Using Model Checking to Solve Supervisor Synthesis Problems

Andreas Morgenstern and Klaus Schneider

Abstract—Verification procedures, which check whether a given system satisfies a given specification, are nowadays mature for industrial usage. The more general supervisor synthesis problem asks how a system has to be restricted or which actions have to be selected such that the system satisfies a given specification. Supervisor synthesis problems are often formulated in frameworks like game structures that are more general than the Kripke structures that are traditionally used in verification. For this reason, current verification tools can not be used for supervisory control problems.

In this paper, however, we present a reduction of alternating time $\mu$-calculus model checking problems (on game structures) to model checking problems of the $\mu$-calculus on Kripke structures. As a result, arbitrary model checkers can be used to solve supervisor synthesis problems. As a demonstration of the applicability of our approach, we show how the classical supervisory control problems of Ramadge and Wonham can be solved within our framework.

I. INTRODUCTION

In the past two decades, many verification procedures for the temporal behavior of reactive systems have been developed [Schneider, 2003], and this research already lead to tools that are used in industry. These tools are able to check whether a system $K$ satisfies a given temporal specification $\varphi$. There are a lot of formalisms, in particular, the $\mu$-calculus [Kozen, 1983], $\omega$-automata [Thomas, 1990], as well as various temporal logics [Pnueli, 1977], [Emerson and Clarke, 1982] and (monadic) predicate logics [Büchi, 1960] to formulate the specification $\varphi$ [Schneider, 2003]. Moreover, the increased industrial interest already lead to standardization efforts on specification logics [Beer et al., 2001], [Accellera, 2004].

Besides the verification problem, where the entire system $K$ and its specification must be already available, the more general controller/supervisor synthesis problem is of interest for the development of safety-critical systems. The task is here to check whether it is possible to restrict a given system $K$ such that the restriction $K'$ satisfies a given specification $\varphi$. Obviously, this problem is more general than the verification problem. Efficient solutions for this problem could be used to guide design decisions in the development of reactive systems.

The controller synthesis problem is not new: well established variants are, in particular, the supervisory control problem [Ramadge and Wonham, 1987], module checking [Kupferman and Vardi, 1996], and alternating time $\mu$-calculus model checking [Alur et al., 2002]. In all of the mentioned references, the general question is to check whether the system can be restricted such that a given specification is met. While in [Kupferman and Vardi, 1996] and [Alfaro et al., 2001], it was discussed whether supervisor synthesis problems can be solved by ordinary model checkers, [Alur et al., 2002] and [Ramadge and Wonham, 1987] used special specification languages and tools. We briefly consider the two latter approaches.

In supervisory control [Ramadge and Wonham, 1987], both the system and the specification are usually given in form of finite automata, and the specifications are usually safety and nonblocking properties. Recently, this formalism has been extended in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005] to more general specifications given in the $\mu$-calculus. This is achieved by translating supervisory control problems to model checking problems on a Kripke structure. However, the temporal logic that was used required a slight extension of the $\mu$-calculus by a special operator $\kappa$.

Alternating time $\mu$-calculus specifications as presented in [Alur et al., 2002] are evaluated on finite state transition systems where two players (representing the system and the environment) concurrently and independently select actions that are allowed in the considered state. Alternating time $\mu$-calculus extends the ordinary $\mu$-calculus with new modal operators to describe that one of the players can enforce by its choice of an action that a certain state set is reached, regardless of the choice of the other player. Using alternating time $\mu$-calculus, it is therefore easier to formulate statements like ‘the controller can enforce that certain states are visited irrespectively of how the environment behaves’. Such typical properties can not be directly formulated using the ordinary $\mu$-calculus. Therefore, alternating time $\mu$-calculus and the derived alternating time temporal logics ATL and ATL* have been proposed as extensions to traditional temporal logics like LTL or CTL* that are already used for verification.

In this paper, however, we will show how alternating time $\mu$-calculus model checking problems (on concurrent game structures) are translated to ordinary $\mu$-calculus model checking problems (on Kripke structures). The purpose of this translation is to use the efficient machinery offered by existing model checking tools to solve supervisory control problems. We emphasize that we do not argue that alternating time $\mu$-calculus is unnecessary. Instead, we want to show that already available $\mu$-calculus model checkers can be used to solve problems that were formulated with alternating time $\mu$-
To demonstrate the applicability of our translation, we describe the supervisory control problem of [Ramadge and Wonham, 1987] in alternating time $\mu$-calculus, which allows us a further translation to an ordinary $\mu$-calculus model checking problem. In comparison to the translation given in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005], the approach presented in this paper does not require the introduction of new operators in the $\mu$-calculus. Hence, a side product of our approach is the reduction of the classical supervisory control problem to a pure $\mu$-calculus model checking problem that can be solved with standard tools.

The outline of the paper is as follows: In the next section, we review basic definitions of Kripke structures, the propositional $\mu$-calculus, and the alternating time $\mu$-calculus. In Section III, we present our reduction from alternating time $\mu$-calculus model checking to the propositional $\mu$-calculus model checking. Section IV describes the reduction from supervisory control problems to alternating time $\mu$-calculus model checking.

II. FORMAL BACKGROUND

A. The Propositional $\mu$-Calculus

The propositional $\mu$-calculus is a well-known specification logic whose model checking algorithms form the heart of state-of-the-art verification tools [Schneider, 2003]. We briefly present its syntax and semantics in this section.

Definition 1 (Syntax of $\mu$-Calculus): Given a set of variables $\mathcal{V}$, the set of $\mu$-calculus formulas over $\mathcal{V}$ is defined as the least set $\mathcal{L}_\mu$ that satisfies the following rules:

- $\mathcal{V} \cup \{0, 1\} \subseteq \mathcal{L}_\mu$
- $\neg \varphi, \varphi \land \psi, \varphi \lor \psi \in \mathcal{L}_\mu$, provided that $\varphi, \psi \in \mathcal{L}_\mu$
- $\diamond \varphi, \Box \varphi \in \mathcal{L}_\mu$, provided that $\varphi \in \mathcal{L}_\mu$
- $\mu x. \varphi, \nu x. \varphi \in \mathcal{L}_\mu$, provided that $\varphi \in \mathcal{L}_\mu$ and $x \in \mathcal{V}$.

The semantics is defined on labeled transition systems which are often called Kripke structures due to the relationship to modal logic [Schneider, 2003].

Definition 2 (Kripke Structures): A Kripke structure $\mathcal{K} = (I, S, R, \mathcal{L})$ for a finite set of variables $\mathcal{V}$ is given by a finite set of states $S$, a set of initial states $I \subseteq S$, a transition relation $R \subseteq S \times S$, and a label function $\mathcal{L} : S \rightarrow 2^\mathcal{V}$ that maps each state to a set of variables. The meaning of a $\mathcal{L}_\mu$ formula is a set of states that is recursively described in the next definition. To this end, we make use of the following abbreviations of existential and universal predecessors of a state set $Q$:

\[
\begin{align*}
\text{pre}^R_{\exists} (Q) &= \{ s \in S \mid \exists s' \in S \cdot R(s, s') \land s' \in Q \} \\
\text{pre}^R_{\forall} (Q) &= \{ s \in S \mid \forall s' \in S \cdot R(s, s') \rightarrow s' \in Q \}
\end{align*}
\]

$\text{pre}^R_{\exists} (Q)$ is the set of states that have at least one successor state in $Q$, and $\text{pre}^R_{\forall} (Q)$ is the set of states that have no successor states outside $Q$. Using these set operators, the semantics is as follows:

**Definition 3 (Semantics of $\mu$-Calculus):** Given a Kripke structure $\mathcal{K} = (I, S, R, \mathcal{L})$ over the variables $\mathcal{V}$, we associate with each formula $\Phi \in \mathcal{L}_\mu$ a set of states $[[\Phi]]_{\mathcal{K}} \subseteq S$ by the following rules:

- $[[\emptyset]]_{\mathcal{K}} := \{\}$
- $[[1]]_{\mathcal{K}} := S$
- $[[\varphi]]_{\mathcal{K}} := \{ s \in S \mid x \in \mathcal{L}(s) \}$ for all variables $x \in \mathcal{V}$
- $[[\neg \varphi]]_{\mathcal{K}} := S \setminus [[\varphi]]_{\mathcal{K}}$
- $[[\varphi \lor \psi]]_{\mathcal{K}} := [[\varphi]]_{\mathcal{K}} \cup [[\psi]]_{\mathcal{K}}$
- $[[\varphi \land \psi]]_{\mathcal{K}} := [[\varphi]]_{\mathcal{K}} \cap [[\psi]]_{\mathcal{K}}$
- $[[\diamond \varphi]]_{\mathcal{K}} := \{ s \in S \mid \exists s' \in S \cdot R(s, s') \land s' \in [[\varphi]]_{\mathcal{K}} \}$
- $[[\Box \varphi]]_{\mathcal{K}} := \{ s \in S \mid \forall s' \in S \cdot R(s, s') \rightarrow s' \in [[\varphi]]_{\mathcal{K}} \}$
- $[[\mu x. \varphi]]_{\mathcal{K}} := \{ s \in S \mid [[\varphi]]_{\mathcal{K}} \subseteq S \}$, where $[[\varphi]]_{\mathcal{K}}$ is obtained by changing the label function $\mathcal{L}$ of $\mathcal{K}$ such that exactly the states in $Q$ are labeled with $x$.
- $[[\nu x. \varphi]]_{\mathcal{K}} := \{ s \in S \mid [[\varphi]]_{\mathcal{K}} \subseteq S \}$, where $[[\varphi]]_{\mathcal{K}}$ is defined as above.

**B. Alternating Time $\mu$-Calculus**

Concurrent game structures [Alur et al., 2002] extend Kripke structures as follows: For every transition, two players $A$ and $B$ choose independently of each other actions $\alpha$ and $\beta$, respectively, and the resulting pair $(\alpha, \beta)$ uniquely determines the next state of the game.

**Definition 4 (Concurrent Game Structure):** A concurrent game structure $\mathcal{G} = (I, S, R, \Gamma_A, \Gamma_B, \mathcal{L})$ for a set of variables $\mathcal{V}$ and a set of actions $A$ is given by a finite set of states $S$, a set of initial states $I \subseteq S$, a partial transition function $\delta : S \times A \times A \rightarrow S$, and label functions $\Gamma_A, \Gamma_B : S \rightarrow 2^A$, and $\mathcal{L} : S \rightarrow 2^\mathcal{V}$.

![Example for a Concurrent Game Structure](image-url)

Figure 1 illustrates an example for a concurrent game structure that formalizes a protocol for a train entering a railroad station (we use * to represent any action).

Since concurrent game structures are generalizations of Kripke structures, we can directly use Definition 3 to evaluate $\mu$-calculus formulas on concurrent game structures. However, the additional labels on the transitions induce further modal operators [Alur et al., 2002]: These additional modal operators express that one of the players can enforce by a suitable choice of an action that, regardless of the choice of the other player, a certain state set is reached. The addition of these modal operators to the $\mu$-calculus yields the alternating time $\mu$-calculus:
Definition 5 (Syntax of Alternating Time μ-Calculus): Given a set of variables \( \mathcal{V} \), the formulas of the alternating time μ-calculus \( \mathcal{L}_{AT} \) is the least set that satisfies the following conditions:
- \( \mathcal{V} \cup \{0, 1\} \subseteq \mathcal{L}_{AT} \)
- \( \neg \phi, \varphi \wedge \psi, \varphi \vee \psi \in \mathcal{L}_{AT} \), provided that \( \varphi, \psi \in \mathcal{L}_{AT} \)
- \( \bigcirc \varphi, \Box \varphi \in \mathcal{L}_{AT} \), provided that \( \varphi \in \mathcal{L}_{AT} \)
- \( \mu_x \varphi, v_x \varphi \in \mathcal{L}_{AT} \), provided that \( \varphi \in \mathcal{L}_{AT} \)

The semantics of \( \mathcal{L}_{AT} \) is defined on concurrent game structures analogously to the semantics of \( \mu \)-calculus on Kripke structures. The interesting new point is the semantics of the new modal operators \( \Box_A \) and \( \Box_B \):

Definition 6 (Semantics of \( \mathcal{L}_{AT} \)): Given a concurrent game structure \( G = (\mathcal{I}, \mathcal{S}, \delta, \Gamma_A, \Gamma_B, \mathcal{L}) \) for a set of variables \( \mathcal{V} \) and a set of actions \( \mathcal{A} \), we associate with each formula \( \Phi \in \mathcal{L}_{AT} \) a set of states \( \langle \Phi \rangle_G \subseteq \mathcal{S} \) by following the rules:
- \( \langle \top \rangle_G := \{ \} \)
- \( \langle \bot \rangle_G := \mathcal{S} \)
- \( \langle x \rangle_G := \{ s \in \mathcal{S} \mid \mathcal{L}(s) \} \) for all variables \( x \) \in \( \mathcal{V} \)
- \( \langle \neg \varphi \rangle_G := \mathcal{S} \setminus \langle \varphi \rangle_G \)
- \( \langle \varphi \wedge \psi \rangle_G := \langle \varphi \rangle_G \cap \langle \psi \rangle_G \)
- \( \langle \varphi \vee \psi \rangle_G := \langle \varphi \rangle_G \cup \langle \psi \rangle_G \)
- \( \langle \Box \varphi \rangle_G := \{ s \in \mathcal{S} \mid \forall \alpha \in \Gamma_A(s). \exists b \in \Gamma_B(s). \delta(s, a, b) \in \langle \varphi \rangle_G \} \)
- \( \langle \Box_A \varphi \rangle_G := \{ s \in \mathcal{S} \mid \exists a \in \Gamma_A(s). \forall b \in \Gamma_B(s). \delta(s, a, b) \in \langle \varphi \rangle_G \} \)
- \( \langle \Box_B \varphi \rangle_G := \{ s \in \mathcal{S} \mid \exists b \in \Gamma_B(s). \forall a \in \Gamma_A(s). \delta(s, a, b) \in \langle \varphi \rangle_G \} \)
- \( \langle \mu_x \varphi \rangle_G := \bigcap \{ Q \subseteq \mathcal{S} \mid \langle \varphi \rangle_Q \subseteq Q \} \), where \( \subseteq Q \) is obtained by changing the label function \( \mathcal{L} \) of \( G \) such that exactly the states in \( Q \) are labeled with \( x \).
- \( \langle v_x \varphi \rangle_G := \bigcup \{ Q \subseteq \mathcal{S} \mid \langle \varphi \rangle_Q \subseteq Q \} \), where \( \subseteq Q \) is defined as above.

Intuitively, a state \( s \) satisfies \( \Box_A \varphi \) if player \( A \) can choose an action in state \( s \) such that no matter which action player \( B \) chooses, a state \( s' \) is reached where \( \varphi \) holds. Analogously, \( \Box_B \varphi \) is the set of states where player \( B \) can enforce that a state \( s' \) where \( \varphi \) holds is reached, independently of \( A \)'s choice. Note that the above four modal operators cover all possible quantifier prefixes. As \( \neg \Box \varphi \) is equivalent to \( \Diamond \varphi \), we can even eliminate either \( \Box \) or \( \Diamond \). Note, however, that there is no such duality between \( \Box_A \) and \( \Box_B \), since their duals correspond to two further operators.

III. TRANSLATING \( \mathcal{L}_{AT} \) TO \( \mathcal{L}_\mu \)

In this section, we show that \( \mathcal{L}_{AT} \) model checking and \( \mathcal{L}_\mu \) model checking are equivalent problems that can be easily reduced to each other. Thus, we can solve essentially the same problems with both formalisms. As \( \mathcal{L}_{AT} \) and concurrent game structures are extensions of \( \mathcal{L}_\mu \) and Kripke structures, respectively, one reduction is immediately clear. To show the other reduction, we formally describe a translation from concurrent game structures to associated Kripke structures, and from \( \mathcal{L}_{AT} \) formulas \( \varphi \) to corresponding \( \mathcal{L}_\mu \) formulas \( \mu_A(\varphi) \).

Definition 7 (Kripke Structure of a Game Structure): Given a concurrent game structure \( G = (\mathcal{I}, \mathcal{S}, \delta, \Gamma_A, \Gamma_B, \mathcal{L}) \) for a set of variables \( \mathcal{V} \) and a set of actions \( \mathcal{A} \), we define the associated Kripke structure \( K_G = (\mathcal{I}', \mathcal{S}', \mathcal{R}, \mathcal{L}') \) over the variables \( \mathcal{V} \cup \{x_A, x_B, x_N\} \) as follows:
- \( \mathcal{I}' = \mathcal{I} \times \{ \text{nop} \} \times \{ \} \)
- \( \mathcal{S}' = \mathcal{S} \times (\mathcal{A} \cup \{ \text{nop} \}) \times \{0, 1\} \)
- \( \mathcal{R}(s, (s, \alpha, l), (s', (s', \alpha, l))) \)

The idea behind the construction of \( K_G \) is the introduction of intermediate states as shown in Figure 2 to convert the concurrent game structure into a turn based game structure, where the order in which the players make their moves is irrelevant: States \( (s, \text{nop}, 0) \) directly correspond to states \( s \in \mathcal{S} \) of the game structure. Every transition from a state \( s \in \mathcal{S} \) with \( s' = \delta(s, a, b) \) is split into four transitions with two intermediate states as shown in Figure 2: one that leads from state \( (s, \text{nop}, 0) \) to \( (s, a, 1) \) which means that player \( A \) made the first move. Moreover, there is a transition from \( (s, \text{nop}, 0) \) to \( (s, b, 0) \) which means that player \( B \) made the first move. From states \( (s, a, 1) \) and \( (s, b, 0) \) there are further transitions to \( (s', \text{nop}, 0) \) to complete the transition of \( G \) in \( K_G \).

States of the form \( (s, \text{nop}, 0) \) are called choice states. As each choice state \( (s, \text{nop}, 0) \) directly corresponds to the state \( s \) of the game structure, it is possible to identify sets of states of the game structure to corresponding states of the Kripke structure.

Fig. 2. From Game Structures to Kripke Structures
As an example for the translation, consider the associated Kripke structure shown in Figure 3 that is obtained from the game structure of Figure 1 (we omitted however the labels for \( \{x_A, x_B, x_N\} \)).

For the following discussion, we make implicitly use of certain invariants of the above construction which are listed in the next lemma.

**Lemma 1 (Invariants for the Construction of \( \mathcal{K}_G \)):**

Given a concurrent game structure \( \mathcal{G} = (I, S, \delta, \Gamma_A, \Gamma_B, \mathcal{L}) \) and its associated Kripke structure \( \mathcal{K}_G \) as defined in Definition 7. Then, the following holds for every transition \( \mathcal{R}(s, \alpha, t, (s', \alpha', \iota')) \):

- \( \alpha = \text{nop} \) implies \( \iota = 0 \)
- \( \alpha' = \text{nop} \) implies \( \iota' = 0 \)
- \( \alpha \in \mathcal{A} \) holds iff \( \alpha' \in \text{nop} \)
- \( \alpha = \text{nop} \) holds iff \( \alpha' \in \mathcal{A} \)
- \( \alpha \in \mathcal{A} \) and \( \iota = 1 \) imply that \( \alpha \in \Gamma_A(s) \)
- \( \alpha \in \mathcal{A} \) and \( \iota = 0 \) imply that \( \alpha \in \Gamma_B(s) \)

In particular, the above lemma implies that the transition system of \( \mathcal{K}_G \) is a bipartite graph that consists of two kinds of states: those with actions \( \alpha \in \mathcal{A} \) and those with action \( \alpha = \text{nop} \). Moreover, the action \( \text{nop} \) determines the boolean flag \( \iota = 0 \). For actions other than \( \text{nop} \), the boolean flag \( \iota \) stores the information which one of the players was responsible for the first half of the move.

We already explained the translation from game structures to Kripke structures. Next, we proceed with the translation of \( \mathcal{L}_{AT} \) to \( \mathcal{L}_\mu \):

**Definition 8 (Translating \( \mathcal{L}_{AT} \) to \( \mathcal{L}_\mu \)):** For every formula \( \varphi \in \mathcal{L}_{AT} \) over the variables \( \mathcal{V} \), we define a formula \( \mathcal{AT}_\mu(\varphi) \in \mathcal{L}_\mu \) over the variables \( \mathcal{V} \cup \{x_A, x_B, x_N\} \) inductively as follows:

- \( \mathcal{AT}_\mu(0) := 0 \)
- \( \mathcal{AT}_\mu(1) := x_N \)
- \( \mathcal{AT}_\mu(x) := x \) for variables \( x \in \mathcal{V} \)
- \( \mathcal{AT}_\mu(\lnot \varphi) := \mathcal{AT}_\mu(\varphi) \land \lnot \mathcal{AT}_\mu(\varphi) \)
- \( \mathcal{AT}_\mu(\varphi \land \psi) := \mathcal{AT}_\mu(\varphi) \land \mathcal{AT}_\mu(\psi) \)
- \( \mathcal{AT}_\mu(\varphi \lor \psi) := \mathcal{AT}_\mu(\varphi) \lor \mathcal{AT}_\mu(\psi) \)
- \( \mathcal{AT}_\mu(\Box \varphi) := \Box \mathcal{AT}_\mu(\varphi) \)
- \( \mathcal{AT}_\mu(\Box_A \varphi) := \Box \mathcal{AT}_\mu(\varphi) \)
- \( \mathcal{AT}_\mu(\Box_B \varphi) := \Box \mathcal{AT}_\mu(\varphi) \)
- \( \mathcal{AT}_\mu(\mu x. \varphi) := \mu x. \mathcal{AT}_\mu(\varphi) \)
- \( \mathcal{AT}_\mu(\nu x. \varphi) := \nu x. \mathcal{AT}_\mu(\varphi) \)

Note that according to the definition of the associated Kripke structure \( \mathcal{K}_G \), we have \( \llbracket x_N \rrbracket_{\mathcal{K}_G} = S \times \{\text{nop}\} \times \{0\} \), i.e., the choice states. Moreover, \( \llbracket x_N \rrbracket_{\mathcal{K}_G} \cap \llbracket x_A \lor x_B \rrbracket_{\mathcal{K}_G} = \{\}, \) i.e., each state where either \( x_A \) or \( x_B \) holds is not a choice state. For this reason, we could also perform the construction without the variable \( x_N \) and use \( \lnot (x_A \lor x_B) \) instead of \( x_N \) in the above translation.

**Theorem 1:** Given a concurrent game structure \( \mathcal{G} = (I, S, \delta, \Gamma_A, \Gamma_B, \mathcal{L}) \) for a set of variables \( \mathcal{V} \) and a set of actions \( \mathcal{A} \), and an arbitrary formula \( \varphi \in \mathcal{L}_{AT} \). Then, the following holds:

\[
\llbracket \mathcal{AT}_\mu(\varphi) \rrbracket_{\mathcal{K}_G} = \llbracket \varphi \rrbracket_{\mathcal{G}} \times \{\text{nop}\} \times \{0\}
\]

Hence, we can reduce the computation of \( \llbracket \varphi \rrbracket_{\mathcal{G}} \) to the computation of \( \llbracket \mathcal{AT}_\mu(\varphi) \rrbracket_{\mathcal{K}_G} \), which can be solved by ordinary \( \mu \)-calculus model checking.

Concerning the complexity, we note that every alternating time \( \mu \)-calculus formula is translated to an equivalent \( \mu \)-calculus formula of the same alternation depth and a size which is linear in the size of the original formula. The number of transitions of \( \mathcal{K}_G \) is at most multiplied by four, and the number of states of \( \mathcal{K}_G \) is \( O(|S| |\mathcal{A}|) \). It is well-known that model checking of \( \mu \)-calculus formulas \( \varphi \) of alternation depth \( \ell \) and length \( |\varphi| \) can be done on every Kripke structure \( \mathcal{K} = (I, S, \delta, \mathcal{L}) \) in time \( O\left(\left(\frac{|\varphi| |S| |\mathcal{A}|}{\ell}\right)^{\ell-1} \cdot |\delta| |\varphi|\right) \) (see [Schneider, 2003]).

We immediately obtain the following result:

**Theorem 2:** For every \( \mathcal{L}_{AT} \)-formula \( \varphi \) of alternation depth \( \ell \) and length \( |\varphi| \), and every concurrent game \( \mathcal{G} = (I, S, \delta, \Gamma_A, \Gamma_B, \mathcal{L}) \) there is an algorithm to compute the satisfying states \( \llbracket \varphi \rrbracket_{\mathcal{G}} \) in time

\[
O\left(\left(\frac{|\varphi| |S| |\mathcal{A}|}{\ell}\right)^{\ell-1} \cdot |\delta| |\varphi|\right).
\]

In [Alur et al., 2002] an algorithm is presented that modified an existing \( \mu \)-calculus model-checking algorithm to be suited for alternating time \( \mu \)-calculus model-checking. This algorithm involves the calculation of some turn-based games in each iteration step. Instead, our algorithm performs this work in one simple step at the beginning of the algorithm. The above improvement of the complexity compared to the complexity \( O\left(\left(|\varphi| |\delta|\right)^{\ell+1}\right) \) given in [Alur et al., 2002] is due to our use of improved algorithms for \( \mu \)-calculus model checking.
that can be prevented from occurring by a supervisor over the Boolean variables. It is possible to reach a marked state in developed translations. In supervisory control, a [Ramadge and Wonham, 1987] with the so far the supervisory control problem stated in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005], we only check whether there is a solution, but also to derive a suitable reduction is as follows:

• $I = \{q_0\}$
• $S = Q$
• $\Gamma_C(s) = \{\text{accept, reject}\}$
• $\Gamma_U(s) = \{\sigma \in \Sigma_c \cup \Sigma_u \mid \exists s', s' = \delta(s, \sigma)\}$
• $s' = \delta_G(s, \alpha, \sigma)$

Hence, for every uncontrolable transition $s \sigma s'$ with $\sigma \in \Sigma_u$, there are two transitions $s \xrightarrow{\text{accept, } \sigma} s'$ and $s \xrightarrow{\text{reject, } \sigma} s'$ in the game structure $G_M$, and for every controllable transition $s \sigma s'$ with $\sigma \in \Sigma_c$, there is only one transition $s \xrightarrow{\text{accept, } \sigma} s'$. The latter means that the controller $C$ has no possibility to either disable or enable the transition, while in the former case, the transition can be taken for input $\sigma$, regardless of the choice of the controller. Thus, the definition of the transition relation $\delta_G$ ensures that the controllability property is captured in our game structure: an uncontrollable input may not be prevented by player $C$, while every controllable input may be prevented (disabled) by player $C$.

To finally solve the supervisory control problem, we can proceed similarly to [Ziller and Schneider, 2003]: A state does not violate the specification, if it is

• Co-reachable, i.e., a marked state can be reached and
• good, i.e., no bad states can be reached via an uncontrollable input.
Therefore, in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005], an equation system\(^3\) is given that works on the above defined Kripke structure:

\[
\begin{align*}
\forall x_{C_0} & \quad \mu x_G & \equiv & \kappa(x_G) \land (\diamond x_{C_0} \lor x_m) \\
\forall x_G & \quad \nu & \equiv & x_a \land (\Box(x_G \land \kappa(x_{C_0})) \land \neg x_b).
\end{align*}
\]

This is an equation system of the ordinary \(\mu\)-calculus with the difference that an operator \(\kappa()\) is used to switch from one part of the Kripke structure to the other according to the following definition: \(\kappa(Q) \equiv \{(q, i) \mid (q, i) \in Q\}\). A detailed discussion on the correctness of the above equation system is found in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005].

In the following, we concentrate on the corresponding equation system on the associated game. The coreachable states can be calculated by the following formula:

\[x'_{C_0} = x'_{G} \land (\Box x_{C_0} \lor x_m),\]

where \(x'_{C_0}\) denotes the coreachable states and \(x'_{G}\) denotes the good states. For calculating the bad states, we note that a state is bad, if an uncontrollable input leads to a state which is known to be bad or not coreachable. Therefore a state is bad, if player \(E\) can enforce a transition to an already known bad state, or to a state which is known to be not coreachable. This is calculated by the following formula:

\[x'_B = 2E(x'_B \lor \neg x'_{C_0}) \land x_b.\]

By negation of the above equation, we get the good states, i.e., those states that should never be left:

\[x'_E = 2E(\neg x'_E \lor \neg x'_{C_0}) \land \neg x_b.\]

We therefore get the following equation system:

\[E' = \begin{cases}
\forall x_{C_0} & \quad \mu x'_G & \equiv & \kappa(x'_G) \land (\Box x'_{C_0} \lor x_m) \\
\forall x'_G & \quad \nu & \equiv & 2E(\neg x'_E \lor \neg x'_{C_0}) \land \neg x_b.
\end{cases}\]

The above equation system is very similar to the one that has been presented in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005]. A solution to the supervisory control problem may be given by evaluating the above equation system which is a standard problem, once we translate it further to a \(\mu\)-calculus model checking problem. To show the correctness of the presented algorithm, we will give a reduction to the equation system formulated in [Ziller and Schneider, 2003], [Ziller and Schneider, 2005]:

**Theorem 3:** Given an automaton \(\mathcal{M} = \langle Q, \Sigma, \delta, q_0, Q_m, Q_b\rangle\), its associated Kripke structure \(\mathcal{K}_{\mathcal{M}}\), and its associated game \(G_{\mathcal{M}}\), the following holds:

\[\llbracket x_{C_0} \rrbracket_{\mathcal{K}_{\mathcal{M}}} = \llbracket x'_{C_0} \rrbracket_{G_{\mathcal{M}}} \times \{1\}\]

\(^3\)Equation systems with minimality and maximality constraints are an equivalent formulation of the \(\mu\)-calculus [Schneider, 2003].

V. CONCLUSIONS

In this paper, we described a reduction from alternating time \(\mu\)-calculus model-checking problems (on concurrent game structures) to equivalent propositional \(\mu\)-calculus model-checking problems (on Kripke structures). The purpose of this reduction is that already existing model checking tools can be used to solve control problems that are formalized by the alternating time \(\mu\)-calculus. In addition, we presented a reduction of the supervisory control problem of [Ramadge and Wonham, 1987] to an equivalent alternating time \(\mu\)-calculus model-checking problem. Using the results given in this paper, we can further translate the problem to an ordinary \(\mu\)-calculus model-checking problem. This allows us to use \(\mu\)-calculus model checking algorithms to solve the supervisory control problems like the classical one introduced by Ramadge and Wonham.

REFERENCES


