A CTL-Based Logic for Program Abstractions

Martin Lange[1] and Markus Latte[2]

[1] Dept. of Computer Science, University of Kassel, Germany
[2] Dept. of Computer Science, Ludwig-Maximilians-University Munich, Germany

Abstract. We define an action-based extension of the branching-time temporal logic CTL which allows path quantifiers to be restricted by formal languages. The main purpose of this logic is its use in abstract interpretation. A reduction from a concrete system to an abstract one may contain spurious traces which can render the verification of the abstract system useless with respect to the concrete one. We pick up the suggestion to verify a modified property on the abstract system instead of the one that the concrete system is supposed to have. The logic introduced here enables a systematic modification of such properties. We present some ways of such a modification which aim at implicitly excluding spurious traces in the verification of abstracted systems.

1 Introduction

Model checking is one of the most successful automatic verification techniques for all kinds of programs: hardware, protocols, reactive software, etc. In model checking, the program to be verified is given as a transition systems representing the operational semantics of a program with states and transitions between the states, and the property specifying correctness of the program is formalised in a temporal logic.

Various temporal logics have been introduced for model checking. The most prominent ones are the linear-time temporal logic LTL [14] and the branching-time temporal logic CTL [7]. These are not only incomparable in terms of their expressive power but also — and partly thus — incomparable in terms of their pragmatics. CTL model checking is easier than LTL model checking (P- vs. PSPACE-complete [5, 15]) whereas LTL satisfiability checking is easier than CTL satisfiability checking (PSPACE- vs. EXPTIME-complete [15, 7]).

These results, in particular the model checking complexities, hold w.r.t. finite models. However, many programs, in particular software, occupy an infinite state space. Clearly, model checking infinite-state programs is undecidable in general but it remains decidable for certain classes of infinite-state programs, e.g. pushdown processes, and weak temporal logics like CTL and LTL. It is still just PSPACE-complete for LTL but EXPTIME-complete for CTL [3, 19].

This does not immediately enable automatic program verification for infinite-state programs because of several reasons. Programs may not fall into these classes, in particular if the cause for infinity is the use of variables over unbounded domains etc., or the relatively high worst-case complexities may not

allow efficient implementations in practice. In such cases, it may be necessary to employ a technique that generally reduces the complexity of the underlying verification problem at the expense of total correctness: *abstraction* [6]. It is also applicable in cases of finite systems where state-space explosion renders the verification problem inefficient in practice.

Abstraction is the process of transforming a transition system $\mathcal{T}$ into a (typically) smaller transition system $\mathcal{T}^{\mathsf{abs}}$ which contains at least some of the information that is present in $\mathcal{T}$. Verification of the smaller system is then easier or even possible. However, the abstraction must be chosen such that the verification of the abstract system allows to make assertions about the underlying property and the original system.

Consider, for instance, one of the most well-known abstraction schemes, called $\exists\exists$-abstraction. There, states of the abstract transition system $\mathcal{T}^{\mathsf{abs}}$ result from collapsing sets of states of the original transition system $\mathcal{T}$, and there is a transition between collapsed states S and T iff there are $s \in S$ and $t \in T$ with a transition from s to t in the original system. It is not hard to see that the runs or paths of $\mathcal{T}^{\mathsf{abs}}$ form a superset of the paths of $\mathcal{T}$. Every path in $\mathcal{T}$ can be found in $\mathcal{T}^{\mathsf{abs}}$ but the latter also contains *spurious traces* which are paths that only arise as artefacts of the abstraction but do not exist in the original system $\mathcal{T}$. Now consider a class of simple properties φ to be checked on $\mathcal{T}$, namely temporal properties which quantify over paths in a universal manner only. It is the case that $\mathcal{T}^{\mathsf{abs}} \models \varphi$ implies $\mathcal{T} \models \varphi$ for such φ but not vice-versa because of the relationship between paths in $\mathcal{T}^{\mathsf{abs}}$ and in $\mathcal{T}$. Thus, if the abstracted system is correct w.r.t. φ then so is the original one. If the abstracted system is faulty then nothing is known about the original one because the reason for the error may be a spurious trace. Still, abstraction can thus enable the verification of systems which cannot be model checked under normal circumstances at the expense of completeness for instance.

It turns out that this is all very well in theory but in practice it happens very often that the abstracted system fails the desired property, i.e. spurious traces interfere with the verification task too much. Now note that there is no reason for considering the *original property* on the abstracted system. This observation has led to suggestions regarding the *weakening* of universally path quantified properties, for instance by considering *fair* traces in the abstract system only [2]. This does work in certain cases. However, there is no general relationship between the abstraction scheme and fairness which would guarantee it to work in many cases. A more precise weakening would relativise the path quantification in the property to be checked on the abstract system to paths occurring in the original system. Note that, if this was possible, then it would not only be restricted to $\exists\exists$-abstractions and universally path quantified properties. The same could be done with existentially path quantified properties and all kinds of abstractions.

For this purpose, we suggest a logic which is based on the commonly used branching-time temporal logic CTL and which allows relativised path quantifiers. While there are logics around with very high expressive power, even temporal ones, these typically extend the modal μ-calculus by incorporating all kinds

of expressive operators. Such logics are more or less useless in the setting of abstraction. We propose to base such a logic on a commonly used logic $\mathcal{L}$ such that the problem of determining $\mathcal{T} \models \varphi$ for some $\varphi \in \mathcal{L}$ can be transformed into the problem of determining $\mathcal{T}^{\mathsf{abs}} \models \varphi^{\mathsf{abs}}$ for an abstracted system $\mathcal{T}^{\mathsf{abs}}$ and a property φ^{abs} which incorporates information that gets lost between $\mathcal{T}$ and $\mathcal{T}^{\mathsf{abs}}$ into φ. Maybe this could even be automised such that to the outside, only $\mathcal{T}$ and φ would be visible which underpins the need for φ to belong to a commonly used specificaton logic and therefore φ^{abs} to be something based upon that.

In Sect. 2 we introduce *Path Relativised Computation Tree Logic* ($\mathrm{CTL}^{\mathrm{rel}}$). It is a simple branching-time temporal logic which is interpreted – like action-based CTL [13] – over transition systems with labeled edges. It extends CTL by allowing path quantifiers to be restricted. The restriction is realised by languages of ω-words aiming at maximal flexibility for the abstraction process. Hence, $\mathrm{CTL}^{\mathrm{rel}}$ is in fact a family of branching-time temporal logics parametrised by a class of formal languages of ω-words. Sect. 3 then exemplifies the possible use of this logic in the framework of abstraction.

$\mathrm{CTL}^{\mathrm{rel}}$ is closely related to *Propositional Dynamic Logic with the Delta operator* (ΔPDL) [17, 12]. This relationship is, for instance, exploited in Sect. 4 where we first analyse the complexity of model checking and satisfiability problems depending on the class of languages used for quantifier relativisation. From the discussion above it should be clear that model checking is an important problem for such a logic in such a framework. Satisfiability checking is, too. Note that a satisfiable formula CTL formula could easily become unsatisfiable when path quantifiers are arbitrarily relativised. A decidable logic then allows such formulas to be automatically checked before they are being used in verification.

In Sect. 5 we consider the use of $\mathrm{CTL}^{\mathrm{rel}}$ in the framework of abstract interpretation. We suggest a generic use of the path quantifier relativisation in $\mathrm{CTL}^{\mathrm{rel}}$ formulas whcih forms the basis for two heuristics that aim at implicitly excluding spurious traces in the verification of abstracted systems. Finally, Sect. 6 contains remarks about future work.

2 CTL with Path Relativisation

Models of CTL with path relativisation are transition systems which — as opposed to ordinary CTL models and like models of action-based CTL — also have labeled edges and need not be total. Let Σ be a finite alphabet and $\mathcal{P}$ be a countably infinite set of atomic propositions. A *transition system* is a tuple $\mathcal{T} = (\mathcal{S}, \rightarrow, \lambda)$ where $\mathcal{S}$ is a set of *states*, $\rightarrow \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$ is the *transition relation*, and $\lambda : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ labels each state with a finite set of propositions that are true in this state. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$.

Let $\mathcal{T} = (\mathcal{S}, \rightarrow, \lambda)$ be a transition system. An $\alpha \colon \mathcal{S} \rightarrow \mathcal{S}$ is an *abstraction function for* $\mathcal{T}$ if it satisfies the following consistency condition for all $s, t \in \mathcal{S}$: if $\alpha(s) = \alpha(t)$ then $\lambda(s) = \lambda(t)$. The function introduces an equivalence relation $\sim_\alpha$ on $\mathcal{S}$ where $s \sim_\alpha s'$ iff $\alpha(s) = \alpha(s')$. Equivalence classes and the quotient set are written as $[\_]_\alpha$ and $\mathcal{S}/\alpha$, respectively. An equivalence class is called an

abstract state. We may omit the index α whenever α is clear from the context. The *abstraction of* $\mathcal{T}$ *w.r.t.* α is the transition system $\mathcal{T}^\alpha = (\mathcal{S}/\alpha, \rightarrow_\alpha, \lambda_\alpha)$ such that for all $a \in \Sigma$:

- $t \xrightarrow{a}_\alpha t'$ iff there are $s \in t$ and $s' \in t'$ such that $s \xrightarrow{a} s'$, and
- $\lambda_\alpha([t]_\alpha) = \lambda(t)$.

Note that the consistency condition above ensures well-definedness of the labeling function λ_α.

In order to simplify technical details, we assume that Σ always contains a special character d and that each transition system has a distinct state end with $s \xrightarrow{\mathsf{d}} \mathsf{end}$ for every s including end itself. Furthermore, end has no other incoming or outgoing transitions than these. This means that transition systems are total in the sense that in any state at least a d-action is possible. However, afterwards nothing else is possible any more. Thus, taking a d-transition somehow indicates being in a deadlock state.

A *path* in $\mathcal{T}$ is an infinite sequence $\pi = s_0, a_0, s_1, a_1, \ldots$, alternating between states and edge labels, s.t. $s_i \xrightarrow{a_i} s_{i+1}$ for all $i \geq 1$. Note that the assumption above ensures that no maximal paths other than infinite ones exist. We write $\Pi_\mathcal{T}(s)$ for the set of all paths through $\mathcal{T}$ that start in s. An *initial path* is a prefix of a path which ends at a state.

A path $\pi = s_0, a_0, s_1, a_1, \ldots$ determines in a unique way the ω-word $a_0 a_1 a_2 \ldots$ over Σ. Abusing notation we will identify a path with its determined word of edge labels and sometimes simply write $\pi \in L$ for a path π and a language L. As usual, Σ^ω denotes the set of all infinite words over Σ.

Formulas of CTL *with path relativisation*, $\mathrm{CTL}^{\mathrm{rel}}$, are built like CTL formulas with the difference that path quantifiers are syntactically indexed by languages of ω-words. We present the logic in positive normal form which simplifies statements about fragments later on.

$$\begin{aligned}\varphi \;::=\; & q \mid \neg q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{EX}_a\varphi \mid \mathtt{AX}_a\varphi \mid \mathtt{E}_L(\varphi \mathtt{U} \varphi) \mid \mathtt{E}_L(\varphi \mathtt{R} \varphi) \mid \\ & \mathtt{A}_L(\varphi \mathtt{U} \varphi) \mid \mathtt{A}_L(\varphi \mathtt{R} \varphi)\end{aligned}$$

where $q \in \mathcal{P}$, $a \in \Sigma$, and $L \subseteq \Sigma^\omega$.

A formula is *purely existential* if does not contain any subformula of the form $\mathtt{AX}_a\psi$, $\mathtt{A}_L(\psi_1 \mathtt{U} \psi_2)$ or $\mathtt{A}_L(\psi_1 \mathtt{R} \psi_2)$. Similarly, it is *purely universal* if it does not contain any subformula of the form $\mathtt{EX}_a\psi$, $\mathtt{E}_L(\psi_1 \mathtt{U} \psi_2)$ or $\mathtt{E}_L(\psi_1 \mathtt{R} \psi_2)$.

Clearly, languages L in the index of a path quantifier are infinite sets of infinite words in general, and the question of syntactic representation of such languages arises. Here we consider automata as such representations, in particular *nondeterministic Büchi automata* (NBA) for ω-regular languages [4], *nondeterministic Büchi visibly-pushdown automata* (NBVPA) [1] for ω-visibly-pushdown languages, and *nondeterministic Büchi pushdown automata* (NBPDA) [16] for ω-context-free languages. By $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{REG}]$, $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{VPL}]$ and $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{CFL}]$ we denote the sets of formulas in which annotated languages are regular, visibly pushdown or context-free, respectively.

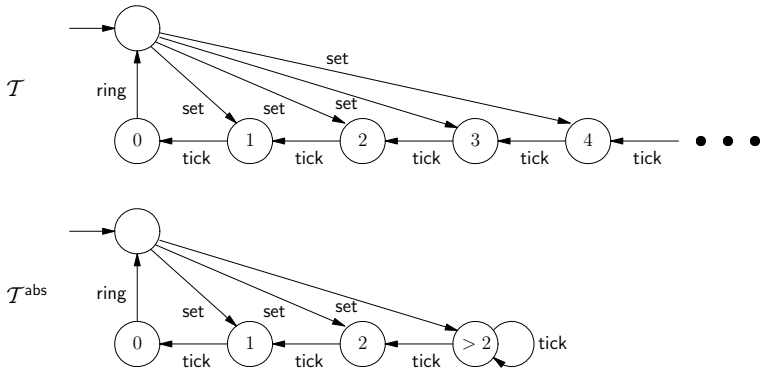

Fig. 1. Transition system of an alarm clock and an abstraction.

We allow more propositional and temporal operators as abbreviations: $\mathtt{tt} := q \vee \neg q$ and $\mathtt{ff} := q \wedge \neg q$ for some $q \in \mathcal{P}$, as well as $Q_L \mathtt{F} \varphi := Q_L(\mathtt{tt} \mathtt{U} \varphi)$, $Q_L \mathtt{G} \varphi := Q_L(\mathtt{ff} \mathtt{R} \varphi)$, and $Q(\varphi \circ \psi) := Q_{\Sigma^\omega}(\varphi \circ \psi)$ for $Q \in \{\mathtt{E}, \mathtt{A}\}$ and $\circ \in \{\mathtt{U}, \mathtt{R}\}$.

The semantics of CTL$^{\text{rel}}$ is given as follows. Let $\mathcal{T} = (\mathcal{S}, \rightarrow, \lambda)$ be a transition system as above. In particular, all paths in it are infinite. For any $s \in \mathcal{S}$ we have:

$$
\begin{array}{rll}
\mathcal{T}, s \models q & \text{iff} & q \in \lambda(s) \\
\mathcal{T}, s \models \neg q & \text{iff} & q \notin \lambda(s) \\
\mathcal{T}, s \models \varphi \vee \psi & \text{iff} & \mathcal{T}, s \models \varphi \text{ or } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \varphi \wedge \psi & \text{iff} & \mathcal{T}, s \models \varphi \text{ and } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \mathtt{EX}_a \varphi & \text{iff} & \text{there is } t \in \mathcal{S} \text{ s.t. } s \xrightarrow{a} t \text{ and } \mathcal{T}, t \models \varphi \\
\mathcal{T}, s \models \mathtt{AX}_a \varphi & \text{iff} & \text{for all } t \in \mathcal{S} : \text{ if } s \xrightarrow{a} t \text{ then } \mathcal{T}, t \models \varphi \\
\mathcal{T}, s \models \mathtt{E}_L(\varphi \mathtt{U} \psi) & \text{iff} & \exists \pi = s_0, a_0, s_1, a_1, \ldots \in \Pi_{\mathcal{T}}(s) \text{ s.t. } a_0 a_1 a_2 \ldots \in L \text{ and} \\
 & & \exists i \in \mathbb{N} \text{ with } \mathcal{T}, s_i \models \psi \text{ and } \forall j < i : \mathcal{T}, s_j \models \varphi \\
\mathcal{T}, s \models \mathtt{E}_L(\varphi \mathtt{R} \psi) & \text{iff} & \exists \pi = s_0, a_0, s_1, a_1, \ldots \in \Pi_{\mathcal{T}}(s) \text{ s.t. } a_0 a_1 a_2 \ldots \in L \text{ and} \\
 & & \forall i \in \mathbb{N} : \mathcal{T}, s_i \models \psi \text{ or } \exists j < i \text{ s.t. } \mathcal{T}, s_j \models \varphi \\
\mathcal{T}, s \models \mathtt{A}_L(\varphi \mathtt{U} \psi) & \text{iff} & \forall \pi = s_0, a_0, s_1, a_1, \ldots \in \Pi_{\mathcal{T}}(s) : \text{ if } a_0 a_1 a_2 \ldots \in L \text{ then} \\
 & & \exists i \in \mathbb{N} \text{ with } \mathcal{T}, s_i \models \psi \text{ and } \forall j < i : \mathcal{T}, s_j \models \varphi \\
\mathcal{T}, s \models \mathtt{A}_L(\varphi \mathtt{R} \psi) & \text{iff} & \forall \pi = s_0, a_0, s_1, a_1, \ldots \in \Pi_{\mathcal{T}}(s) \text{ if } a_0 a_1 a_2 \ldots \in L \text{ then} \\
 & & \forall i \in \mathbb{N} : \mathcal{T}, s_i \models \psi \text{ or } \exists j < i \text{ s.t. } \mathcal{T}, s_j \models \varphi
\end{array}
$$

3 Examples

As a first example, consider an alarm clock $\mathcal{T}$ which can be set to count down an arbitrary number of steps and then ring. Its transition system is depicted in

the top of Fig. 1. Clearly, an alarm clock should ring eventually once it is set to a certain time, therefore, the alarm clock should not have a state from which an infinite tick-path exists. This property is specifiable in action-based CTL as $\mathtt{AG}\neg\mathtt{EG}_{\mathsf{tick}}\mathtt{tt}$.

Now consider an abstraction which identifies all counter values that are greater than 2. This introduces a tick-loop in the state representing all such values. The abstracted system $\mathcal{T}^{\mathsf{abs}}$ is depicted at the bottom of Fig. 1.

It should be clear that $\mathcal{T} \models \mathtt{AG}\neg\mathtt{EG}_{\mathsf{tick}}\mathtt{tt}$ since every sequence of tick-actions must eventually lead to the state with counter value 0 and that has no outgoing tick-action. On the other hand, $\mathcal{T}^{\mathsf{abs}} \not\models \mathtt{AG}\neg\mathtt{EG}_{\mathsf{tick}}\mathtt{tt}$ since state "> 2" is reachable and has an infinite tick-trace. Note that this trace is spurious. It is possible to mend this fault though by introducing fairness and excluding this spurious trace. Take, for instance the fairness predicate $\Phi := \mathtt{GF}\,\mathsf{tick} \Rightarrow \mathtt{GF}\,\mathsf{ring}$, i.e. if infinitely many ticks are being done then also infinitely many rings are being done. Now it is the case that $\mathcal{T}^{\mathsf{abs}} \models_{fair} \mathtt{AG}\neg\mathtt{EG}_{\mathsf{tick}}\mathtt{tt}$ under this fairness predicate, meaning that the CTL path quantifiers in this formula now only range over fair paths, i.e. those that satisfy the fairness predicate Φ. Note that the spurious trace does not, hence, the property holds under this assumption.

While this does work in this particular case, the introduction of a fairness predicate seems rather arbitrary as well as its choice. Furthermore, the chosen fairness predicate almost contradicts the correctness property at hand. Hence, this is almost like only considering that part of the abstracted system which does satisfy the correctness property and then showing that it does indeed. In other words, finding the right fairness predicate may be as hard as showing correctness of the original system.

$\mathrm{CTL}^{\mathrm{rel}}$ offers a more fine-tuned and more systematic way of amending the correctness properties. We will consider another example in which the introduction of fairness is not able to exclude spurious traces that easily. Consider a system containing a buffer into which items can be placed and from which items can be taken. It works such that once something is taken out, it can only be emptied and nothing more can be put into it. The transition system $\mathcal{T}$ is depicted on top in Fig. 2. An abstraction $\mathcal{T}^{\mathsf{abs}}$ which collapses all states containing more than 2 buffer items is depicted below that.

Now consider the correctness property stating that at no point is it possible to execute an out-action followed by an in-action. In action-based CTL it can be written as $\mathtt{AG}\neg\mathtt{EX}_{\mathsf{out}}\mathtt{EX}_{\mathsf{in}}\mathtt{tt}$. Clearly, it is satisfied by the original system $\mathcal{T}$ and not satisfied by the abstraction $\mathcal{T}^{\mathsf{abs}}$ because of the spurious trace through the self-loop in the state representing all large buffer contents. The important observation about this is, though, that no fairness predicate can exclude all the spurious traces which cause the violation of the correctness property. This is simply because fairness is concerned with the infinite occurrence of states / actions, etc. or the absence thereof. The characteristics of the spurious traces in this case, however, is the single occurrence of an in-action after a single out-action. It is therefore sensible to restrict the path quantification to traces of the

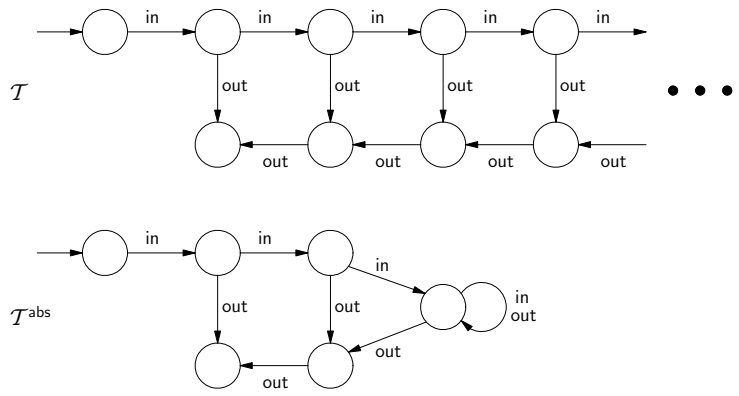

Fig. 2. Transition system of a buffer system and an abstraction.

form $\mathsf{in}^\omega \cup \mathsf{in}^*\mathsf{out}^*\mathsf{d}^\omega$ where action d indicates, as introduced above, a transition into an imaginary deadlock state.

The issue about the right choice of path relativisation still persists, though. As in the first example, the trace predicate $\mathsf{in}^\omega \cup \mathsf{in}^*\mathsf{out}^*\mathsf{d}^\omega$ is somehow found miraculously. However, CTL$^{\text{rel}}$ allows for a more automatic approach. Note that $\mathcal{T}$ is indeed a visibly pushdown system with push-action in and pop-action out. The language of its traces is a visibly pushdown language (ωVPL), characterised by the property that no out-action occurs after in in-action and on any prefix, the number of out-actions is at most as high as the number of in-actions. Let L be that language. Using CTL$^{\text{rel}}$ it is then possible to replace the correctness property above by $\mathtt{AG}\neg\mathtt{E}_{L\cap\Sigma^*\,\mathsf{out}\,\mathsf{in}\,\Sigma^\omega}\mathtt{F}\mathtt{tt}$ for instance and test that on the abstracted system. Note how this restricts path quantification to traces which are present in the original system only. This is of course the essence of excluding spurious traces.

4 Results on CTL$^{\text{rel}}$

We are particularly interested in the complexity of the model checking and satisfiability checking problem for CTL$^{\text{rel}}$ relative to the class of formal languages used for the quantifier restrictions. Upper bounds can easily be obtained by relating it to ΔPDL$^?$ — *Propositional Dynamic Logic with Tests and the Delta operator* — over the corresponding class. We therefore first analyse the relationship between CTL$^{\text{rel}}$ and well-known logics like that one.

4.1 Expressivity

CTL$^{\text{rel}}$ is situated between two cornerstones: CTL [7] and ΔPDL$^?$ [11] i. e. recursive PDL together with delta operators. The former is well-known. The latter

is modal logic over a Kleene algebra of accessibility relations with tests. The delta operator then takes a specification formalisms for infinite words and turns it into an existential quantification over paths labeled with a word in this language. This is of course very similar to the mechanism used in purely existential formulas in $\mathrm{CTL}^{\mathrm{rel}}$. For a comparison to CTL we simply interpret the usual CTL models as $\mathrm{CTL}^{\mathrm{rel}}$ models with a single edge label only.

Theorem 1. $\mathrm{CTL} \leq_{\mathsf{lin}} \mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$ *for* $\Sigma^\omega \in \mathfrak{A}$*, and* $\mathrm{CTL} \lneq_{\mathsf{lin}} \mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$ *for* $\mathfrak{A} \supsetneq \omega\mathrm{REG}$.

Proof. The embedding of CTL is trivial using Σ^ω as a quantifier restriction, and writing $\mathtt{EX}\psi$ as $\mathtt{EX}_a\psi$ for the unique action a the occurs in the underlying models.

For the strictness, consider $\varphi{:=}\mathtt{E}_L\mathtt{G}\mathtt{tt}$ for a language $L \notin \mathrm{REG}$. If this formula had an equivalent CTL-formula then there would be also a Büchi tree automaton which recognizes exactly the models of φ in a certain representation [18]. Hence, there would also be a Büchi word automaton which accepts precisely the words in L which contradicts $L \notin \omega\mathrm{REG}$. □

We remark that $\mathrm{CTL}^{\mathrm{rel}}$ does not seem to be an extension of action-based CTL. For instance, the formula $\mathtt{EF}_a q$ in action-based CTL expresses that there is a path of the form $a^*\Sigma^\omega$ such that q holds in the *first state after the* a^* *prefix*. Clearly, $\mathrm{CTL}^{\mathrm{rel}}$ does not provide a mechanism which can transform information between moments on a path and the inner structure of words in the language restricting those paths.

Theorem 2. $\mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}] \lneq_{\mathsf{lin}} \Delta\mathrm{PDL}^{?}[\mathfrak{A}]$.

Proof. The embedding is proved by induction on the structure of formulas in $\mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$. We detail only the case of $\theta := \mathtt{E}_{\mathcal{A}}\varphi\mathtt{U}\psi$ for an automaton $\mathcal{A}$ with states Q, initial state q_0, and final states F. Let φ' and ψ' be the translations of φ and ψ, respectively. The translation of θ is $\langle\mathcal{B}\rangle\mathtt{tt}$ where $\mathcal{B}$ is an automaton of the same kind as $\mathcal{A}$ with states containing $Q \times \{0,1\}$, initial state $(q_0, 0)$ and final states $F \times \{1\}$. Let $p \xrightarrow[act]{a} q$ denote a transition in $\mathcal{A}$ leading from state p by reading $a \in \Sigma$ to state q while performing operation act on the stack—if applicable. Then $\mathcal{B}$ contains the following three transitions.

$$(q,0) \xrightarrow[nop]{?\varphi'} \xrightarrow[act]{a} (q',0) \qquad (q,0) \xrightarrow[nop]{?\psi'} \xrightarrow[act]{a} (q',1) \qquad (q,1) \xrightarrow[act]{a} (q',1)$$

For the strictness, we consider the property "there is a path on which p holds infinitely often". Obviously, this property is expressible by a delta operator in $\Delta\mathrm{PDL}^{?}[\omega\mathrm{REG}]$. For the sake of contradiction, assume that there is $\mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$-formula φ expressing this property. Hence, φ also characterizes this property over transition systems over a singleton alphabet Σ. For such systems the quantifiers are relativized either to $\emptyset$ or to Σ^ω . Hence, φ can be understood as a CTL-formula. But fairness is not expressible as a CTL-formula [8]. □

4.2 Model Checking

Theorem 3 (Upper and lower bounds). *The model checking problem for* $\mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$ *over a finite transition system is*

- *in PTIME if* $\mathfrak{A}$ *is* ω*-context-free, and*
- *hard for PTIME if* $\Sigma^\omega \in \mathfrak{A}$*.*

Proof. Given a formula $\varphi \in \mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$ and a transition system $\mathcal{T} = (\mathcal{S}, \to, \lambda)$, we compute inductively the set of states in $\mathcal{T}$ which satisfy a subformula of φ. Thereto, we extend λ with those formulas. The cases are similar to that of pure CTL. We detail the case of a formula $\mathtt{E}_L(\varphi \mathtt{U} \psi)$ for $L \in \mathfrak{A}$. For presentation assume that L is given as a Büchi automaton $\mathcal{A} = (Q, q_0, \delta, F)$ where Q is the set of states, $q_0 \in Q$, the transition relation $\delta \in Q \times \Sigma \times Q$, and $F \subseteq Q$ are the final states. We construct for every state $s \in \mathcal{S}$ an automaton $\mathcal{B} := (Q \times \mathcal{S} \times \{0,1\}, (q_0, s, 0), \delta', F')$ recognizing witnessing paths for $\mathtt{E}_L(\varphi \mathtt{U} \psi)$ starting at s. The last component of the state is 1 iff the eventuality is satisfied. So, δ' consists of

$$\begin{array}{rr} ((q,s,0), a, (q',s',0)) & \text{if } \varphi \in \lambda(s) \\ ((q,s,i), a, (q',s',1)) & \text{if } \psi \in \lambda(s) \text{ or } i = 1 \end{array}$$

where each line requires $q' \in \delta(q,a)$ and $s \xrightarrow{a} s'$ for some $a \in \Sigma$. Finally, $F' := F \times \mathcal{S} \times \{1\}$. A similiar construction is available for PDAs. The emptiness check for this ω-PDA can be done in PTIME [3]. Finally, CTL is hard for PTIME. Hence, so is $\mathrm{CTL}^{\mathrm{rel}}[\mathfrak{A}]$. □

4.3 Satisfiability

Theorem 4. *The satisfiability problem for* $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{CFL}]$ *is undecidable.*

Proof. Remember that the universality problem (is $L = \Sigma^*$?) for context-free languages (of finite words) is undecidable. Now let $L \in \mathrm{CFL}$ over some Σ and consider the formula $\varphi_L := \mathtt{E}_{\Sigma^* \mathtt{d}^\omega} \mathtt{F} \neg q \ \wedge \ \mathtt{A}_{L\mathtt{d}^\omega} \mathtt{G} q$.

Remember the assumption about paths in $\mathrm{CTL}^{\mathrm{rel}}$ models being of the form $\Sigma^\omega \cup \Sigma^* \mathtt{d}^\omega$. The first conjunct then says that one of them is of the form $\Sigma^* \mathtt{d}^\omega$ and satisfies $\neg q$ at some point. The second conjunct says that all paths in $L\mathtt{d}^\omega$ satisfy q everywhere. Hence, if $L = \Sigma^*$ then φ_L is clearly unsatisfiable. On the other hand, if there is a $w \in \Sigma^* \setminus L$ then φ_L is for example satisfied in the model which has a single path $w\mathtt{d}^\omega$ such that $\neg q$ holds somewhere on this path. □

Therefore, we consider smaller classes of languages. Those with particular nice algorithmic and algebraic properties are ωREG and ωVPL for instance.

Theorem 5. *The satisfiability problem for* $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{REG}]$ *is EXPTIME-complete, and for* $\mathrm{CTL}^{\mathrm{rel}}[\omega\mathrm{VPL}]$ *is* 2*-EXPTIME-complete.*

Proof. The membership follows from Thm. 2 using that $\Delta\text{PDL}^?[\omega\text{REG}]$ is in EXPTIME [9] and that $\Delta\text{PDL}^?[\omega\text{VPL}]$ is in 2-EXPTIME [12]. Moreover, the logic $\text{CTL}^{\text{rel}}[\omega\text{REG}]$ is EXPTIME-hard as CTL is [10] so. For the hardness of $\text{CTL}^{\text{rel}}[\omega\text{VPL}]$ one can extend the proof [12] of Löding et al. that PDL plus a certain visibly pushdown language is 2-EXPTIME hard. Besides a standard embedding, one $\Delta\text{PDL}^?[\omega\text{VPL}]$-expression needs to be rephrased as it uses an alternating between test operators and labels which is not directly expressible in $\text{CTL}^{\text{rel}}[\omega\text{VPL}]$. An another modification takes account of total transition system. □

5 CTL with Path Relativisation in Abstraction

For the subsequent discussion, we fix a transition system $\mathcal{T} = (\mathcal{S}, \to, \lambda)$ and an abstraction function $\alpha\colon \mathcal{S} \to \mathcal{S}$. Since the transition relation of the abstraction $\mathcal{T}^\alpha$ is defined by existential quantification, a simple induction yields the following statement.

Proposition 6. *For $s \in \mathcal{S}$ we have*

- *If φ is purely existential then $\mathcal{T}, s \models \varphi$ implies $\mathcal{T}^\alpha, [s]_\alpha \models \varphi$.*
- *If φ is purely universal then $\mathcal{T}^\alpha, [s]_\alpha \models \varphi$ implies $\mathcal{T}, s \models \varphi$.*

Hence, it suffices to verify universally quantified formulas on the abstract system, and a positive answer carries over to the concrete system. In general, completeness, i.e. the converse direction, does not hold since the abstraction might admit spurious traces. A negative model checking result on the abstract system therefore need not reflect an error in the concrete system but it could. In order not to stall the design cycle of a system in the verification phase by having negative model checking results too often, one would like to "get as close to completeness as possible". This clearly requires purely existential formulas to be strengthened and purely universal formulas to be weakened. We therefore propose a general mechanism which uses the path quantifier relativisation in CTL^{rel} and realises this strengthening and weaking at the same time. Hence, it is applicable to arbitrary formulas, not just those that are purely existential or universal.

Definition 7. *Let $L \subseteq \Sigma^\omega$ be a language. The* restriction *of a CTL^{rel}-formula φ w.r.t. L is defined as the homomorphic extension over*

$$\big(Q_{L'}(\psi_1 \circ \psi_2)\big) \restriction L := Q_{L' \cap L}\big((\psi_1 \restriction L) \circ (\psi_2 \restriction L)\big) \quad \textit{where } Q \in \{\mathtt{E}, \mathtt{A}\}, \circ \in \{\mathtt{U}, \mathtt{R}\}.$$

Note that ωREG and ωVPL are closed under intersections, hence, $\text{CTL}^{\text{rel}}[\omega\text{REG}]$ and $\text{CTL}^{\text{rel}}[\omega\text{VPL}]$ are closed under restrictions with languages of these respective classes.

A nice property to have would be the following. For all transition systems $\mathcal{T}$, for all abstraction functions α for $\mathcal{T}$ there exists a language $L \neq \emptyset$ such that for all purely existential formulas φ we have: $\mathcal{T}^\alpha, [s]_\alpha \models \varphi \restriction L$ implies $\mathcal{T}, s \models \varphi$. This is not possible however. Assume it was true. Then, it would also apply to

transition systems over a singleton alphabet Σ. But then $L = \Sigma^\omega$ and therefore $\varphi \restriction L \equiv \varphi$. Hence, this property would imply the missing converse directions in Prop. 6 which are easy to refute by counterexample.

In the following we therefore present two heuristics which aim at exluding spurious traces through quantifier relativisation. The first one is rather simple and mainly meant to explain the problems involved in this approach. The second one is more sophisticated and aims at closing down on completeness by making certain requirements on the abstraction.

5.1 A Suffix Heuristic

Suppose $\mathcal{T}$ is a transition system with some initial state s, and $\mathcal{T}^\alpha$ is its abstraction w.r.t. some α. Take the CTL formula $\varphi = \texttt{AGEF}q$ expressing liveness with respect to some proposition q. A natural candidate for the restriction of the path quantifiers in φ would be $L := \Pi_\mathcal{T}(s)$, i.e. the language of all paths in L. Note that not even then does the result of $\mathcal{T}^\alpha, [s]_\alpha \models \varphi \restriction L$ transfer in any way to $\mathcal{T}, s \models \varphi$. The reason for this is the fact that $\Pi_\mathcal{T}(s)$ describes all paths *starting in* s. However, note that the AG-operator intuitively requires the subformula $\texttt{EF}q$ to be interpreted in *arbitrary* states of $\mathcal{T}$, not just s. Hence, $\texttt{EF}q$ should be restricted to paths which start in those states that the formula itself is interpreted in. This would require the subformula to "know" which state it is interpreted in. In other words, the existential quantifier should be restricted to certain suffixes of words in $\Pi_\mathcal{T}(s)$.

This could even mean that $\texttt{E}_L\texttt{F}q$ is interpreted in the starting state of a path which eventually satisfies q but the restriction to L is too rigid and excludes this path. Hence, while one aims at excluding as many spurious traces as possible, one would also exclude good traces. This calls for an overapproximation in order to fix this problem.

Definition 8. *Let $L \subseteq \Sigma^\omega$. The* suffix-closure *of L is*

$$\mathit{Suff}(L) \;:=\; \{w \in \Sigma^\omega \mid \exists v \in \Sigma^* \text{ s.t. } vw \in L\}.$$

The heuristics presented here proposes to reduce the verification task of $\mathcal{T}, s \models \varphi$ on the concrete side to $\mathcal{T}^\alpha, [s]_\alpha \models \varphi \restriction \mathit{Suff}(\Pi_\mathcal{T})$ on the abstract side. Note that the existential quantifier in the definition of $\mathit{Suff}(L)$ realises an overapproximation in the sense that – coming back to the example above – the subformula $\texttt{EF}q$ would of course still be interpreted in an arbitrary but reachable state t of the system, but the quantifier relativisation would restrict the existential path quantifier to suffixes of paths from s. Since some of these pass through t, we have $\Pi_\mathcal{T}(t) \subseteq \mathit{Suff}(\Pi_\mathcal{T}(s))$, and the restricted formula does not exclude good traces. It remains to see how well this does at excluding spurious traces.

Clearly, this heuristic would be worthless if the considered classes of formal languages were not closed under suffixes. However, this is not the case.

Proposition 9. *For all $\mathcal{C} \in \{\omega\mathrm{REG}, \omega\mathrm{VPL}, \omega\mathrm{CFL}\}$ and for all $L \in \mathcal{C}$ we have $\mathit{Suff}(L) \in \mathcal{C}$.*

5.2 A Local Heuristic

Note that the approach suggested in the previous section is global in a sense. Here we propose a local approach which focus on the abstract states, their connections, and the spurious traces that are created within those states

Definition 10. *For a concrete state s in $\mathcal{T}$ we define its* abstraction language *as a subset of $\Sigma^* \cup \Sigma^\omega$ by*

$$\begin{aligned} L_s \;:=\; & \{a_0 \dots a_n \mid \textit{there is an initial path } s_0 a_0 \dots a_{n-1} s_n \textit{ in } \mathcal{T} \textit{ s. th.} \\ & \qquad s_i \in [s] \textit{ for all } 0 \le i < n \textit{ and } s_n \notin [s] \} \\ \cup\; & \{a_0 a_1 \dots \mid \textit{there is a path } s_0 a_0 s_1 a_1 \dots \textit{ in } \mathcal{T} \textit{ s. th.} \\ & \qquad s_i \in [s] \textit{ for all } i \in \mathbb{N} \} \end{aligned}$$

The abstract language *of the abstraction $\mathcal{T}^\alpha$ is $L_\alpha := \left(\bigcup_{s\in\mathcal{S}} L_s\right)^\omega$.*

In other words, the abstract language of a state describes all traces within its class. In particular, fragments of spurious traces are excluded. The language L_α is an over-approximation of the transition system. Indeed, it also admits words in $L_s L_t L_\alpha$ when $[s]$ and $[t]$ are not connected. But, therefore, L_α might have a more condensed description than $\mathcal{T}$ itself.

Prop. 6 can be strengthened by a restriction which is compatible with the induced equivalence classes.

Lemma 11 (Soundness). *For any purely existential formula φ we have that $\mathcal{T}, s \models \varphi$ implies $\mathcal{T}^\alpha, [s]_\alpha \models \varphi \restriction L$, for any $L \supseteq L_s L_\alpha$ and $s \in \mathcal{S}$.*

Proof. Induction on φ. We sketch the case $\varphi = \mathtt{E}_{L'}(\psi_0 \mathtt{U} \psi_1)$ only. Consider a witnessing path $\pi := s_0, a_0, s_1, a_1, \dots$. Then $\pi^\alpha := [s_0]_\alpha, a_0, [s_1]_\alpha, a_1, \dots$ is a path in $\mathcal{T}^\alpha$. By induction hypothesis we have $\mathcal{T}^\alpha, [s]_\alpha \models \mathtt{E}_{L'}((\psi_0 \restriction L)\mathtt{U}(\psi_1 \restriction L))$. A subsequence of π^α might loop in just one equivalence class. This observation gives rise to a factorization along which $\pi^\alpha \in L_s L_\alpha$ can be shown. □

For the converse implication we synchronize traces in the abstract system with those in the concrete one.

Definition 12. *The abstraction $\mathcal{T}^\alpha$ is* syntactically traceable *iff $[s] \xrightarrow{a}_\alpha [s_0]$ and $[s] \xrightarrow{a}_\alpha [s_1]$ imply $s_0 = s_1$ for all $s, s_0, s_1 \in \mathcal{S}$ and $a \in \Sigma$ with $[s_0] \neq [s] \neq [s_1]$.*

Syntactical traceablity is a rather strong and artificial property as it requires that a label determines the targeted state. None of our introductive examples enjoy this property. However in our examples, not the label but the course of the considered trace uniquely specifies the next equivalence class. This observation motivates the following definition.

Definition 13. *The abstraction $\mathcal{T}^\alpha$ is* semantically traceable *iff for all paths $\hat{\pi}$ in $\mathcal{T}^\alpha$ and for all states $s_0, s_1 \in \mathcal{S}$ it holds that $\pi \in L(L_{s_0} L_\alpha \cap L_{s_1} L_\alpha)$ implies $s_0 = s_1$ where $L = \Sigma^* \cap (\bigcup_{s\in\mathcal{S}} L_s)^*$.*

In the alarm clock example, assume that the sequence set tick leads to the class "> 2". Then the label tick either keeps the trace in this class or brings it to the class "1". Hence, the abstracted system is not syntactically traceable. However, it is semantically traceable as the number of ticks before the clock rings determines the next state.

Proposition 14. *If $\mathcal{T}^\alpha$ is syntactically traceable then it is also semantically traceable.*

Theorem 15 (Conditional Completeness). *Let $\mathcal{T}$ be semantically traceable. Suppose that for the formulas ψ_0, $\hat{\psi}_0$, ψ_1 and $\hat{\psi}_1$*

$$\mathcal{T}^\alpha, [s]_\alpha \models \hat{\psi}_i \text{ implies } \mathcal{T}, s \models \psi_i \tag{1}$$

holds for $s \in \mathcal{S}$ and $i \in \{0, 1\}$. Then for $\circ \in \{\mathtt{U}, \mathtt{R}\}$, $s \in \mathcal{S}$ and $L' \subseteq \Sigma^\omega$ we have

$$\mathcal{T}^\alpha, [s]_\alpha \models \mathtt{E}_{L \cap L'}(\hat{\psi}_0 \circ \hat{\psi}_1) \text{ implies } \mathcal{T}, s \models \mathtt{E}_L(\psi_0 \circ \psi_1) \tag{2}$$

where $L' := L_s L_\alpha$.

Proof. Let $\hat{\pi} := \hat{s}_0 a_0 \hat{s}_1 a_1 \ldots$ be a path witnessing the premise of the equation (2). It remains to show that there exists a path π—and not just a sequence of states—in $\mathcal{T}$ which follows $\hat{\pi}$. Given that, the property (1) completes the proof. The path $\hat{\pi}$ can be factorized using L' such that the (finite or infinite) word determined by a factor is in L_t for $t \in \mathcal{S}$. The first factor is in L_s. Along this factorization we construct π as follows. Assume a factor and a path ending at a concrete state such that its abstraction is the first state in the considered factor. For the first factor this path consists of the state s only. Now, the word of the factor is in L_t for some $t \in \mathcal{S}$. Therefore, the definition of this language admits two possibilities. Either, there is an infinite path in $[t]_\alpha$, then we are done. Or there is a finite path π' in $[t]_\alpha$ and a label $a \in \Sigma$ leading π' to a state outside of $[t]_\alpha$. Then we extend π by π' and the said label. By the definition of "semantically traceable", the following node is uniquely determined. □

Although the restriction on the formulas seems to be rather artificial it avoids the suffix problem. However, the consistency condition for abstraction functions ensures that any formula without E and A meets the property (1). Together with Lem. 11, we have completeness of our method with respect to a certain class of formulas.

Corollary 16 (Conditional Faithfulness). *Let $\mathcal{T}$ be semantically traceable and let φ be a formula without* EX, AX, *and nested quantifiers. We have*

$$\mathcal{T}^\alpha, [s]_\alpha \models \varphi \restriction L' \text{ iff } \mathcal{T}, s \models \varphi \tag{3}$$

for $L' := L_s L_\alpha$.

The language L' used by the previous theorem and corollary is subsumed by L_α. Hence, modelchecking $\varphi \upharpoonright L_\alpha$ on the abstract system is almost as good as checking $\varphi \upharpoonright L'$.

Note that Cor. 16 is only formulated for formulas without the next-time operators. The fragment for which Cor. 16 states completeness and thus full elimination of spurious traces is therefore in some sense the stutter-invariant part of $\mathrm{CTL}^{\mathrm{rel}}$ only.

6 Conclusion and Further Work

We have presented a framework for the transformation of correctness properties which should go hand in hand with the transformation of a concrete system into an abstract one. The goal of this transformation is to minimise the significance of spurious traces in the abstract model. We have then suggested two heuristics for certain transformations within this framework.

The work contained herein is obviously not completed. It remains to be seen how these heuristics perform in practice, i.e. how often they can confirm absence of errors in a concrete system (w.r.t. purely universal properties for instance) by confirming that the abstract system is error-free. A dealbreaker may also be the computation of the involved languages which are being factorised into the property. It remains to be seen whether efficient algorithms for these computation problems exist.

On the theoretical side, it is of course possible to consider extensions of $\mathrm{CTL}^{\mathrm{rel}}$. It is not too difficult to see that one could introduce test predicates into the formal languages without losing any of the complexity results. Another obvious extension would be $\mathrm{CTL}^*_{\mathsf{rel}}$, i.e. CTL^* with path relativisation in the same style. This would have a significantly higher complexity in both model checking and satisfiability checking though.

References

1. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. 36th Ann. ACM Symp. on Theory of Computing, STOC'04*, pages 202–211, 2004.
2. D. Bosnacki, N. Ioustinova, and N. Sidorova. Using fairness to make abstractions work. In *Proc. 11th Int. Workshop on Model Checking Software, SPIN'04*, volume 2989 of *LNCS*, pages 198–215. Springer, 2004.
3. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. 8th Int. Conf. on Concurrency Theory, CONCUR'97*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.
4. J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Congress on Logic, Method, and Philosophy of Science*, pages 1–12, Stanford, CA, USA, 1962. Stanford University Press.
5. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logics of Programs: Workshop*, volume 131 of *LNCS*, pages 52–71, Yorktown Heights, New York, 1981. Springer.

6. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.
7. E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.
8. E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
9. E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. In *Foundations of Computer Science, Annual IEEE Symposium on*, pages 328–337, 1988.
10. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
11. C. Löding, C. Lutz, and O. Serre. Propositional dynamic logic with recursive programs. *J. Log. Algebr. Program.*, 73(1-2):51–69, 2007.
12. C. Löding and O. Serre. Propositional dynamic logic with recursive programs. In *Proc. 9th Int. Conf. on Foundations of Software Science and Computation Structures, FOSSACS'06*, volume 3921 of *LNCS*, pages 292–306. Springer, 2006.
13. R. De Nicola and F. Vaandrager. Action versus state based logics for transition systems. In *Proc. Int. Spring School on Theoretical Computer Science: Semantics of Systems of Concurrent Processes*, volume 469 of *LNCS*, pages 407–419. Springer, 1990.
14. A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, 1977. IEEE.
15. A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, 1985.
16. Ludwig Staiger. *Handbook of formal languages, vol. 3: beyond words*, chapter ω-languages, pages 339–387. Springer-Verlag New York, Inc., New York, NY, USA, 1997.
17. R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1/2):121–141, 1982.
18. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.*, 32(2):183–221, 1986.
19. I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.