Model Checking CTL over Restricted Classes of Automatic Structures

Norbert Hundeshagen and Martin Lange

University of Kassel, Germany

Abstract. Interpreting formulas over infinite-state relational structures whose states are words over some alphabet and whose relations are recognised by transducers is known under the term "automatic structures" in the world of predicate logic, or as "regular model checking" in formal verification. Both approaches use synchronised transducers, i.e. finite automata reading tuples of letters in each step. This is a strong transducer model with high expressive power leading to undecidability of model checking for any specification language that can express transitive closure.

We develop conditions on a class of binary word relations which are sufficient for the CTL model checking problem to be computable over the class of automatic structures generated by such relations. As an example, we consider recognisable relations. This is an interesting model from an algebraic point of view but it is also far less expressive than those given by synchronised transducers. As a consequence of the weaker expressive power we obtain that this class satisfies the aforementioned sufficient conditions, hence we obtain a decidability result for CTL model checking over a restricted class of infinite-state automatic structures.

1 Introduction

Model checking is a well-known model-based method for proving correctness of the behaviour of dynamic systems [4]. The earliest approaches were confined to finite-state systems [15], limited by the rather obvious undecidability of checking even the simplest temporal properties – namely reachability – on arbitrary infinite-state spaces. The ability to also model check infinite-state systems is indispensable for the verification of software systems, though. Much effort has therefore gone into the design and study of model checking procedures for infinite-state systems, mainly focussing on particular classes of finitely representable infinite-state systems like pushdown systems [10, 34], Petri nets [28], process algebraic descriptions of infinite-state systems [25, 20], recursion schemes [26], etc.

A rich formalism that gives rise to particular infinite-state systems is known as *automatic structures* [6]. The name is derived from the fact that (finite-state) automata play a major role in the construction of such systems: their states are represented as finite words, and the relations in these structures are recognised by synchronous transducers. Standard automata-theoretic constructions can then be used to show that the model checking problem for First-Order Logic (FO) is decidable over such structures [7]. It is also not difficult to see that model checking for Transitive Closure Logic already – the extension of FO with an operator to express inclusion in the transitive closure of some

binary relation – becomes undecidable as the configuration graph of a Turing Machine can be modelled as an automatic structure.

The richness and flexibility of this framework makes it interesting for verification purposes, despite the fact that even the simplest specification languages for typical correctness properties in verification incorporate transitive closures in some form or other [14]. This has led to the study of *regular model checking* [9], a term describing the framework of verifying labelled transition systems (i.e. relational structures with unary and binary relations only) represented as automatic structures. Interestingly, the use of such structures in the rather difficult domain of verification of temporal properties has started a while before the positive and elegant results on FO model checking were discovered [22, 36].

Research on regular model checking has seen a great amount of effort spent on the computation of transitive closures [12, 31] using various techniques that circumvent undecidability issues, for instance by giving up completeness or precision, like fixpoint acceleration [21, 1], widening [32], abstraction [8], inference [18], etc.

One can argue that the restriction to the computation of transitive closures still facilitates "doing model checking", at least for relatively simple temporal properties like safety or liveness. The approximative nature of procedures like the ones cited above usually prohibits the study of combinations of such properties, as safety verification typically requires over-approximations whereas liveness verification needs under-approximations.

In this paper we want to study the possibility to do model checking for a richer class of temporal properties than just safety or liveness. The simple branching-time temporal logic CTL [11] provides a framework for the specification of combinations of such properties. Our object of interest is therefore the model checking problem for CTL over automatic structures. As stated above, this problem is clearly undecidable, and the multitude of work that has gone into studying the subproblem of verifying liveness or safety properties shows that one cannot expect to find many positive results for regular CTL model checking unless one gives up completeness, precision, or expressive power. We aim to retain completeness and precision and study the case where expressive power is limited on the side of the automatic structures rather than the temporal specification language. We consider a particular case of automatic structures for which the accessibility relation is *recognisable*. The concept of recognisability, defined via morphisms onto a finite monoid, is central in the field of algebraic automata theory. An overview over the classes of relations in focus and the notion of recognisability and synchronisation can be found in [5, 29].

The class of recognisable relations is a proper subclass of the synchronous ones. Hence, the class of automatic structures defined over them is significantly smaller than the class of automatic structures over synchronous transducers. It remains to be seen whether this class includes families of structures that are interesting for software verification purposes for instance. On the other hand, a consequence of this loss in expressive power is – as we show here – that CTL model checking, i.e. including the verification of simple safety or liveness properties, as well as combinations thereof, is decidable over this class of infinite-state systems.

The paper is organised as follows. In Sect. 2 we recall CTL and transition systems as its standard model of interpretation. CTL model checking over automatic structures defined by recognisable relations is not meant to be the ultimate goal in infinite-state verification; instead we want to provide the basis for the study of temporal logic model checking over restricted classes of automatic structures here. We therefore present a generic description of automatic structures as transition systems, parametrised by the machinery used to define its transition relation; recognisable relations and their corresponding automaton model are one example of such machinery that falls into this framework, and it is the one studied in further detail here.

Sect. 3 recalls the generic bottom-up global CTL model checking algorithm, and it then develops necessary criteria on the underlying structures for this algorithm to be terminating and correct. In Sect. 4 we then consider the aforementioned recognisable relations, resp. the automatic structures generated by them and show that they satisfy the necessary conditions laid out before. Hence, we get decidability of CTL model checking over this class of automatic structures. Finally, Sect. 5 concludes with remarks on further work in this area.

2 Preliminaries

2.1 Labelled Transition Systems

Let $\mathcal{P} = \{p, q, \ldots\}$ be a set of proposition symbols. A *labelled transition system* (LTS) is a $\mathcal{T} = (\mathcal{S}, \to, \ell)$ where $\mathcal{S}$ is a (possibly infinite) set of states, $\to \subseteq \mathcal{S} \times \mathcal{S}$ is the transition relation which is always assumed to be total, i.e. for every $s \in \mathcal{S}$ there is a $t \in \mathcal{S}$ with $(s, t) \in \to$. We usually write $s \to t$ instead of $(s, t) \in \to$. Finally, $\ell : \mathcal{P} \to 2^{\mathcal{S}}$ is a partial labelling function which assigns sets $\ell(p)$ of states in which p is true, to some propositions p. We assume that $\ell(p)$ is defined for finitely many p only, for otherwise it is not clear how an LTS should be finitely representable as (part of the) input to an algorithm solving some computation problem.

Let $S \subseteq \mathcal{S}$. We write $Pre_{\mathcal{T}}(S)$ for the set of predecessors of S, i.e. $\{t \in \mathcal{S} \mid \exists s \in S$ s.t. $t \to s\}$.

A *path* in $\mathcal{T}$ starting in state s is an infinite sequence $\pi = s_0, s_1, \ldots$ such that $s_0 = s$ and $s_i \to s_{i+1}$ for all $i \geq 0$. For such a path π and $i \in \mathbb{N}$ let $\pi(i)$ denote its i-th state, i.e. s_i. Let $\Pi_{\mathcal{T}}(s)$ denote the set of all paths in $\mathcal{T}$ that start in s.

2.2 The Branching-Time Logic CTL

Let $\mathcal{P}$ be as above. Formulas of the branching-time logic CTL are built according to the following grammar.

$$\varphi \quad ::= \quad p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathtt{EX}\varphi \mid \mathtt{E}(\varphi \mathtt{U} \varphi) \mid \mathtt{EG}\varphi$$

where $p \in \mathcal{P}$.

Besides the usual abbreviations for the Boolean operators like $\wedge, \to, \mathtt{tt}, \mathtt{ff}$ we also introduce the standard temporal operators via $\mathtt{E}(\varphi \,\mathtt{R}\, \psi) := \mathtt{E}(\psi \,\mathtt{U}\, (\varphi \wedge \psi)) \vee \mathtt{EG}\psi$, $\mathtt{AX}\varphi :=$

$\neg\mathtt{EX}\neg\varphi$, $\mathtt{A}(\varphi\,\mathtt{R}\,\psi) := \neg\mathtt{E}(\neg\varphi\,\mathtt{U}\,\neg\psi)$, $\mathtt{EF}\varphi := \mathtt{E}(\mathtt{tt}\,\mathtt{U}\,\varphi)$, $\mathtt{AG}\varphi := \neg\mathtt{EF}\neg\varphi$, and $\mathtt{AF}\varphi := \neg\mathtt{EG}\neg\varphi$.

Formulas of CTL are interpreted over labelled transition systems $\mathcal{T} = (\mathcal{S}, \to, \ell)$. The semantics inductively defines the set of states at which each subformula is true.

$$
\begin{aligned}
[\![p]\!]^{\mathcal{T}} &:= \ell(p) \\
[\![\varphi \vee \psi]\!]^{\mathcal{T}} &:= [\![\varphi]\!]^{\mathcal{T}} \cup [\![\psi]\!]^{\mathcal{T}} \\
[\![\neg\varphi]\!]^{\mathcal{T}} &:= \mathcal{S} \setminus [\![\varphi]\!]^{\mathcal{T}} \\
[\![\mathtt{EX}\varphi]\!]^{\mathcal{T}} &:= \{s \in \mathcal{S} \mid \exists t \in \mathcal{S} \text{ s.t. } s \to t \text{ and } t \in [\![\varphi]\!]^{\mathcal{T}}\} \\
[\![\mathtt{E}(\varphi\,\mathtt{U}\,\psi)]\!]^{\mathcal{T}} &:= \{s \in \mathcal{S} \mid \exists \pi \in \Pi_{\mathcal{T}}(s), i \geq 0 \text{ s.t. } \pi(i) \in [\![\psi]\!]^{\mathcal{T}} \\
&\qquad\qquad \text{and for all } j < i : \pi(j) \in [\![\varphi]\!]^{\mathcal{T}}\} \\
[\![\mathtt{EG}\varphi]\!]^{\mathcal{T}} &:= \{s \in \mathcal{S} \mid \exists \pi \in \Pi_{\mathcal{T}}(s) \text{ s.t. for all } i \geq 0 : \pi(i) \in [\![\varphi]\!]^{\mathcal{T}}\}
\end{aligned}
$$

The (global) model checking problem for CTL and a class of labelled transition systems $\mathfrak{K}$ is: given a $\mathcal{T} \in \mathfrak{K}$ and a $\varphi \in$ CTL (over the same set of atomic propositions), compute $[\![\varphi]\!]^{\mathcal{T}}$. It is well-known that the model checking problem for CTL over finite LTS is computable in polynomial time [15].

2.3 Automatic Structures

We are interested in particular LTS over infinite state spaces, known as *automatic structures* [6]. Originally, the term refers to (possibly infinite) relational structures that can be represented using automata. Here we consider a slightly modified variant that does not bear any essential differences. First, we restrict our attention to unary and binary relations – note that LTS are specific relational structures such that the arities of their relations are two (for the transition relation) and one (for all the atomic propositions).

Second, we consider a slight generalisation, owed to the limits that the original proposal faces in terms of decidability issues. In the original definition of automatic structures, relations are recognised by synchronous transducers, i.e. finite automata over alphabets of the form Σ^k for some $k \geq 1$ (which equals the arity of the underlying relation). This makes the concept of an automatic structure a syntactic definition.

The aim of this work is to find (restricted) classes of automatic structures for which the CTL model checking problem is computable. One way to obtain this is to study restricted mechanisms for defining the relations in an LTS. We therefore prefer a semantic definition of automatic structures here, allowing the representation mechanism to become a parameter for a class of infinite-state structures.

We assume the reader to be familiar with the basic notions of formal language theory and the theory of finite-state automata. We use Σ for a finite alphabet and Σ^* for the set of all finite words over Σ. The empty word is denoted by ε.

A nondeterministic finite automaton (NFA) over Σ is a $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$ with finite state set Q, initial state $q_I \in Q$, final states $F \subseteq Q$ and transition relation $\delta : Q \times \Sigma \to 2^Q$. The language of $\mathcal{A}$ is denoted $L(\mathcal{A})$, and it consists of all words $w \in \Sigma^*$ for which there is an accepting run of $\mathcal{A}$ on w. We use the standard homomorphic

extension $\hat{\delta}$ of δ to words via $\hat{\delta}(q,\varepsilon) = \{q\}$ and $\hat{\delta}(q,wa) = \{q' \mid \exists q'' \in \hat{\delta}(q,w) \text{ s.t. } q' \in \delta(q'',a)\}$. Hence, $L(\mathcal{A}) = \{w \mid \exists f \in \hat{\delta}(q_I,w)\}$.

For our notion of automatic structure we need an abstract concept of a mechanism that represents binary relations over words.

Definition 1. A *binary acceptor* $\mathcal{A}$ is any finite representation of a binary relation $R(\mathcal{A}) \subseteq \Sigma^* \times \Sigma^*$.

This yields a parametric notion of automatic structures.

Definition 2. Let $\mathfrak{A}$ be a class of binary acceptors over some alphabet Σ. An LTS $\mathcal{T} = (S, \to, \ell)$ is said to be an $\mathfrak{A}$-*automatic transition system*, or $\mathfrak{A}$-automatic in short, if

- $S = \Sigma^*$,
- for each $p \in \mathcal{P}$ with $\ell(p) \neq \texttt{undef}$ there is an NFA $\mathcal{A}_p$ s.t. $L(\mathcal{A}_p) = \ell(p)$,
- there is a binary acceptor $\mathcal{A}_{\mathsf{tr}} \in \mathfrak{A}$ s.t. $R(\mathcal{A}_{\mathsf{tr}}) = \{(s,t) \mid s \to t\}$.

Thus, roughly speaking, a transition system is automatic, if the labels are represented by an NFA and the transition relation by a binary acceptor. The size of an $\mathfrak{A}$-automatic structure $\mathcal{T}$, denoted $|\mathcal{T}|$, is the sum of the sizes of the NFA used to define the interpretation of the atomic propositions plus the size of the binary acceptor, assuming that some sensible notion of representation size is given for it.

The standard notion of an automatic structure as known from [6] – at least when restricted to one binary and otherwise only unary relations – is obtained in this setting as a $\mathfrak{T}_{\mathsf{sync}}$-automatic transition system with $\mathfrak{T}_{\mathsf{sync}}$ being the class of synchronous transducers, i.e. NFA over the alphabet $\Sigma^2 \cup \{(a,\#),(\#,a) \mid a \in \Sigma\}$. The relation of such a transducer $\mathcal{A}$ is then defined as $R(\mathcal{A}) = \{(u,v) \mid zip(u,v) \in L(\mathcal{A})\}$ where zip merges the two words $u, v \in \Sigma^*$ into a two-tracked word over Σ^2, possibly appending the padding symbol # in case their lengths are not equal. It is inductively defined via

$$zip(au,bv) := \binom{a}{b} zip(u,v) \ , \qquad zip(\varepsilon,bv) := \binom{\#}{b} zip(\varepsilon,v) \ ,$$

$$zip(au,\varepsilon) := \binom{a}{\#} zip(u,\varepsilon) \ , \qquad zip(\varepsilon,\varepsilon) := \varepsilon \ ,$$

where $a, b \in \Sigma$ and $u, v \in \Sigma^*$.

Another example of a binary acceptor is given by the notion of recognisable relations, to be looked at in detail in Sect. 4 as a mechanism to define a class of automatic structures we call *recognisable automatic structures*. The notion of binary acceptor is flexible enough, though, to incorporate all sorts of other mechanisms for defining binary relations. For instance a pair of two NFAs $(\mathcal{A},\mathcal{B})$ with $R(\mathcal{A},\mathcal{B}) = L(\mathcal{A}) \times L(\mathcal{B})$ would also be a very simple case of a binary acceptor, leading to what one may call *fully asynchronous automatic structures*. In fact, such a pair yields a very special case of a recognisable relation.

Algorithm 1 The standard procedure for model checking CTL.

```
procedure ModelCheck(φ)                                   ▷ assume LTS T = (S, →, ℓ) fixed
    case φ of
    p:  return ℓ(p)                                       ▷ p ∈ P
    ¬ψ:
        return S \ ModelCheck(ψ)
    ψ1 ∨ ψ2:
        return ModelCheck(ψ1) ∪ ModelCheck(ψ2)
    EXψ:
        return Pre_T(ModelCheck(ψ))
    E(ψUχ):
        L1 ← ModelCheck(ψ); L2 ← ModelCheck(χ); M ← ∅
        repeat
            M' ← M; M ← L2 ∪ (L1 ∩ Pre_T(M))
        until M = M'
        return M
    EGψ:
        L ← ModelCheck(ψ); M ← S
        repeat
            M' ← M; M ← L ∩ Pre_T(M)
        until M = M'
        return M
    end case
end procedure
```

3 Model Checking CTL

We describe the generic and well-known procedure that can be used to compute the set of states in a transition system which satisfy a given CTL formula [11]. It can immediately be derived from the semantics and the fixpoint principle, stating that the set of states satisfying $\mathtt{E}(\varphi \mathtt{U} \psi)$, resp. $\mathtt{EG}\varphi$, can be computed iteratively in a least, resp. greatest fixpoint recursion.

Note that the procedure ModelCheck as given in Algorithm 1 is not an algorithm strictly speaking: if $|\mathcal{S}| < \infty$ then clearly $Pre_{\mathcal{T}}(\cdot)$ is computable, and termination of the `repeat-until`-loops is guaranteed by monotonicity and boundedness of the values of the variable T in both cases. Hence, procedure ModelCheck can safely be called an algorithm for CTL model checking on finite structures.

In case of $|\mathcal{S}| = \infty$, termination is not necessarily guaranteed. This does not mean, though, that computability of the model checking problem is not given. As in the case of FO model checking on automatic structures which only uses computable operations on possibly infinite sets, a thorough look at ModelCheck reveals some sufficient conditions under which CTL model checking becomes computable. For this, we assume the given LTS to be $\mathfrak{A}$-automatic for some class $\mathfrak{A}$. Then the computability of the Boolean operations is guaranteed for as long as they are applied to sets of states which form a regular language. Moreover, computability of the $Pre(\cdot)$-predicate is needed, which is the counterpart to closure under projections in the decidability proof for FO model

checking on automatic structures. At last, we need one more property which has no counterpart in FO model checking, since FO has no recursion mechanism but CTL has one in the form of the temporal operators EU and EG.

Definition 3. Let $\mathfrak{A}$ be a class of binary acceptors and $\mathfrak{T}$ be a class of $\mathfrak{A}$-automatic structures. We say that $\mathfrak{T}$ has *finite* U*-closure ordinals* if for any $\mathcal{T} \in \mathfrak{T}$ and any regular languages L_1, L_2 the increasing chain $M_0 \subseteq M_1 \subseteq \ldots$ becomes stationary where

$$M_0 := \emptyset \quad , \quad M_{i+1} := L_2 \cup (L_1 \cap Pre_{\mathcal{T}}(M_i)) \, .$$

Likewise, we say that $\mathfrak{T}$ has *finite* G*-closure ordinals* if for any $\mathcal{T} \in \mathfrak{T}$ and any regular language L the decreasing chain $M_0 \supseteq M_1 \supseteq \ldots$ becomes stationary where

$$M_0 := \Sigma^* \quad , \quad M_{i+1} := L \cap Pre_{\mathcal{T}}(M_i) \, .$$

We say that $\mathfrak{T}$ has *finite closure ordinals* if it has finite U- and finite G-closure ordinals.

Here, becoming stationary means that there is an $n \in \mathbb{N}$ such that $M_{n+1} = M_n$. It is a simple consequence of the monotonicity of the operators $Pre_{\mathcal{T}}(\cdot)$, union and intersection that the series $(M_i)_{i \geq 0}$ indeed forms an increasing, resp. decreasing chain.

Lemma 4. *The model checking problem for* CTL *over the class* $\mathfrak{T}$ *of* $\mathfrak{A}$*-automatic transition systems is computable if*

a) for any LTS $\mathcal{T} \in \mathfrak{T}$ *and any regular language L, the set* $Pre_{\mathcal{T}}(L)$ *is effectively regular, i.e. an NFA can be computed for it from an NFA for L, and*
b) $\mathfrak{T}$ *has finite closure ordinals.*

Proof. It is a standard exercise to show by induction on the structure of φ that calling ModelCheck(φ) on $\mathcal{T}$ returns $[\![\varphi]\!]^{\mathcal{T}}$ [4, Thm. 6.23] [13, Lem. 7.3.4], provided that it terminates. It then only remains to see that termination is guaranteed when each call to any of the two repeat-until loops terminates.

First we note that by assumption (a), each subcall to ModelCheck returns a regular language. Then assumption (b) is applicable and guarantees termination of the loops since they iterate through the values of the chains from Def. 3 in their variables M and M' until they become stable. □

4 CTL Model Checking over Recognisable Automatic Transition Systems

In this section we examine a particular class of binary acceptor and the computability of the CTL model checking problem over automatic structures generated by this class. Semantically, it consists of the class of recognisable relations which forms a proper subclass of the relations represented by synchronous transducers. These, in turn, are included in the well-known class of rational relations [29, Thm. 6.4].

An automaton model for the class of recognisable relations can immediately be derived from the fact that every recognisable relation can be expressed as the finite union of the product of some regular languages [5, Thm. 1.5]. This gives rise to a syntactic transducer model for these relations.

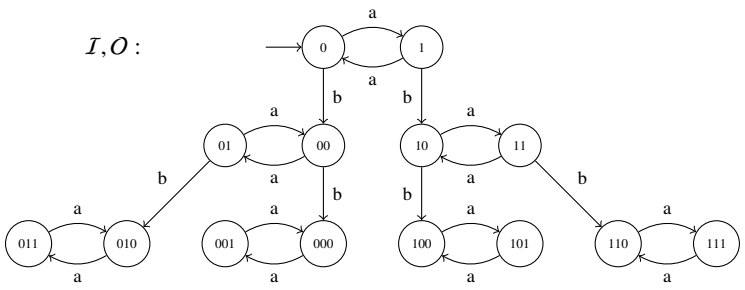
Definition 5. An *input-output-independent (IOI) automaton* is a triple $\mathcal{A} = (\mathcal{I}, \mathcal{O}, F)$ such that $\mathcal{I} = (Q^{\mathcal{I}}, \Sigma, q_I^{\mathcal{I}}, \delta^{\mathcal{I}}, \emptyset)$ and $\mathcal{O} = (Q^{\mathcal{O}}, \Sigma, q_I^{\mathcal{O}}, \delta^{\mathcal{O}}, \emptyset)$ are NFAs and $F \subseteq Q^{\mathcal{I}} \times Q^{\mathcal{O}}$.

The relation defined by an IOI automaton is

$$R(\mathcal{A}) := \{(u, v) \in \Sigma^* \times \Sigma^* \mid \exists (p, q) \in F \text{ s.t. } p \in \hat{\delta}_i^{\mathcal{I}}(q_I^{\mathcal{I}}, u) \text{ and } q \in \hat{\delta}_i^{\mathcal{O}}(q_I^{\mathcal{O}}, v)\} \,.$$

Intuitively, an IOI automaton is a pair of NFAs which are only synchronised via final states. They read the input and output word independently, and the acceptance condition prescribes which pairs of states their runs need to end in for the pair of words to be accepted. Clearly, IOI automata are a special form of binary acceptors according to Def. 1. Hence, they give rise to a class of automatic structures, henceforth called *recognisable automatic structures*.

Example 6. Let $\mathcal{A} = (\mathcal{I}, \mathcal{O}, F)$ be the IOI automaton such that $\mathcal{I}$ and $\mathcal{O}$ are both the following NFA.



A state at the bottom is reached by a word that contains exactly two b's, hence, it is of the form $a^{n_1}ba^{n_2}ba^{n_3}$ for some $n_1, n_2, n_3 \geq 0$. Such a state $x_1x_2x_3$ then indicates the parities (even/odd) of n_1, n_2 and n_3.

The final state pairs of $\mathcal{A}$ are those of the row at the bottom that differ in at least two positions, i.e.

$$F \;:=\; \{(x_1x_2x_3, y_1y_2y_3) \mid x_i \neq y_i \text{ for at least two } i \in \{1, 2, 3\}\} \,.$$

Thus, a pair of words $(a^{n_1}ba^{n_2}ba^{n_3}, a^{m_1}ba^{m_2}ba^{m_3})$ is in $R(\mathcal{A})$, iff $n_i = m_i \mod 2$ for at most one $i \in \{1, 2, 3\}$.

$\mathcal{A}$ generates a recognisable automatic structure with state space $\{a, b\}^*$ that is partly shown in Fig. 1. The grey circles denote subgraphs of nodes of the form $a^{n_1}ba^{n_2}ba^{n_3}$ for which the values of $n_1 + n_2 + n_3$ do not differ. The dashed line abbreviates edges from every node in the left subgraph to the node on the right. Note that $R(\mathcal{A})$ is symmetric in this case, simply because $\mathcal{I} = \mathcal{O}$ and F happens to be symmetric.

In order to prove computability of the model checking problem for CTL over recognisable automatic structures it suffices to show that this class satisfies the two conditions laid out in Lemma 4.

Lemma 7. *Let L be a regular language and $\mathcal{T}$ be a recognisable automatic structure. Then $Pre_{\mathcal{T}}(L)$ is effectively regular.*

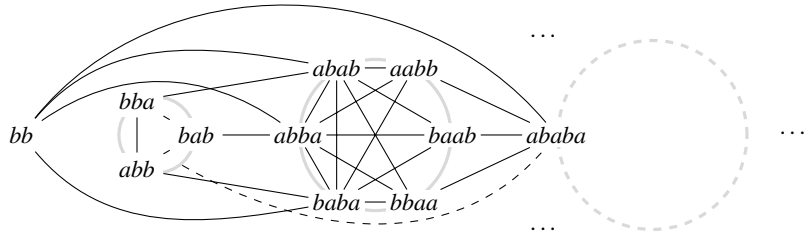


Fig. 1. An excerpt of the relation $R(\mathcal{A})$ for the IOI automaton $\mathcal{A}$ from Ex. 6.

Proof. Let L be a regular language accepted, e.g., by some NFA $\mathcal{B} = (Q^{\mathcal{B}}, \Sigma, q_I^{\mathcal{B}}, \delta^{\mathcal{B}}, F^{\mathcal{B}})$ and let $\mathcal{A} = (\mathcal{I}, \mathcal{O}, F^{\mathcal{A}})$ be the IOI automaton that recognises the transition relation of some recognisable automatic structure $\mathcal{T}$. Consider the IOI automaton $\mathcal{A} \rtimes \mathcal{B} := (\mathcal{I}, \mathcal{O}', F)$ with $\mathcal{O}' = (Q^{\mathcal{O}} \times Q^{\mathcal{B}}, \Sigma, (q_I^{\mathcal{O}}, q_I^{\mathcal{B}}), \delta, \emptyset)$,

$$\delta((p,q),a) = \{(p',q') \mid p' \in \delta^{\mathcal{O}}(p,a), q' \in \delta^{\mathcal{B}}(q,a)\}$$

and $F := \{(f_1,(f_2,f)) \mid (f_1,f_2) \in F^{\mathcal{A}}, f \in F^{\mathcal{B}}\}$. It has the same input component as $\mathcal{A}$, but its output component $\mathcal{O}'$ is the synchronous product of the one of $\mathcal{A}$ and $\mathcal{B}$. Hence, it recognises the relation $R(\mathcal{A} \rtimes \mathcal{B}) = \{(u,v) \mid (u,v) \in R(\mathcal{A}) \text{ and } v \in L(\mathcal{B})\}$.

Next, consider the NFA $\mathcal{I}' := (Q^{\mathcal{I}}, \Sigma, q_I^{\mathcal{I}}, \delta^{\mathcal{I}}, F')$ with $F' := \{f_1 \mid \exists (f_2, f) \text{ s.t. } (f_1,(f_2,f)) \in F\}$. We then have $L(\mathcal{I}') = \{u \mid \exists v \text{ s.t. } (u,v) \in R(\mathcal{A} \rtimes \mathcal{B})\} = Pre_{\mathcal{T}}(L)$. □

This is of course a standard construction of forming the intersection of the automaton's second component with a regular language and then projecting it onto its first component. We have spelled out the construction in detail because of an important observation to be made: note that the transition table of the NFA for $Pre_{\mathcal{T}}(L)$ does not depend on L; instead, L only determines its accepting states. This can be seen as an indication of the weakness of IOI automata as a model for automatic structures; however, some sort of weakness is necessary in order to obtain computability.

Lemma 8. *The class of recognisable automatic structures has finite closure ordinals.*

Proof. We will only prove the claim for finite U-closure ordinals. The case of G-closure ordinals is analogous.

Let $\mathcal{A} = (\mathcal{I}, \mathcal{O}, F)$ be the IOI automaton underlying some recognisable automatic structure $\mathcal{T}$, and let L_1, L_2 be two regular languages. Consider the chain $M_0 \subseteq M_1 \subseteq \ldots$ approximating the set of states in $\mathcal{T}$ that satisfy – by slight abuse of notation – $\mathsf{E}(L_1 \,\mathsf{U}\, L_2)$, as constructed in Def. 3.

By the observation following the previous lemma, we have that $Pre_{\mathcal{T}}(L)$ is recognised by an NFA of the form $(Q^{\mathcal{I}}, \Sigma, q_I^{\mathcal{I}}, \delta^{\mathcal{I}}, F)$ for some $F \subseteq Q^{\mathcal{I}}$. Thus, the graph structure of the NFA does not depend on the input language L, only the set of final states does. Therefore, there are at most $2^{|Q^{\mathcal{I}}|}$ many different languages $Pre_{\mathcal{T}}(L)$ for arbitrary regular L.

Now consider the chain $M_0 \subseteq M_1 \ldots$. Each M_i with $i > 0$ is obtained as $L_2 \cup (L_1 \cap Pre_{\mathcal{T}}(M_{i-1}))$. Assuming that union and intersection are always formed using the same procedure on the same fixed NFA for L_1 and L_2, we get that there are at most $2^{|Q^{\mathcal{I}}|}$ many different NFA for the M_i. With union and intersection being monotone operations, the chain $M_0 \subseteq M_1 \subseteq \ldots$ has to become stable after at most $2^{|Q^{\mathcal{I}}|}$ many steps. □

Putting Lemmas 4, 7 and 8 together, we immediately obtain the following.

Theorem 9. *The model checking problem for* CTL *over the class of recognisable automatic structures is computable.*

An immediate question arising from such a decidability result concerns the worst-case complexity of the model checking problem for CTL over recognisable automatic transition systems. We note that the time needed to compute $[\![\varphi]\!]^{\mathcal{T}}$ for some such $\mathcal{T}$ and arbitrary CTL formula φ is determined by several factors: (1) the use of intersection and complementation constructions arising from conjunctions and negated subformulas; (2) upper bounds on the number of iterations needed to obtain stability in the in-/decreasing chains of Def. 3. The following lemma shows that stability is reached after a small number of iterations.

Lemma 10. *Consider an IOI automaton $\mathcal{A} = (\mathcal{I}, \mathcal{O}, F)$ and two regular languages $L_1, L_2 \subseteq \Sigma^*$ represented by NFA $\mathcal{B}_1, \mathcal{B}_2$. Let $\mathcal{A}_0, \mathcal{A}_1, \ldots$ be the sequence of NFA recognising the languages $M_0, M_1, \ldots$ in an in-/decreasing chain according to Def. 3, and let $F_0, F_1, \ldots$ be their final states respectively. Then $F_0, F_1, \ldots$ also forms an increasing, resp. decreasing chain.*

Proof. We assume that in each step of building the $\mathcal{A}_i$, $i \geq 1$, the standard constructions for forming the union and intersection of two languages are being used. Hence, for every final state f in some F_i we have that f is either a final state of $\mathcal{B}_2$, or it is of the form (f', f'') such that f' is a final state of $\mathcal{B}_1$ and f'' is a final state of the NFA constructed in the proof of Lemma 7 by projecting the automaton $\mathcal{A} \rtimes \mathcal{A}_{i-1}$ accordingly.

Now consider the case in which $M_0 \subseteq M_1 \subseteq \ldots$ forms an increasing chain. The case of a decreasing chain is entirely analogous. W.l.o.g. we can assume that $F_0 = \emptyset$ since $M_0 = \emptyset$. Clearly, we have $F_0 \subseteq F_1$. Now let $i > 0$ and assume that $F_{i-1} \subseteq F_i$. We want to show that $F_i \subseteq F_{i+1}$ holds.

Take some $f \in F_i$. If f is a final state of $\mathcal{B}_2$ then it clearly also belongs to F_{i+1}. Hence, suppose that $f = (f', f'')$ with f' being a final state of $\mathcal{B}_1$ and f'' being a final state of the NFA for $Pre_{\mathcal{T}}(M_{i-1})$. According to the construction of the automaton $\mathcal{A} \rtimes \mathcal{A}_{i-1}$ as in the proof of Lemma 7 there must exist some g, g' such that $(f'', (g, g'))$ is a final state of $\mathcal{A} \rtimes \mathcal{A}_{i-1}$. This is only possible if (f'', g) is a final state of the automaton $\mathcal{A}$ and g' is a final state of the NFA $\mathcal{A}_{i-1}$, thus $g' \in F_{i-1}$. Then we can apply the induction hypothesis and get $g' \in F_i$ and therefore $(f'', (g, g'))$ as a final state of $\mathcal{A} \rtimes \mathcal{A}_i$. Then f'' is also a final state of the NFA for $Pre_{\mathcal{T}}(M_i)$ and therefore f is a final state of the NFA for $L_1 \cap Pre_{\mathcal{T}}(M_i)$ and, hence, $f \in F_{i+1}$. □

Note that this does not necessarily yield a polynomial bound on the number of iterations needed to compute $[\![\mathtt{E}(\varphi \mathtt{U} \psi)]\!]^{\mathcal{T}}$ for instance. Lemma 10 shows that the fixpoint will be reached after after at most $n_\varphi + n_\psi \cdot n$ steps where n_φ, n_ψ are the number of states

of an NFA recognising $[\![\varphi]\!]^{\mathcal{T}}$ and $[\![\psi]\!]^{\mathcal{T}}$, respectively. Again, similar considerations can be made for the decreasing chains in Def. 3 and formulas of the form $\mathtt{EG}\varphi$. In any case, n equals the number of states of the output component of the IOI automaton recognising the accessibility relation of the underlying recognisable transition system. Hence, n is clearly bounded by $|\mathcal{T}|$, the size of a representation of $\mathcal{T}$. However, n_φ and n_ψ are not a priori bounded since these subformulas can be arbitrary and in particular make use of expensive intersection and complementation constructions.

5 Conclusion and Further Work

We have defined a simple framework for the study of restricted classes of automatic structures in which the binary relations are defined by weaker automata than synchronous ones. This can of course be extended to relations of arbitrary arity, but automatic structures that represent transition systems (i.e. have relations of arity at most two) are most interesting for purposes of verification of reactive and concurrent systems. This also motivates the choice of specification language, here the branching-time temporal logic CTL.

There are plenty of ways that this work can be extended to in the future. The exact complexity of CTL model checking over recognisable automatic transition systems needs to be established. It also remains to be seen whether the sufficient conditions (or similar ones) on IOI automata can be used to prove decidability of model checking problems for richer or similar specification languages like PDL [17] with various extensions [30], regular extensions of CTL [19, 24, 3] or even the modal μ-calculus [23]. Note that these logics are all state-based in the sense that typical global model checking procedures can proceed in a bottom-up fashion similar to Alg. 1.

The next question that comes up in terms of investigations w.r.t. specification languages concerns linear-time logics like LTL [27] and PSL [2] and then combinations with branching-time features resulting in something like CTL* [16]. Note that model checking for such logics typically requires very different techniques like automata- [33] or tableau-based [35] ones. It therefore remains to see if the sufficient conditions laid out in Lemma 4 would also yield computability of model checking problems for linear-time properties, or whether other conditions can be found similarly.

Another obvious direction for future work is of course to find further instantiations of the relaxed framework of binary acceptors which preferably leads to richer classes of automatic structures but still satisfies the conditions of Lemma 4. One way to go about this is to give up working with essentially two-tracked words since this is one of the main course of undecidability. A simple suggestion for a binary acceptor that is based in the world of one-tracked words is, for instance, the following: given an NFA $\mathcal{A}$, let $R(\mathcal{A}) = \{(u, v) \mid uv \in L(\mathcal{A})\}$. Hence, it defines a relation by cutting words in a regular language apart. It is a simple exercise, though, to see that this model of binary acceptor is effectively equivalent to the IOI automata studied here. Hence, it does not generate a new class. We therefore propose a slight variant and leave it open whether this model of binary acceptor satisfies the conditions of Lemma 4: given an NFA $\mathcal{A}$, let $R(\mathcal{A}) = \{\, (u, v) \mid$ there is $w \in L(\mathcal{A})$ such that u is a prefix of w and v is a suffix of $w\}$. We

suspect that CTL model checking is computable for the class of automatic structures defined by such binary acceptors but have no formal proof at the moment.

We also suspect that recognisable relations may form the largest class of syntactically definable relations for which CTL model checking, or even model checking for some weaker logic like EF, is decidable. It remains to be seen whether it is possible to encode some undecidable reachability problem using an arbitrary relation that incorporates only the slightest form of synchronisation between the runs on the input and the output word.

References

1. P. A. Abdulla, B. Jonsson, M. Nilsson, and J. d'Orso. Regular model checking made simple and efficient. In *Proc. 13th Int. Conf. on Concurrency Theory, CONCUR'02*, volume 2421 of *LNCS*, pages 116–130. Springer, 2002.
2. I. Accellera Organization. Formal semantics of Accellera property specification language, 2004. In Appendix B of `http://www.eda.org/vfv/docs/PSL-v1.1.pdf`.
3. R. Axelsson, M. Hague, S. Kreutzer, M. Lange, and M. Latte. Extended computation tree logic. In *Proc. 17th Int. Conf. on Logic for Programming and Artificial Reasoning, LPAR'10*, volume 6397 of *LNCS*, pages 67–81. Springer, 2010.
4. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
5. J. Berstel. *Transductions and Context-Free Languages*. Leitfäden der angewandten Mathematik und Mechanik. Teubner, 1979.
6. A. Blumensath. Automatic structures. Master's thesis, RWTH Aachen, 1999.
7. A. Blumensath and E. Grädel. Automatic structures. In *Proc. 15th Symp. on Logic in Computer Science, LICS'00*, pages 51–62. IEEE, 2000.
8. A. Bouajjani, P. Habermehl, and T. Vojnar. Abstract regular model checking. In *Proc. 16th Int. Conf. on Computer Aided Verification, CAV'04*, volume 3114 of *LNCS*, pages 372–386. Springer, 2004.
9. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Proc. 12th Int. Conf. on Computer Aided Verification, CAV'00*, volume 1855 of *LNCS*, pages 403–418. Springer, 2000.
10. O. Burkart and B. Steffen. Model checking the full modal μ-calculus for infinite sequential processes. In *Proc. 24th Coll. on Automata, Languages and Programming, ICALP'97*, volume 1256 of *LNCS*, pages 419–429. Springer, 1997.
11. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Proc. Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71, Yorktown Heights, New York, 1981. Springer.
12. D. Dams, Y. Lakhnech, and M. Steffen. Iterating transducers. *J. Log. Algebr. Program*, 52-53:109–127, 2002.
13. S. Demri, V. Goranko, and M. Lange. *Temporal Logics in Computer Science*, volume I – Finite State Systems of *Cambridge Tracts in Theor. Comp. Sc.* Cambridge Univ. Press, 2016.
14. E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs as fixpoints. In *Proc. 7th Int. Coll. on Automata, Languages and Programming, ICALP'80*, volume 85 of *LNCS*, pages 169–181. Springer, 1981.
15. E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.
16. E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

17. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
18. P. Habermehl and T. Vojnar. Regular model checking using inference of regular languages. In *Proc. 6th Int. Workshop on Verification of Infinite-State Systems, INFINITY'04*, volume 138(3), pages 21–36, 2005.
19. K. Hamaguchi, H. Hiraishi, and S. Yajima. Branching time regular temporal logic for model checking with linear time complexity. In E. M. Clarke and R. P. Kurshan, editors, *Proc. 2nd Int. Conf. on Computer Aided Verification, CAV'90*, volume 531 of *LNCS*, pages 253–262, Berlin, Germany, 1991. Springer.
20. C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.
21. B. Jonsson and M. Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In *Proc. 6th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems, TACAS'00*, volume 1785 of *LNCS*, pages 220–234. Springer, 2000.
22. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In *Proc. 9th Int. Conf. on Computer Aided Verification, CAV'97*, volume 1254 of *LNCS*, pages 424–435. Springer, 1997.
23. D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.
24. R. Mateescu, P. T. Monteiro, E. Dumas, and H. de Jong. Computation tree regular logic for genetic regulatory networks. In *Proc. 6th Int. Conf. on Automated Technology for Verification and Analysis, ATVA'08*, volume 5311 of *LNCS*, pages 48–63. Springer, 2008.
25. R. Milner. A calculus of communicating systems. *LNCS*, 92, 1980.
26. C.-H. L. Ong. Higher-order model checking: An overview. In *Proc. 30th IEEE Symp. on Logic in Computer Science, LICS'15*, pages 1–15. IEEE Computer Society, 2015.
27. 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.
28. W. Reisig. *Petri Nets (an Introduction)*. Number 4 in EATCS Monographs on Theoretical Computer Science. Springer, 1985.
29. J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.
30. R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1/2):121–141, 1982.
31. K. Sutner. Iterating transducers. *Fundam. Inform*, 138(1-2):259–272, 2015.
32. T. Touili. Regular model checking using widening techniques. In *Proc. Workshop on Verification of Parameterized Systems, VEPAS'01*, volume 50(4) of *Electr. Notes Theor. Comput. Sci*, pages 342–356. Elsevier, 2001.
33. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
34. I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.
35. P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, 28(110–111):119–136, 1985.
36. P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. Conf. on Computer Aided Verification, CAV'98*, volume 1427 of *LNCS*, pages 88–97. Springer, 1998.