is given inductively by the following semantic rules:

$$\begin{array}{ll}
a \models \mathit{true} & \text{always} \\
a \models \alpha & \text{iff } a = \alpha \\
a \models \neg\chi & \text{iff } a \not\models \chi \\
a \models \chi \wedge \chi' & \text{iff } a \models \chi \wedge a \models \chi' \\
p \models \mathit{true} & \text{always} \\
p \models \neg\varphi & \text{iff } p \not\models \varphi \\
p \models \varphi \wedge \varphi' & \text{iff } p \models \varphi \wedge p \models \varphi' \\
p \models \mathbf{E}\,\gamma & \text{iff there exists a finite fullpath } \pi \text{ where: } p = \mathit{st}(\pi,0) \wedge \pi \models \gamma \\
 & \text{or there exists an infinite fullpath } \sigma \text{ where: } p = \mathit{st}(\sigma,0) \wedge \sigma \models \gamma \\
p \models \mathbf{A}\,\gamma & \text{iff for all finite fullpaths } \pi\text{: } p = \mathit{st}(\pi,0) \Longrightarrow \pi \models \gamma \\
 & \text{and for all infinite fullpaths } \sigma\text{: } p = \mathit{st}(\sigma,0) \Longrightarrow \sigma \models \gamma \\
\pi \models [\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] & \text{iff } \mathit{st}(\pi,0) \models \varphi \ \wedge\ \exists i \in [1, \mathit{len}(\pi)]\,.\,(\\
 & \quad \mathit{act}(\pi,i) \models \chi' \ \wedge\ \mathit{st}(\pi,i) \models \varphi' \ \wedge \\
 & \quad \forall j \in [1, i-1].\,(\mathit{act}(\pi,j) \models \chi \ \wedge\ \mathit{st}(\pi,j) \models \varphi)) \\
\pi \models [\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi'] & \text{iff } \pi \models [\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] \ \text{ or } \ \mathit{st}(\pi,0) \models \varphi \wedge \\
 & \quad \forall i \in [1, \mathit{len}(\pi)]\,.\,(\mathit{act}(\pi,i) \models \chi \wedge \mathit{st}(\pi,i) \models \varphi) \\
\sigma \models [\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] & \text{iff } \mathit{st}(\sigma,0) \models \varphi \ \wedge\ \exists i \geq 1.\,(\\
 & \quad \mathit{act}(\sigma,i) \models \chi' \ \wedge\ \mathit{st}(\sigma,i) \models \varphi' \ \wedge \\
 & \quad \forall j \in [1, i-1].\,(\mathit{act}(\sigma,j) \models \chi \ \wedge\ \mathit{st}(\sigma,j) \models \varphi)) \\
\sigma \models [\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi'] & \text{iff } \sigma \models [\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] \ \text{ or } \ \mathit{st}(\sigma,0) \models \varphi \wedge \\
 & \quad \forall i \geq 1\,.\,(\mathit{act}(\sigma,i) \models \chi \wedge \mathit{st}(\sigma,i) \models \varphi) \quad \blacksquare
\end{array}$$

A state p where ACTL formula φ holds is called *φ-state* and a transition (p, a, q) where action formula χ holds for action a is called *χ-transition.* A χ-transition (p, a, q) where ACTL formula φ holds in state q is called *(χ, φ)-transition.* Further, we will also use the following notation:

– $\mathcal{S}_\varphi \subseteq \mathcal{S}$ denotes the set of all φ-states from $\mathcal{S}$ and $\mathcal{S}_{\neg\varphi} \subseteq \mathcal{S}$ denotes the set of all states from $\mathcal{S}$ which are not φ-states;
– $\delta_\chi \subseteq \delta$ denotes the set of all χ-transitions from δ and $\delta_{\neg\chi} \subseteq \delta$ denotes the set of all transitions from δ which are not χ-transitions;
– $\delta_{(\chi,\varphi)} \subseteq \delta$ denotes the set of all (χ, φ)-transitions from δ and $\delta_{\neg(\chi,\varphi)} \subseteq \delta$ denotes the set of all transitions from δ which are not (χ, φ)-transitions.

In ACTL formulae, each temporal operator is immediately preceded by a path quantifier forming thus an *ACTL operator*. The meaning of ACTL operators **EU**, **AU**, **EW**, and **AW** is given in Definition 3. ACTL operators **EX**, **AX**, **EF**, **AF**, **EG** and **AG** can be derived from them:

$$\begin{array}{ll}
\mathbf{EX}\{\chi\}\,\varphi = \mathbf{E}[\mathit{true}\ \{\mathit{false}\}\,\mathbf{U}\,\{\chi\}\,\varphi] & \mathbf{AX}\{\chi\}\,\varphi = \mathbf{A}[\mathit{true}\ \{\mathit{false}\}\,\mathbf{U}\,\{\chi\}\,\varphi] \\
\mathbf{EF}\{\chi\}\,\varphi = \mathbf{E}[\mathit{true}\ \{\mathit{true}\}\,\mathbf{U}\,\{\chi\}\,\varphi] & \mathbf{AF}\{\chi\}\,\varphi = \mathbf{A}[\mathit{true}\ \{\mathit{true}\}\,\mathbf{U}\,\{\chi\}\,\varphi] \\
\mathbf{EG}\,\varphi\{\chi\} = \mathbf{E}[\varphi\ \{\chi\}\,\mathbf{W}\,\{\mathit{false}\}\,\mathit{false}] & \mathbf{AG}\,\varphi\{\chi\} = \mathbf{A}[\varphi\ \{\chi\}\,\mathbf{W}\,\{\mathit{false}\}\,\mathit{false}]
\end{array}$$

An informal explanation of ACTL formulae may be helpful. Path quantifier **E** requires that the property expressed by the path formula is satisfied for at least one fullpath starting in the given state. On the other hand, path quantifier **A** requires that the property expressed by the path formula is satisfied for all fullpaths starting in the given state. The meaning of the temporal operators can be explained as follows:

– $\mathbf{X}\{\chi\}\,\varphi$ is satisfied on a fullpath iff its first transition is a (χ, φ)-transition.
– $\mathbf{F}\{\chi\}\,\varphi$ is satisfied on a fullpath iff there exists a (χ, φ)-transition on it.
– $\mathbf{G}\,\varphi\{\chi\}$ is satisfied on a fullpath iff the ACTL formula φ holds in its first state and the fullpath is an empty fullpath or all transitions on it are (χ, φ)-transitions.
– $[\varphi\ \{\chi\}\ \mathbf{U}\ \{\chi'\}\ \varphi']$ is satisfied on a fullpath iff the ACTL formula φ holds in its first state and the fullpath consists of a finite and possibly empty sequence of (χ, φ)-transitions followed by a (χ', φ')-transition.
– $[\varphi\ \{\chi\}\ \mathbf{W}\ \{\chi'\}\ \varphi']$ is satisfied on a fullpath iff formula $[\varphi\ \{\chi\}\ \mathbf{U}\ \{\chi'\}\ \varphi']$ is satisfied on it or formula $\mathbf{G}\,\varphi\{\chi\}$ is satisfied on it.

There are some trivial cases for evaluating ACTL formulae. In a deadlocked state, formulae $\mathbf{EG}\,\varphi\{\chi\}$, $\mathbf{AG}\,\varphi\{\chi\}$, $\mathbf{E}\,[\varphi\ \{\chi\}\ \mathbf{W}\ \{\chi'\}\ \varphi']$, and $\mathbf{A}\,[\varphi\ \{\chi\}\ \mathbf{W}\ \{\chi'\}\ \varphi']$ hold only if the state is a φ-state. Formulae $\mathbf{EX}\ \{\chi\}\ \varphi$, $\mathbf{AX}\ \{\chi\}\ \varphi$, $\mathbf{EF}\ \{\chi\}\ \varphi$, $\mathbf{AF}\ \{\chi\}\ \varphi$, $\mathbf{E}\,[\varphi\{\chi\}\mathbf{U}\{\chi'\}\varphi']$, and $\mathbf{A}\,[\varphi\{\chi\}\mathbf{U}\{\chi'\}\varphi']$ do not hold in a deadlocked state. Moreover, if there exists a finite fullpath starting in state p which is an empty fullpath or consists only of (χ, φ)-transitions, then ACTL formulae $\mathbf{E}[\varphi\{\chi\}\mathbf{W}\{\chi'\}\varphi']$ and $\mathbf{EG}\varphi\{\chi\}$ hold in state p whenever p is a φ-state. If there exists a finite fullpath starting in state p which is an empty fullpath or consists only of transitions which are not (χ', φ')-transitions, then ACTL formulae $\mathbf{A}\,[\varphi\ \{\chi\}\ \mathbf{U}\ \{\chi'\}\ \varphi']$ and $\mathbf{AF}\{\chi'\}\ \varphi'$ do not hold in state p.

In many ACTL formulae the constant *true* can be omitted without introducing ambiguities. For example:

$$\mathbf{E}[\mathit{true}\,\{\chi\}\,\mathbf{U}\,\{\chi'\}\,\varphi'] = \mathbf{E}\,[\{\chi\}\,\mathbf{U}\,\{\chi'\}\,\varphi']$$
$$\mathbf{A}[\varphi\,\{\mathit{true}\}\,\mathbf{U}\,\{\mathit{true}\}\,\varphi'] = \mathbf{A}\,[\varphi\,\mathbf{U}\,\varphi']$$

Despite many operators and abbreviations, a comprehensible parser can be easily constructed. Moreover, patterns of ACTL formulae for expressing safety and liveness properties are pretty simple and similar to CTL patterns. BNF-style rules in Fig. 2 represent the complete ACTL syntax as defined in this section, including derived Boolean operators and derived ACTL operators and considering the proposed abbreviations.

```
<ACTL> ::= 'FALSE' | 'TRUE'
<ACTL> ::= 'E' '[' <LEFT> 'U' <RIGHT> ']'
<ACTL> ::= 'A' '[' <LEFT> 'U' <RIGHT> ']'
<ACTL> ::= 'E' '[' <LEFT> 'W' <RIGHT> ']'
<ACTL> ::= 'A' '[' <LEFT> 'W' <RIGHT> ']'
<ACTL> ::= 'EX' <RIGHT> | 'AX' <RIGHT>
<ACTL> ::= 'EF' <RIGHT> | 'AF' <RIGHT>
<ACTL> ::= 'EG' <LEFT> | 'AG' <LEFT>
<ACTL> ::= '(' <ACTL> ')' | 'NOT' <ACTL> |
           <ACTL> 'AND' <ACTL> | <ACTL> 'OR' <ACTL> |
           <ACTL> 'EQV' <ACTL> | <ACTL> 'IMPL' <ACTL>
<LEFT> ::= <ACTL> | <ACTL> '{' <ACTION> '}' | '{' <ACTION> '}'
<RIGHT> ::= <ACTL> | '{' <ACTION> '}' <ACTL> | '{' <ACTION> '}'
<ACTION> ::= 'FALSE' | 'TRUE' | 'TAU' | visible_action
<ACTION> ::= '(' <ACTION> ')' | 'NOT' <ACTION> |
             <ACTION> 'AND' <ACTION> | <ACTION> 'OR' <ACTION> |
             <ACTION> 'IMPL' <ACTION> | <ACTION> 'EQV' <ACTION>
```

Fig. 2. The syntactic rules for an effective ACTL formulae parser

The ACTL with *unless* operator is an action-based version of CTL. Although this paper is not concerned with CTL, let us make some comparison, which will help the reader understand similarities and differences in the meaning of CTL and ACTL formulae. First of all, CTL is defined over Kripke structures, where all fullpaths are infinite. We defined the ACTL with *unless* operator over LTSs where fullpaths can either be infinite or end in a deadlocked state. This is not an essential difference between these two logics because the meaning of CTL formulae over finite fullpaths can be defined, too. Further in this comparison we will address only LTSs without deadlocked states.

The presented variant of ACTL contains *until* operator **U** and *unless* operator **W**. *Unless* operator is not so common in CTL, but it can be introduced as follows:

$$\mathbf{E}[\varphi\,\mathbf{W}\,\varphi'] = \neg\mathbf{A}[\neg\varphi'\,\mathbf{U}\,(\neg\varphi \wedge \neg\varphi')]$$
$$\mathbf{A}[\varphi\,\mathbf{W}\,\varphi'] = \neg\mathbf{E}[\neg\varphi'\,\mathbf{U}\,(\neg\varphi \wedge \neg\varphi')]$$

An *adequate set* of temporal connectives for the given temporal logic is a subset of the logic's temporal connectives that is sufficient to express equivalents for all formulae.

Considering only standard CTL operators, a minimal adequate set for CTL has 3 elements. One of them must be **EX** or **AX**, one of them must be **EG**, **AF**, or **AU**, and finally, one of them must be **EU** [20, 21]. On the other hand, at least 4 elements are neccessary in any adequate set for the ACTL with *unless* operator, because none of its operators **EU**, **AU**, **EW**, and **AW** can be expressed with others three. It seems that in comparison to CTL, the ACTL with *unless* operator needs one extra basic connective and it is not clear, which one. But beware of jumping to conclusions since their semantics are not compatible. The ACTL with *unless* operator uses strict *until* operator [22, 23], from which temporal operator **X** can be derived. This is not possible in CTL with standard reflexive operators. However, strict *until* $\mathbf{U}^>$ and strict *unless* $\mathbf{W}^>$ operators, which ignore the first state on the path, can be introduced into CTL as well:

$$\mathbf{E}[\varphi\,\mathbf{U}^>\,\varphi'] = \mathbf{EX}\,\mathbf{E}[\varphi\,\mathbf{U}\,\varphi'] \qquad \mathbf{A}[\varphi\,\mathbf{U}^>\,\varphi'] = \mathbf{AX}\,\mathbf{A}[\varphi\,\mathbf{U}\,\varphi']$$
$$\mathbf{E}[\varphi\,\mathbf{W}^>\,\varphi'] = \mathbf{EX}\,\mathbf{E}[\varphi\,\mathbf{W}\,\varphi'] \qquad \mathbf{A}[\varphi\,\mathbf{W}^>\,\varphi'] = \mathbf{AX}\,\mathbf{A}[\varphi\,\mathbf{W}\,\varphi']$$

Then, all the standard CTL operators can be expressed either using only $\mathbf{EU}^>$ and $\mathbf{EW}^>$, or using only $\mathbf{EU}^>$ and $\mathbf{AU}^>$, as it is shown below. In the ACTL with *unless* operator, none of this is possible. We may conclude that it needs twice as many different temporal operators as CTL to express all different formulae.

$$\mathbf{EX}\,\varphi = \mathbf{E}[\mathit{false}\,\mathbf{U}^>\,\varphi] \qquad \mathbf{AX}\,\varphi = \mathbf{A}[\mathit{false}\,\mathbf{U}^>\,\varphi]$$
$$\mathbf{EF}\,\varphi = \varphi \vee \mathbf{E}[\mathit{true}\,\mathbf{U}^>\,\varphi] \qquad \mathbf{AF}\,\varphi = \varphi \vee \mathbf{A}[\mathit{true}\,\mathbf{U}^>\,\varphi]$$
$$\mathbf{EG}\,\varphi = \varphi \wedge \mathbf{E}[\varphi\,\mathbf{W}^>\,\mathit{false}] \qquad \mathbf{AG}\,\varphi = \varphi \wedge \mathbf{A}[\varphi\,\mathbf{W}^>\,\mathit{false}]$$
$$\mathbf{E}[\varphi\,\mathbf{U}\,\varphi'] = \varphi' \vee (\varphi \wedge \mathbf{E}[\varphi\,\mathbf{U}^>\,\varphi']) \qquad \mathbf{A}[\varphi\,\mathbf{U}\,\varphi'] = \varphi' \vee (\varphi \wedge \mathbf{A}[\varphi\,\mathbf{U}^>\,\varphi'])$$

3 ACTL Model Checking

ACTL model checking problem is to determine if the given ACTL formula is valid in the given finite-state LTS, i.e. if it holds in its initial state. We will present two approaches for global model checking based on fixed point calculation, similar to the one devised for CTL model checking in [24]. With global model checking the evaluation of a given ACTL formula includes the evaluation of all its subformulae. Such methods also require the complete construction of the LTS before starting the verification.

The first method is derived directly from the definition of ACTL and it is implemented by introducing auxiliary ACTL operators $\mathbf{EX}^+$ and $\mathbf{AX}^+$ [8]:

$$\mathbf{EX}^+[\{\chi\}\,\varphi \vee \{\chi'\}\,\varphi'] = \mathbf{EX}\{\chi\}\,\varphi \;\vee\; \mathbf{EX}\{\chi'\}\,\varphi'$$
$$\mathbf{AX}^+[\{\chi\}\,\varphi \vee \{\chi'\}\,\varphi'] = \neg\mathbf{EX}\{\neg\chi \wedge \neg\chi'\}\,\mathit{true} \;\wedge\; \neg\mathbf{EX}\{\mathit{true}\}\,(\neg\varphi \wedge \neg\varphi') \;\wedge$$
$$\neg\mathbf{EX}\{\neg\chi\}\,\neg\varphi' \;\wedge\; \neg\mathbf{EX}\{\neg\chi'\}\,\neg\varphi$$

Let $L = (\mathcal{S}, \mathcal{A}_\tau, \delta, s_0)$ be a finite-state LTS. ACTL formula $\mathbf{EX}^+[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']$ holds in a state p iff there exists either a (χ, φ)-transition or a (χ', φ')-transition from state p. ACTL formula $\mathbf{AX}^+[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']$ holds in a state p iff each transition from state p is a (χ, φ)-transition or a (χ', φ')-transition. Let $/\varphi/$ denote the subset of states in L where ACTL formula φ holds. The sets of states $/\mathbf{EX}^{+}[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']/$ and $/\mathbf{AX}^{+}[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']/$ can be calculated using expressions 1 and 2, respectively.

$$/\mathbf{EX}^{+}[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']/ = \{q \in \mathcal{S} \mid \exists a \in \mathcal{A}_\tau\ \exists q' \in \mathcal{S}\,.\ ((q,a,q') \in \delta_{(\chi,\varphi)} \vee (q,a,q') \in \delta_{(\chi',\varphi')})\} \tag{1}$$

$$/\mathbf{AX}^{+}[\{\chi\}\,\varphi \vee \{\chi'\}\varphi']/ = \{q \in \mathcal{S} \mid \forall a \in \mathcal{A}_\tau\ \forall q' \in \mathcal{S}\,.\ (q,a,q') \in \delta \Longrightarrow ((q,a,q') \in \delta_{(\chi,\varphi)} \vee (q,a,q') \in \delta_{(\chi',\varphi')})\} \tag{2}$$

Finally, let Z denote a set of states, Φ_Z an ACTL formula which holds in a state s iff $s \in Z$, and $\mathbf{lfp}\,Z.\,f(Z)$ and $\mathbf{gfp}\,Z.\,f(Z)$ the least fixed point and greatest fixed point of function $f(Z)$, respectively. The sets of those states in LTS L where basic ACTL formulae with ACTL operators **EU**, **AU**, **EW**, and **AW** hold can be calculated as stated in formulae 3, 4, 5, and 6.

$$/\mathbf{E}[\varphi\{\chi\}\mathbf{U}\{\chi'\}\varphi']/ = \mathbf{lfp}\,Z.\,(\mathcal{S}_\varphi \cap /\mathbf{EX}^{+}[\{\chi'\}\varphi' \vee \{\chi\}\,\Phi_Z]/) \tag{3}$$

$$/\mathbf{A}[\varphi\{\chi\}\mathbf{U}\{\chi'\}\varphi']/ = \mathbf{lfp}\,Z.\,(\mathcal{S}_\varphi \cap /\mathbf{AX}^{+}[\{\chi'\}\varphi' \vee \{\chi\}\,\Phi_Z]/) \tag{4}$$

$$/\mathbf{E}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi']/ = \mathbf{gfp}\,Z.\,(\mathcal{S}_\varphi \cap (\mathcal{S}_{dead} \cup /\mathbf{EX}^{+}[\{\chi'\}\,\varphi' \vee \{\chi\}\,\Phi_Z]/)) \tag{5}$$

$$/\mathbf{A}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi']/ = \mathbf{gfp}\,Z.\,(\mathcal{S}_\varphi \cap (\mathcal{S}_{dead} \cup /\mathbf{AX}^{+}[\{\chi'\}\,\varphi' \vee \{\chi\}\,\Phi_Z]/)) \tag{6}$$

A path showing that an ACTL formula is valid or not valid in the given finate-state LTS is called a *witness* or a *counterexample*, respectively. For each ACTL formula, either its validity or invalidity can be confirmed with a single path, but not both. The so far proposed characterisation is not directly amenable to the generation of diagnostics, i.e. witnesses and counterexamples. We propose a different approach, where a backward global computation of validity of an ACTL formula can be followed by a forward computation generating the diagnostic.

First of all, the ACTL formula to be evaluated is transformed to contain only ACTL operators **EU**, **EG**, **AW**, and **AF**. This is always possible as ACTL operators **EW** and **AU** can be expressed as follows:

$$\mathbf{E}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi'] = \mathbf{E}[\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] \vee \mathbf{EG}\,\varphi\{\chi\}$$
$$\mathbf{A}[\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi'] = \mathbf{A}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi'] \wedge \mathbf{AF}\,\{\chi'\}\varphi'$$

Then, considering ideas from [25] and [26], ACTL model checking and diagnostics generation is implemented resolving each ACTL operator separately. Valid ACTL formulae $\mathbf{E}[\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi']$ and $\mathbf{EG}\,\varphi\{\chi\}$ have at least one witness and invalid ACTL formulae $\mathbf{A}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi']$ and $\mathbf{AF}\,\{\chi'\}\varphi'$ have at least one counterexample.

Resolving ACTL operator EU: Let $X_{(\chi,\varphi,\varphi',\chi')}$ be the set of states calculated by the fixed point formula 7:

$$X_{(\chi,\varphi,\varphi',\chi')} = \mathbf{lfp}\,Z\,.\,\{q \in \mathcal{S}_\varphi \mid \exists a \in \mathcal{A}_\tau\ \exists q' \in \mathcal{S}\,.\ (q,a,q') \in \delta_{(\chi',\varphi')}\ \vee\ \exists a \in \mathcal{A}_\tau\ \exists q' \in Z.\ (q,a,q') \in \delta_\chi\} \tag{7}$$

Then, $X_{(\chi,\varphi,\varphi',\chi')}$ is the set of all states where ACTL formula $\mathbf{E}[\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi']$ holds. The least fixed point can be effectively determined by calculating the sequence of sets of states $X^0_{(\chi,\varphi,\varphi',\chi')}$, $X^1_{(\chi,\varphi,\varphi',\chi')}$, ... where $X^i_{(\chi,\varphi,\varphi',\chi')} \subseteq \mathcal{S}$ is given as follows:

$$X^0_{(\chi,\varphi,\varphi',\chi')} = \{q \in \mathcal{S}_\varphi \mid \exists a \in \mathcal{A}_\tau\ \exists q' \in \mathcal{S}\,.\,(q,a,q') \in \delta_{(\chi',\varphi')}\} \tag{8}$$

$$\forall i > 0:\ X^i_{(\chi,\varphi,\varphi',\chi')} = X^{i-1}_{(\chi,\varphi,\varphi',\chi')} \cup \{q \in \mathcal{S}_\varphi \mid \exists a \in \mathcal{A}_\tau\ \exists q' \in X^{i-1}_{(\chi,\varphi,\varphi',\chi')}\,.\,(q,a,q') \in \delta_\chi\} \tag{9}$$

Suppose that ACTL formula $\mathbf{E}[\varphi\{\chi\}\,\mathbf{U}\,\{\chi'\}\varphi']$ holds in the initial state s_0 and let $X^n_{(\chi,\varphi,\varphi',\chi')}$ be the first set in the sequence $X^0_{(\chi,\varphi,\varphi',\chi')}$, $X^1_{(\chi,\varphi,\varphi',\chi')}$, ... which contains state s_0. Then, a witness exists which is a finite path beginning in φ-state s_0 and consisting of a sequence of n (χ,φ)-transitions followed by a (χ',φ')-transition (see Fig. 3). To construct this witness, we start in state s_0 and follow a path where $\forall i \in [1,n-1]\,.\,s_i \in X^{n-i}_{(\chi,\varphi,\varphi',\chi')} \setminus X^{n-i-1}_{(\chi,\varphi,\varphi',\chi')}$, $s_n \in X^0_{(\chi,\varphi,\varphi',\chi')}$, $\forall i \in [0,n-1]\,.\,(s_i,a_{i+1},s_{i+1}) \in \delta_{(\chi,\varphi)}$, and $(s_n,a_{n+1},s_{n+1}) \in \delta_{(\chi',\varphi')}$.

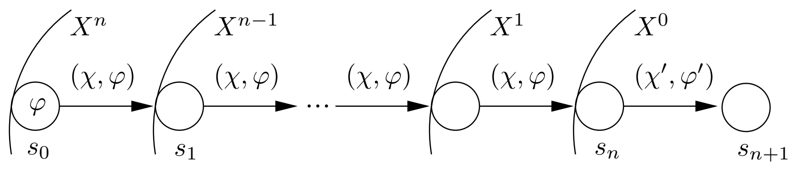

Fig. 3. Resolving ACTL operator **EU**

Resolving ACTL operator EG: Let $\mathcal{C}_{(\chi,\varphi)}$ be the set of states calculated by the fixed point formula 10:

$$\mathcal{C}_{(\chi,\varphi)} = \mathbf{gfp}\,Z.\,(\{q \in \mathcal{S}_\varphi \mid q \in \mathcal{S}_{dead} \vee \exists a \in \mathcal{A}_\tau\ \exists q' \in Z.\,(q,a,q') \in \delta_{(\chi,\varphi)}\}) \tag{10}$$

Then, $\mathcal{C}_{(\chi,\varphi)}$ is the set of all states where ACTL formula $\mathbf{EG}\ \varphi\{\chi\}$ holds.

Suppose that ACTL formula $\mathbf{EG}\ \varphi\{\chi\}$ holds in the initial state s_0. A witness is trivial if s_0 is a deadlocked φ-state. Otherwise, a witness exists which is a finite path beginning in φ-state s_0 consisting only of (χ,φ)-transitions and leading to a deadlocked state or an infinite path beginning in φ-state s_0 and consisting of a sequence of j (χ,φ)-transitions followed by a cycle of n (χ,φ)-transitions (see Fig. 4). To construct this witness, we start in state s_0 and follow a path where $\forall i \in [0,j-1]\,.\,(s_i,a_{i+1},s_{i+1}) \in \delta_{(\chi,\varphi)}$ until a state appears which is on a cycle of (χ,φ)-transitions or the deadlocked state is reached. Checking whether a state $s \in \mathcal{C}_{(\chi,\varphi)}$ is on a cycle of (χ,φ)-transitions can be done by calculating a sequence of sets of states $Y^0_{(\chi,\varphi)}(s)$, $Y^1_{(\chi,\varphi)}(s)$, ... where $Y^i_{(\chi,\varphi)}(s) \subseteq \mathcal{S}$ is given as follows:

$$Y^0_{(\chi,\varphi)}(s) = \{q \in \mathcal{C}_{(\chi,\varphi)} \mid \exists a \in \mathcal{A}_\tau.\,(q,a,s) \in \delta_{(\chi,\varphi)}\} \tag{11}$$

$$\forall i>0:\ Y^{i}_{(\chi,\varphi)}(s) = Y^{i-1}_{(\chi,\varphi)}(s)\ \cup$$
$$\{q\in\mathcal{C}_{(\chi,\varphi)} \mid \exists q'\in Y^{i-1}_{(\chi,\varphi)}(s)\ \exists a\in\mathcal{A}_\tau .\ (q,a,q')\in\delta_{(\chi,\varphi)}\} \qquad (12)$$

The state s is on a cycle of n (χ,φ)-transitions if $s\in Y^{n-1}_{(\chi,\varphi)}(s)\setminus Y^{n-2}_{(\chi,\varphi)}(s)$.

Suppose that s_j is the first state found to be on a cycle of (χ,φ)-transitions. After reaching it, the witness continues with states on the cycle. They can be determined considering the already calculated sets of states $Y^{0}_{(\chi,\varphi)}(s_j)$, $Y^{1}_{(\chi,\varphi)}(s_j)$, ..., $Y^{n-1}_{(\chi,\varphi)}(s_j)$. We start with the state s_j and follow a path where $\forall i\in[j{+}1,\ j{+}n{-}\ 2]\,.\,s_i\in Y^{j+n-i-1}_{(\chi,\varphi)}(s_j)\setminus Y^{j+n-i-2}_{(\chi,\varphi)}(s_j)$, $s_{j+n-1}\in Y^{0}_{(\chi,\varphi)}(s_j)$, $s_{j+n}=s_j$, and $\forall i\in[j,\ j+n-1]\,.\,(s_i,a_{i+1},s_{i+1})\in\delta_{(\chi,\varphi)}$.

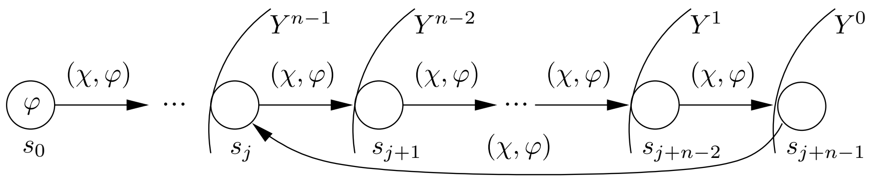

Fig. 4. Resolving ACTL operator **EG**

Resolving ACTL operator AW: Let $X_{\neg(\chi,\varphi,\varphi',\chi')}$ be the set of states calculated by the fixed point formula 13:

$$X_{\neg(\chi,\varphi,\varphi',\chi')} = \mathbf{lfp}\, Z.\ (\ \mathcal{S}_{\neg\varphi}\ \vee\ \{\ q\in\mathcal{S}\ \mid$$
$$\exists a\in\mathcal{A}_\tau\ \exists q'\in\mathcal{S}.\ (q,a,q')\in(\delta_{\neg(\chi,\varphi)}\cap\delta_{\neg(\chi',\varphi')})\ \vee$$
$$\exists a\in\mathcal{A}_\tau\ \exists q'\in Z.\ (q,a,q')\in\delta_{\neg(\chi',\varphi')}\ \}\ ) \qquad (13)$$

Then, $X_{\neg(\chi,\varphi,\varphi',\chi')}$ is the set of all states where ACTL formula $\mathbf{A}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi']$ does not hold. The least fixed point can be effectively determined by calculating the sequence of sets of states $X^{0}_{\neg(\chi,\varphi,\varphi',\chi')}$, $X^{1}_{\neg(\chi,\varphi,\varphi',\chi')}$, ... where $X^{i}_{\neg(\chi,\varphi,\varphi',\chi')}\subseteq\mathcal{S}$ is given as follows:

$$X^{0}_{\neg(\chi,\varphi,\varphi',\chi')} = \{q\in\mathcal{S}\mid q\in\mathcal{S}_{\neg\varphi}\ \vee$$
$$\exists a\in\mathcal{A}_\tau\ \exists q'\in\mathcal{S}\,.\ (q,a,q')\in(\delta_{\neg(\chi,\varphi)}\cap\delta_{\neg(\chi',\varphi')})\} \qquad (14)$$

$$\forall i>0:\ \ X^{i}_{\neg(\chi,\varphi,\varphi',\chi')} = X^{i-1}_{\neg(\chi,\varphi,\varphi',\chi')}\ \cup$$
$$\{q\in\mathcal{S}\mid \exists a\in\mathcal{A}_\tau\ \exists q'\in X^{i-1}_{\neg(\chi,\varphi,\varphi',\chi')}\,.\ (q,a,q')\in\delta_{\neg(\chi',\varphi')}\} \qquad (15)$$

Suppose that ACTL formula $\mathbf{A}[\varphi\{\chi\}\,\mathbf{W}\,\{\chi'\}\varphi']$ does not hold in the initial state s_0 and let $X^{n}_{\neg(\chi,\varphi,\varphi',\chi')}$ be the first set in the sequence $X^{0}_{\neg(\chi,\varphi,\varphi',\chi')}$, $X^{1}_{\neg(\chi,\varphi,\varphi',\chi')}$, ... which contains state s_0. A counterexample is trivial if s_0 is not φ-state. Otherwise, a counterexample exists which is a finite path beginning in state s_0 and consisting of a sequence of n transitions which are (χ, φ)-transitions but not (χ', φ')-transitions followed by a transition which is not a (χ, φ)-transition and also not a (χ', φ')-transition (see Fig. 5). To construct this counterexample, we start in state s_0 and follow a path where $\forall i \in [1, n-1] \,.\, s_i \in X^{n-i}_{\neg(\chi,\varphi,\varphi',\chi')} \setminus X^{n-i-1}_{\neg(\chi,\varphi,\varphi',\chi')}$, $s_n \in X^{0}_{\neg(\chi,\varphi,\varphi',\chi')}$, $\forall i \in [0, n-1] \,.\, (s_i, a_{i+1}, s_{i+1}) \in \delta_{\neg(\chi',\varphi')}$, and $(s_n, a_{n+1}, s_{n+1}) \in \delta_{\neg(\chi,\varphi)} \cap \delta_{\neg(\chi',\varphi')}$.

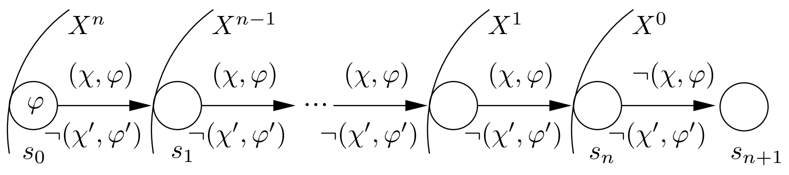

Fig. 5. Resolving ACTL operator **AW**

Resolving ACTL operator AF: Let $\mathcal{C}_{\neg(\chi,\varphi)}$ be the set of states calculated by the fixed point formula 16:

$$\mathcal{C}_{\neg(\chi,\varphi)} = \mathbf{gfp}\, Z.\, (\{q \in \mathcal{S} \mid q \in \mathcal{S}_{dead} \vee \exists a \in \mathcal{A}_\tau\ \exists q' \in Z.\, (q, a, q') \in \delta_{\neg(\chi,\varphi)}\}) \tag{16}$$

Then, $\mathcal{C}_{\neg(\chi,\varphi)}$ is the set of all states where ACTL formula $\mathbf{AF}\{\chi\}\ \varphi$ does not hold.

Suppose that ACTL formula $\mathbf{AF}\{\chi\}\ \varphi$ does not hold in the initial state s_0. A counterexample is trivial if s_0 is a deadlocked state. Otherwise, a counterexample exists which is a finite path beginning in state s_0 consisting only of transitions which are not (χ, φ)-transitions and leading to a deadlocked state or an infinite path beginning in state s_0 and consisting of a sequence of j transitions which are not (χ, φ)-transitions followed by a cycle of n transitions which are not (χ, φ)-transitions (see Fig. 6). To construct this counterexample, we start in state s_0 and follow a path where $\forall i \in [0, j-1] \,.\, (s_i, a_{i+1}, s_{i+1}) \in \delta_{\neg(\chi,\varphi)}$ until a state appears which is on a cycle of transitions which are not (χ, φ)-transitions or the deadlocked state is reached. Checking whether a state $s \in \mathcal{C}_{\neg(\chi,\varphi)}$ is on a cycle of transitions which are not (χ, φ)-transitions can be done by calculating a sequence of sets of states $Y^0_{\neg(\chi,\varphi)}(s)$, $Y^1_{\neg(\chi,\varphi)}(s)$, ... where $Y^i_{\neg(\chi,\varphi)}(s) \subseteq \mathcal{S}$ is given as follows:

$$Y^0_{\neg(\chi,\varphi)}(s) = \{q \in \mathcal{C}_{\neg(\chi,\varphi)} \mid \exists a \in \mathcal{A}_\tau.\, (q, a, s) \in \delta_{\neg(\chi,\varphi)}\} \tag{17}$$

$$\begin{aligned} \forall i > 0: \quad Y^i_{\neg(\chi,\varphi)}(s) = {} & Y^{i-1}_{\neg(\chi,\varphi)}(s) \ \cup \\ & \{q \in \mathcal{C}_{\neg(\chi,\varphi)} \mid \exists q' \in Y^{i-1}_{\neg(\chi,\varphi)}(s)\ \exists a \in \mathcal{A}_\tau.\, (q, a, q') \in \delta_{\neg(\chi,\varphi)}\} \end{aligned} \tag{18}$$

The state s is on a cycle of n transitions which are not (χ, φ)-transitions if $s \in Y^{n-1}_{\neg(\chi,\varphi)}(s) \setminus Y^{n-2}_{\neg(\chi,\varphi)}(s)$.

Suppose that s_j is the first state found to be on a cycle of transitions which are not (χ, φ)-transitions. After reaching it, the counterexample continues with states on the cycle. They can be determined considering already calculated sets of states $Y^0_{\neg(\chi,\varphi)}(s_j)$, $Y^1_{\neg(\chi,\varphi)}(s_j)$, ..., $Y^{n-1}_{\neg(\chi,\varphi)}(s_j)$. We start with the state s_j and follow a path where $\forall i \in [j+1,\, j+n-2]\,.\, s_i \in Y^{j+n-i-1}_{\neg(\chi,\varphi)}(s_j) \setminus Y^{j+n-i-2}_{\neg(\chi,\varphi)}(s_j)$, $s_{j+n-1} \in Y^0_{\neg(\chi,\varphi)}(s_j)$, $s_{j+n} = s_j$, and $\forall i \in [j,\, j+n-1]\,.\,(s_i, a_{i+1}, s_{i+1}) \in \delta_{\neg(\chi,\varphi)}$.

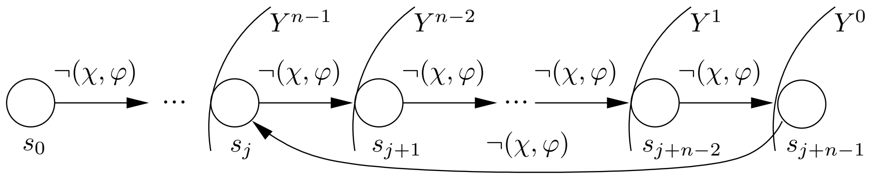

Fig. 6. Resolving ACTL operator **AF**

4 Discussion

There are some important differences between the ACTL syntax and semantics proposed in this paper and those introduced by R. De Nicola and F. Vaandrager in [4] (addressed as the classical ACTL). Our definition contains the additional operator **W**, the path operator $\varphi_\chi \mathbf{U}\, \varphi'$ is absent, and the operators **EX** and **AX** are derived from **U**. Additionally, we allow deadlocked states in the LTS and the usage of silent action τ in action formulae. It is also worth noting that the classical ACTL is given in terms of states and actions (e.g. "from the initial state, such a next state can be reached with a χ-action, where the formula φ holds etc."), whereas the variant of ACTL proposed here is elegantly given in terms of transitions (e.g. "in the initial state there is a (χ,φ)-transition etc.").

Internal transitions are an important concept in process algebrae. The classical ACTL semantics explicitly differs between the silent action and visible actions. In the ACTL with *unless* operator, a special meaning of action τ is ignored. Consequently, its formulae have different meaning as syntactically similar formulae in the classical ACTL. For example, let us compare formulae $\mathbf{E}\,[\varphi\{\alpha\}\;\mathbf{U}\;\{\alpha'\}\varphi']$ and $\exists(\varphi\;{}_\alpha\mathbf{U}_{\alpha'}\;\varphi')$, where α and α' are visible actions. The first formula is valid only if there exists a path consisting of (α, φ)-transitions followed by a (α', φ')-transition. Considering the last formula, the satisfactory path can also include internal transitions. Further we show that the properties stated with the classical ACTL can be equivalently expressed using the ACTL with *unless* operator by explicitly using τ in the formulae.

Let χ and χ' be action formulae which do not contain silent action τ. Then the following equivalences between the classical ACTL (denoted by $\models_{DV90}$) and the ACTL with *unless* operator (denoted by $\models$) exist:

$$\begin{array}{lcl} p \models_{DV90} \exists(\varphi_\chi \mathbf{U}_{\chi'}\, \varphi') & \equiv & p \models \mathbf{E}\,[\varphi\{\chi \vee \tau\}\;\mathbf{U}\;\{\chi'\}\varphi'] \\ p \models_{DV90} \forall(\varphi_\chi \mathbf{U}_{\chi'}\, \varphi') & \equiv & p \models \mathbf{A}\,[\varphi\{\chi \vee \tau\}\;\mathbf{U}\;\{\chi'\}\varphi'] \end{array}$$

$$
\begin{aligned}
p \models_{DV90} \exists(\varphi_\chi \mathbf{U}\, \varphi') &\equiv p \models (\varphi' \;\vee\; \mathbf{E}\,[\varphi\{\chi \vee \tau\}\; \mathbf{U}\; \{\chi \vee \tau\}\varphi'])\\
p \models_{DV90} \forall(\varphi_\chi \mathbf{U}\, \varphi') &\equiv p \models (\varphi' \;\vee\; \mathbf{A}\,[\varphi\{\chi \vee \tau\}\; \mathbf{U}\; \{\chi \vee \tau\}\varphi'])\\
p \models_{DV90} \exists \mathbf{X}_\chi \varphi &\equiv p \models \mathbf{EX}\,\{\chi\}\,\varphi\\
p \models_{DV90} \forall \mathbf{X}_\chi \varphi &\equiv p \models \mathbf{AX}\,\{\chi\}\,\varphi\\
p \models_{DV90} \forall \mathbf{X}_\tau \varphi &\equiv p \models \mathbf{EX}\,\{\tau\}\,\varphi\\
p \models_{DV90} \exists \mathbf{X}_\tau \varphi &\equiv p \models \mathbf{AX}\,\{\tau\}\,\varphi
\end{aligned}
$$

Hence, the ACTL with *unless* operator can render all formulae expressible in the classical ACTL. One should notice that formula $\mathbf{E}[\varphi\,\{\chi\}\;\mathbf{U}\;\varphi']$ is the abbreviation of formula $\mathbf{E}[\varphi\,\{\chi\}\,\mathbf{U}\,\{\mathit{true}\}\,\varphi']$ and has different meaning than the formula $\exists(\varphi_\chi \mathbf{U}\,\varphi')$ in the classical ACTL. The mapping of derived temporal operators and Hennessy-Milner modalities of the classical ACTL is shown by the following equivalences:

$$
\begin{aligned}
p \models_{DV90} \exists \mathbf{F}\,\varphi &\equiv p \models \varphi \vee \mathbf{EF}\,\varphi\\
p \models_{DV90} \forall \mathbf{F}\,\varphi &\equiv p \models \varphi \vee \mathbf{AF}\,\varphi\\
p \models_{DV90} \exists \mathbf{G}\,\varphi &\equiv p \models \mathbf{EG}\,\varphi\\
p \models_{DV90} \forall \mathbf{G}\,\varphi &\equiv p \models \mathbf{AG}\,\varphi\\
p \models_{DV90} \varphi <\chi> \varphi' &\equiv p \models \mathbf{E}\,[\varphi\{\tau\}\;\mathbf{U}\;\{\chi\}\varphi']\\
p \models_{DV90} \varphi <\varepsilon> \varphi' &\equiv p \models (\varphi' \vee \mathbf{E}\,[\varphi\{\tau\}\;\mathbf{U}\;\{\tau\}\varphi'])
\end{aligned}
$$

In the classical ACTL, *unless* operator is not expressible due to the presence of actions in temporal operators. Additionaly, temporal operator **G** derived from *unless* operator is more expressive than the one defined in the classical ACTL. Suppose that a and b are visible actions and that χ and χ' are action formulae without silent action τ. For an LTS without internal transitions, the following interesting properties cannot be expressed with the classical ACTL, but they can be expressed using the ACTL with *unless* operator:

– On all paths, the transitions are χ-transitions at least as long as they are χ'-transitions: $\mathbf{A}[\{\chi\}\;\mathbf{W}\;\{\neg\chi'\}]$.
– There exists a path, such that a can be performed in its states at least as long as action b can be performed in them: $\mathbf{E}[(\mathbf{EX}\{a\})\;\mathbf{W}\;(\neg\mathbf{EX}\{b\})]$.
– There exists a path consisting only of χ-transitions such that in all states on this path action a can be performed: $\mathbf{EG}\;(\mathbf{EX}\{a\})\;\{\chi\}$.

5 Conclusion

ACTL is a propositional branching-time temporal logic similar to CTL, but it describes the occurence of transitions rather than states over time. In this paper, we introduced a new enriched variant of ACTL with *unless* operator, which could be denoted as ACTL-W. It can render all formulae expressible in the classical ACTL. The proposed syntax allows constant *true* to be omitted from the formulae, which contributes a lot to clarity and flexibility. This results in a framework where the properties can be expressed with patterns similar to those used with CTL.

We expressed the formulae of the classical ACTL with the mathematical notation for path quantifiers and action formulae written in indices. On the other hand, the formulae of the ACTL with *unless* operator are given in a parser-friendly notation. The distinct syntax prevents the reader from misunderstanding the meaning of temporal operators, which are not the same in these approaches. On the computer, a different solution must be used. We propose that path quantifiers in the syntax of the ACTL with *unless* operator are doubled, for example **EEX**, **EEF**, **EEG**, **EEU**, and **EEW**, and similar for the ACTL operators with path quantifier **A**. In this way, both types of operators could coexist and the compatibility with existing ACTL model checkers would be preserved.

While the syntax and semantics of the ACTL with *unless* operator are comprehensively explored, there is still a lot of work to do in the field of model checking. We described two fixed point characterisations of ACTL operators and showed how to generate witnesses and counterexamples with a two-pass algorithm. Because the presented approaches are intended for global model checking, the symbolic representations (such as BDDs) are indispensable to avoid the state explosion problem. Another drawback of the proposed methods is that only single paths are produced, which may be a serious limitation in practice. A full diagnostic in the form of a tree-like structure could be generated for a formula containing nested temporal operators. Local model checking, *on-the-fly* verification, and the generation of the full diagnostics are widely investigated topics on the μ-calculus, which is strictly more expressive than ACTL. Those results can serve as a starting point for more powerful ACTL model checkers.

There are some other directions for extending the presented work. Fairness constraints can be introduced. An interface for translation from the natural languages to ACTL formulae would be of great value and would increase the usability of ACTL model checkers signifficantly. An important field of further work is also the adaption of ACTL for process algebrae with data-passing.

References

1. Hennessy, M., Milner, R.: Algebraic Laws for Nondeterminism and Concurrency. Journal of the ACM **32** (1985) 137–161
2. Stirling, C.: An Introduction to Modal and Temporal Logics for CCS. In: Proceedings of the 1989 UK/Japan Workshop on Concurrency: Theory, Language, and Architecture. LNCS 491 (1991) 2–20
3. De Nicola, R., Vaandrager, F.: Three logics for branching bisimulation. Journal of the ACM **42** (1995) 458–487
4. De Nicola, R., Vaandrager, F.W.: Action versus State based Logics for Transition Systems. In: Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science. LNCS 469 (1990) 407–419
5. Fantechi, A., Gnesi, S., Ristori, G.: Model checking for action-based logics. Formal Methods in System Design **4** (1994) 187–203
6. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronisation skeletons using branching time temporal logic. In: Logics of Programs. (1981) 52–71
7. De Nicola, R., Fantechi, A., Gnesi, S., Ristori, G.: An action based framework for verifying logical and behavioural properties of concurrent systems. Computer Networks and ISDN Systems **25** (1993) 761–778

8. Ferrero, M., Cusinato, M.: Severo: A symbolic equivalence verification tool. Tesi di Laurea. Politecnico di Torino, Italy (1994)
9. Mateescu, R.: Formal Description and Analysis of a Bounded Retransmission Protocol. In: Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design. (1996) 98–113
10. Garavel, H., Lang, F., Mateescu, R.: An overview of CADP 2001. EASST Newsletter **4** (2002) 13–24
11. Bouali, A., Gnesi, S., Larosa, S.: The Integration Project for the JACK Environment. In: Bulletin of the EATCS, n.54. (1994) 207–223
12. Fantechi, A., Gnesi, S., Mazzanti, F., Pugliese, R., Tronci, E.: A Symbolic Model Checker for ACTL. In: Proceedings of FM-Trends'98. LNCS 1641 (1998) 228–242
13. Fantechi, A., Gnesi, S.: From ACTL to μ-calculus. In: Proceedings of the ERCIM Workshop on Theory and Practice in Verification. (1992)
14. Anselmi, A., Bernardeschi, C., Fantechi, A., Gnesi, S., Larosa, S., Mongardi, G., Torielli, F.: An Experience in Formal Verification of Safety Properties of a Railway Signalling Control System. In: Proceedings of 14th International Conference on Computer Safety, Reliability and Security (SAFECOMP'95). (1995) 474–488
15. Bernardeschi, C., Fantechi, A., Gnesi, S., Larosa, S., Mongardi, G., Romano, D.: A Formal Verification Environment for Railway Signaling System Design. Formal Methods in System Design **12** (1998) 139–161
16. Middelburg, C.A.: A simple language for expressing properties of telecommunication services and features. Technical Report 94-PU-356, PTT Research (1994)
17. Asirelli, P., Gnesi, S.: Specification and verification of reactive systems using a deductive database. In: Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation. (1998)
18. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science **27** (1983) 333–354
19. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. In: Handbook of Philosophical Logic, 2nd Edition. Volume 4. (2002) 99–217
20. Laroussinie, F.: About the expressive power of CTL combinators. Information Processing Letters **54** (1995) 343–345
21. Martin, A.: Adequate Sets of Temporal Connectives in CTL. In: Proceedings of the 8th International Workshop on Expressiveness in Concurrency (EXPRESS'01). Volume 52 of ENTCS. (2002)
22. Kamp, H.: Tense Logic and the Theory of Linear Order. PhD thesis, University of California, Los Angeles (1968)
23. Emerson, E.A.: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science. Volume B. (1990) 995–1072
24. Burch, J.R., et al.: Symbolic model checking: 10^{20} states and beyond. Information and Computation **98** (1992) 142–170
25. Clarke, E.M., et al.: Efficient Generation of Counterexamples and Witnesses in Symbolic Model Checking. In: Proceedings of 32nd Design Automation Conference (DAC 95). (1995) 427–432
26. Kick, A.: Generation of counterexamples for the my–calculus. Technical Report iratr-1995-37, Universität Karlsruhe, Germany (1995)