A linear translation from LTL to the first-order modal μ-calculus

A linear translation from LTL to the first-order modal μ-calculus

Sjoerd Cranen Jan Friso Groote Michel Reniers
{s.cranen, j.f.groote, m.a.reniers}@tue.nl
Eindhoven University of Technology
Department of Mathematics and Computer Science
Den Dolech 2, 5612 AZ, Eindhoven, The Netherlands

Abstract

The modal μ-calculus is a very expressive temporal logic. In particular, logics such as LTL, CTL and CTL* can be translated into the modal μ-calculus, although existing translations of LTL and CTL* are at least exponential in size. We show that an existing simple first-order extension of the modal μ-calculus allows for a linear translation from LTL. Furthermore, we show that solving the translated formulae is as efficient as the best known methods to solve LTL formulae directly.

1 Introduction

Designing complex distributed systems in such a way that they behave correctly is a challenging task. One attempt to deal with this challenge is to describe the system using a behavioural modelling formalism, such as interacting automata or process algebra. Experience teaches that such descriptions are not by itself correct and therefore it is useful to establish so called behavioral properties such as absence of deadlock, safety and liveness properties that are described in a modal logic.

We use mCRL2 as a behavioural modelling formalism, which is a process algebraic behavioural specification formalism endowed with data and time, which is used extensively to model real life systems [1, 13]. Properties about the behaviour of processes are described in a modal μ-calculus enriched with data and time, the first-order modal μ-calculus [14]. The modal μ-calculus [9, 17] is an extension of Hennessy-Milner logic [16] with fixpoint operators.

By using data in mCRL2, one can specify state machines with an infinite action alphabet, giving rise to the need for a formalism that can express behavioural properties over such systems. The first-order modal μ-calculus, in which quantification over data can be used and in which fixpoint variables may have data parameters, fulfills this need and is more practical as it is syntactically less minimalistic. It is very expressive and—after some training—very pleasant to use. Indeed, over the years we have not yet encountered any behavioural property that we could not express in this formalism. Driven by these positive results, we developed theories [12, 15] and tools to verify properties in the first order modal μ-calculus with data. These tools are distributed in the open source and freely available mCRL2 toolkit [1, 13].

The purpose of this paper is to formally establish what we already experienced in practice, namely that the first order μ-calculus with data is indeed very expressive in the sense that properties formulated in other modal logics can be translated to it with only a linear growth in size.

Already in 1986, Emerson and Lei suggested that the modal μ-calculus might serve as a uniform model checking framework, and showed that CTL can be translated succinctly into the modal μ-calculus, but also noted that the only known translation from CTL* to the μ-calculus was not succinct [10]. But if the modal μ-calculus is to become a framework for model checking, it is

certainly of importance that system properties can be expressed in a formula that is roughly comparable in size with a CTL* formula.

The original translation that Emerson and Lei mentioned consisted of the composition of an unpublished translation from CTL* to PDL-Δ by Wolper, and a translation from PDL-Δ to the μ-calculus [10].

A simpler translation procedure was proposed in [8], but this translation still yields formulae doubly exponential in the size of the input formula. Only in 1996, this translation was improved upon by Bhat et al. with an algorithm that translates CTL* to an equational variant of the modal μ-calculus, only causing a single exponential blowup.

In this paper, we use a strategy similar to that of Bhat et al., but as the complexity of their construction is in the translation of the linear fragments of formulae, we focus on translating LTL. We show that a linear translation to the first-order modal μ-calculus is possible using only very simple data types.

From the context of the mCRL2 toolkit, there is also a very practical reason to have a succinct translation from LTL or CTL*. For those unfamiliar with the modal μ-calculus, or for those who favour these logics over the modal μ-calculus (and admittedly, many people do at the time of writing), a linear translation enables us to easily use available μ-calculus-checkers to verify properties formulated in these other formalisms also. To support this, we show that model checking the translated formula is as efficient as the most efficient known algorithms for model checking the original.

2 LTL and Büchi automata

In this section we introduce LTL and its semantics in terms of Kripke structures. Then, a translation from LTL to Büchi automata is discussed, which is the basis of the translation presented in section 4. Readers familiar with the subject matter may skip any part of this section, although we point out that the rest of this article relies quite heavily on the topics presented in sections 2.3 and 2.4, and therefore also on the notation used in those sections. Furthermore, we note that Kripke structures considered in this article are always deadlock free. It will become clear that this restriction is not relevant for the translation in section 4.

2.1 Kripke structures

A *Kripke structure* M is a tuple $\langle S, \rightarrow, I, AP, L\rangle$, where

- S is a set of states,
- $\rightarrow \subseteq S \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions, and
- $L : S \rightarrow 2^{AP}$ is a labeling function.

A path π is a (possibly infinite) sequence $s_0, s_1, \ldots$ of nodes from S in which every pair s_i, s_{i+1} of subsequent nodes satisfies $s_i \rightarrow s_{i+1}$.

The *size* of a Kripke structure M, denoted $|M|$, is equal to the number of states plus the number of transitions in M, i.e. $|M| = |S| + |\rightarrow|$.

A state $s \in S$ is called a *deadlock* state iff there is no $s' \in S$ such that $s \rightarrow s'$.

2.2 LTL

The following grammar defines the set of well-formed propositional LTL formulae over some set AP of atomic propositions.

$$\varphi, \psi ::= a \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \mathsf{U} \psi$$

In the above, $a \in AP$. The semantics of an LTL formula are defined on paths in a Kripke structure $\langle S, \to, I, AP, L\rangle$. If π is such a path and π^i denotes π without its i first states, then π *satisfies* an LTL formula φ, written $\pi \models \varphi$, according to the semantics below.

$$\begin{aligned}
\pi &\models a \text{ iff } s \text{ is the first state of } \pi \text{ and } a \in L(s)\\
\pi &\models \neg\varphi \text{ iff not } \pi \models \varphi\\
\pi &\models \varphi \wedge \psi \text{ iff } \pi \models \varphi \text{ and } \pi \models \psi\\
\pi &\models \bigcirc\varphi \text{ iff } \pi^1 \models \varphi\\
\pi &\models \varphi \,\mathsf{U}\, \psi \text{ iff } \exists_{j\in\mathbb{N}}(\pi^j \models \psi) \text{ and } \forall_{i<j}(\pi^i \models \varphi)
\end{aligned}$$

A Kripke structure $M = \langle S, \to, I, AP, L\rangle$ satisfies an LTL formula φ, denoted $M \models \varphi$, if and only if $\pi \models \varphi$ for all $\pi \in I \times S^*$, i.e. all paths starting in an initial state of M satisfy φ.

The following standard abbreviations are used:

$$\begin{aligned}
\varphi \,\mathsf{R}\, \psi &= \neg(\neg\varphi \,\mathsf{U}\, \neg\psi)\\
\mathbf{true} &= a \vee \neg a & \mathbf{false} &= \neg\mathbf{true}\\
\Diamond\varphi &= \mathbf{true} \,\mathsf{U}\, \varphi & \Box\varphi &= \mathbf{false} \,\mathsf{R}\, \varphi
\end{aligned}$$

In this paper we use the notion of subformulae. Every atomic proposition in an LTL formula φ is a subformula of φ. Furthermore, every operator occurring in φ, together with its argument(s), forms a subformula of φ.

2.3 Nondeterministic Büchi automata

A *nondeterministic Büchi automaton* is defined as a tuple $\langle Q, \Sigma, \delta, Q_0, F\rangle$ where

- Q is a set of states,
- Σ is a signature,
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function,
- $Q_0 \subseteq Q$ is the set of initial states and
- $F \subseteq Q$ is the acceptance set.

If the signature is not relevant, Σ and δ may be replaced by a transition relation $\rightarrowtail \subseteq Q \times Q$. A *run* in a Büchi automaton is an infinite sequence of states $q_0, q_1, \ldots$ such that $\exists_{p\in\Sigma}\, q_{i+1} \in \delta(q_i, p)$ for all $i \in \mathbb{N}$. Such a run is *accepting* if and only if it passes through an accepting state infinitely often, i.e. $\{i \,|\, q_i \in F\}$ is infinitely large.

A *generalized Büchi automaton* is a Büchi automaton that has a set of acceptance sets $\mathcal{F}$ rather than a single acceptance set F. A run $q_0, q_1, \ldots$ in such an automaton is accepting if and only if it passes through a state in every acceptance set infinitely often, i.e. for all $F \in \mathcal{F}$, $\{i \,|\, q_i \in F\}$ is infinitely large.

The *accepted language* of a (generalized) Büchi automaton is the set of all accepting runs in that automaton starting in a state from Q_0. The *product* of a Kripke structure $M = \langle S, \to, I, AP, L\rangle$ and a Büchi automaton $\mathcal{A} = \langle Q, AP, \delta, Q_0, F\rangle$ is defined as the Büchi automaton $M \otimes \mathcal{A} = \langle Q', \rightarrowtail, Q_0', F'\rangle$, where

- $Q' = S \times Q$,
- $\langle s, q\rangle \rightarrowtail \langle s', q'\rangle$ if and only if $s \to s'$ and $q' \in \delta(q, L(s))$,
- $Q_0' = \{\langle s_0, q\rangle \in Q' \,|\, s_0 \in I \wedge \exists_{q_0\in Q_0} q \in \delta(q_0, L(s_0))\}$ and
- $F' = \{\langle s, q\rangle \in Q' \,|\, q \in F\}$.

Note that the signature of $\mathcal{A}$ is the set of atomic propositions from M.

2.4 Translation from LTL to Büchi automata

Below we sketch how to create a generalized Büchi automaton $\mathcal{A}$ for an LTL formula φ of which the accepted language consists of all sentences (paths) that satisfy φ. Checking that the accepted language of $M \otimes \mathcal{A}$ is empty is then sufficient to conclude that there is no path in M that satisfies φ. The below definitions construct such a Büchi automaton for an LTL formula. These definitions are taken from and explained in full in [3].

The *closure* of an LTL formula φ is the set $\mathit{Closure}(\varphi)$ of all subformulae of φ and their negation. Double negations are omitted, i.e. formulae of the form $\neg\neg\phi$ are represented by ϕ. For example, $\mathit{Closure}(a \,\mathsf{U}\, \neg b)$ is defined to be the set $\{a, \neg a, \neg b, b, a \,\mathsf{U}\, \neg b, \neg(a \,\mathsf{U}\, \neg b)\}$.

An *LTL automaton* is a generalized Büchi automaton $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, \mathcal{F} \rangle$ belonging to an LTL formula φ over AP.

- Q is the largest subset of $2^{\mathit{Closure}(\varphi)}$ such that for all $B \in Q$ we have the following:
  - $\psi \notin B \Leftrightarrow \neg\psi \in B$
  - $\psi_1 \wedge \psi_2 \in B \Leftrightarrow \psi_1 \in B$ and $\psi_2 \in B$
  - if $\psi_1 \,\mathsf{U}\, \psi_2 \in \mathit{Closure}(\varphi)$, then
    - $\psi_2 \in B \Rightarrow \psi_1 \,\mathsf{U}\, \psi_2 \in B$
    - $\psi_1 \,\mathsf{U}\, \psi_2 \in B$ and $\psi_2 \notin B \Rightarrow \psi_1 \in B$
- $Q_0 = \{B \in Q \mid \varphi \in B\}$
- $\mathcal{F} = \{F_{\psi_1 \,\mathsf{U}\, \psi_2} \mid \psi_1 \,\mathsf{U}\, \psi_2 \in \mathit{Closure}(\varphi)\}$, where $F_{\psi_1 \,\mathsf{U}\, \psi_2} = \{B \in Q \mid \psi_1 \,\mathsf{U}\, \psi_2 \notin B \text{ or } \psi_2 \in B\}$
- $\delta(B, A) = B'$ if and only if
  - $A = B \cap AP$
  - For every $\bigcirc\psi \in \mathit{Closure}(\varphi)$: $\bigcirc\psi \in B \Leftrightarrow \psi \in B'$
  - For every $\psi_1 \,\mathsf{U}\, \psi_2 \in \mathit{Closure}(\varphi)$: $\psi_1 \,\mathsf{U}\, \psi_2 \in B \Leftrightarrow (\psi_2 \in B \vee (\psi_1 \in B \wedge \psi_1 \,\mathsf{U}\, \psi_2 \in B'))$

An LTL automaton $\mathcal{A}^G = \langle Q^G, \Sigma^G, \delta^G, Q_0^G, \mathcal{F} \rangle$ can be transformed to a normal Büchi automaton by making a copy for every acceptance set and linking those copies together cyclically. This construction is also explained in [3]. We give a precise definition here. Suppose that $k = |\mathcal{F}|$ for some k and $f : \{0, \ldots, k-1\} \to \mathcal{F}$ enumerates $\mathcal{F}$ in an arbitrary way. A regular Büchi automaton that is equivalent to $\mathcal{A}^G$ is given by $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, F \rangle$, where

$$Q = Q^G \times \{0, \ldots, k-1\} \qquad Q_0 = \{\langle q, 0\rangle \in Q \mid q \in Q_0^G\}$$
$$\Sigma = \Sigma^G \qquad F = \{\langle q, i\rangle \in Q \mid q \in f(i)\}$$

and where δ is defined as follows:

$$\langle p, \langle q', j\rangle\rangle \in \delta(\langle q, i\rangle) \text{ iff } \langle p, q'\rangle \in \delta^G(q) \text{ and } \begin{cases} i = j, & q \notin f(i) \\ (i+1) \bmod k = j & q \in f(i) \end{cases}$$

The resulting Büchi automaton accepts the same language as $\mathcal{A}^G$.

3 The first-order modal μ-calculus

In this section we introduce a first order extension of the modal μ-calculus. It is a propositional variant of the μ-calculus described in [12]. In this formalism, a notion of data is used, which we present first.

A data sort D is a set of atomic elements, associated with a semantic set $\mathbb{D}$. Operations on data sorts represent operations on their semantic sets and yield (closed) terms that represent elements

from those sets. We assume the existence of an interpretation function $[\![\_]\!]$ that maps a closed term t of sort D to an element $[\![t]\!]$ of $\mathbb{D}$.

Throughout the paper, we assume that for a sort D there is an associated set of variables $\mathcal{D}$ of sort D. A data environment $\varepsilon : \mathcal{D} \to \mathbb{D}$ is used to map variable names to elements of $\mathbb{D}$. In the obvious way, $[\![\_]\!]$ is extended to open terms, and we denote the data element associated with an open term t given a data environment ε with $[\![t]\!]^\varepsilon$.

We write $\varepsilon[d \mapsto v]$ to denote a data environment ε' for which $\varepsilon'(d') = \varepsilon(d')$ for all $d' \neq d$ and $\varepsilon'(d) = v$.

In this paper we assume the existence of a data sort B representing the booleans $\mathbb{B}$ and a sort N representing the natural numbers $\mathbb{N}$.

3.1 Syntax and semantics

We assume the existence of sort D with variables $\mathcal{D}$ that corresponds to some semantic set $\mathbb{D}$ as explained earlier. The syntax of a μ-calculus formula is defined by the following grammar:

$$\varphi ::= b \mid p \mid X(d) \mid \neg\varphi \mid \varphi \wedge \psi \mid [\cdot]\varphi \mid (\mu X(d{:}D = e)\,.\,\varphi)$$

In the above, b is a Boolean expression, p is an atomic proposition (sometimes called propositional constant), $d \in \mathcal{D}$ is a variable name, e is a data expression of sort D and X is a fixpoint variable taken from a set $\mathcal{X}$ of variable names.

The interpretation of a μ-calculus formula φ, denoted by $[\![\varphi]\!]^{\rho\varepsilon}$, is given in the context of a data environment $\varepsilon : D \to \mathbb{D}$, a predicate environment $\rho : \mathcal{X} \to (\mathbb{D} \to 2^S)$ and a Kripke structure $\langle S, \to, I, AP, L\rangle$.

$$\begin{aligned}
[\![b]\!]^{\rho\varepsilon} &\triangleq \begin{cases} S, & [\![b]\!]^\varepsilon \\ \emptyset, & \text{otherwise} \end{cases} \\
[\![p]\!]^{\rho\varepsilon} &\triangleq \{s \in S \mid p \in L(s)\}, \qquad p \in AP \\
[\![X(e)]\!]^{\rho\varepsilon} &\triangleq \rho(X)([\![e]\!]^\varepsilon) \\
[\![\neg\varphi]\!]^{\rho\varepsilon} &\triangleq S \setminus [\![\varphi]\!]^{\rho\varepsilon} \\
[\![\varphi \wedge \psi]\!]^{\rho\varepsilon} &\triangleq [\![\varphi]\!]^{\rho\varepsilon} \cap [\![\psi]\!]^{\rho\varepsilon} \\
[\![[\cdot]\varphi]\!]^{\rho\varepsilon} &\triangleq \{s \in S \mid \forall_{s' \in S}(s \to s' \Rightarrow s' \in [\![\varphi]\!]^{\rho\varepsilon})\} \\
[\![\forall_{d:D}\varphi]\!]^{\rho\varepsilon} &\triangleq \bigcap_{v \in \mathbb{D}} [\![\varphi]\!]^{\rho\varepsilon[d \mapsto v]} \\
[\![\mu X(d{:}D = e).\varphi]\!]^{\rho\varepsilon} &\triangleq (\mu\Phi)([\![e]\!]^\varepsilon)
\end{aligned}$$

Where $\Phi : (D \to 2^S) \to (D \to 2^S)$ is given as

$$\Phi \triangleq \lambda F{:}D \to 2^S.\lambda v : \mathbb{D}.[\![\varphi]\!]^{\rho[X \mapsto F]\varepsilon[d \mapsto v]}$$

It is important to note that the least fixpoint of Φ does not always exist. However, if in the above definition, φ is transformed to *positive normal form* [6], in which negation only occurs on the level of atomic propositions and in which all bound variables are distinct, then the existence of such a fixpoint is guaranteed. This claim is justified by the fact that we can define an ordering $\sqsubseteq$ on $D \to 2^S$ such that $f \sqsubseteq g$ if and only if for all $d{:}D$, $f(d) \subseteq g(d)$. Then $\langle D \to 2^S, \sqsubseteq\rangle$ is a complete lattice and because the functionals are monotonic over this lattice (see [12]), Tarski's theorem [18] can be applied to establish that the least fixpoint of Φ exists.

Furthermore, this fixpoint may be approximated by applying Φ a number of times to the infimum of the lattice (in case of a least fixpoint) or to the supremum of the lattice (for a greatest fixpoint).

We use the following standard abbreviations to denote some useful derived operators, where $\varphi[\neg X/X]$ stands for the expression φ in which every occurrence of X has been replaced by $\neg X$:

$$\begin{aligned} \varphi \vee \psi &\triangleq \neg(\neg\varphi \wedge \neg\psi) \\ \langle\cdot\rangle\varphi &\triangleq \neg[\cdot]\neg\varphi \\ \exists_{d:D}\varphi &\triangleq \neg\forall_{d:D}\neg\varphi \\ \nu X(d{:}D = e)\,.\,\varphi &\triangleq \neg\mu X(d{:}D = e)\,.\,\neg\varphi[\neg X/X] \end{aligned}$$

For readability, we allow fixpoints to be parameterised with multiple data parameters, separated by commas, rather than using a structured sort and projection functions. We also introduce one non-standard abbreviation. If P is a set of atomic propositions, then the μ-calculus formula P represents states that are labelled with exactly the labels in P:

$$P \triangleq \bigwedge_{a\in P} a \wedge \bigwedge_{a\in AP\setminus P} \neg a, \qquad P \subseteq AP$$

When a data domain Γ is used that consists of the single element γ (i.e. when data is not used), the formula $\sigma X(d:\Gamma = \gamma)\,.\,\varphi$ is abbreviated to $\sigma X\,.\,\varphi$ (for $\sigma \in \{\mu,\nu\}$).

In the following text we assume that $D = \mathbb{D}$ to make reasoning about the semantics of a formula less troublesome. In section 4.1 we show how to use standard sorts N and B to enable automatic solving of these formulae.

4 Translating LTL to the first-order μ-calculus

Translation of LTL to the standard propositional μ-calculus is not straightforward, in the sense that a simple syntactic translation procedure has not been found. Consider the following two standard translations from LTL to the μ-calculus.

$$p\,\mathsf{U}\,q \overset{\text{trans}}{=} \mu X\,.\,(p \wedge [\cdot]X) \vee q \qquad p\,\mathsf{R}\,q \overset{\text{trans}}{=} \nu X\,.\,(p \vee [\cdot]X) \wedge q$$

The above translations appear often in literature, and at first sight seem very convenient. For example, it is easy to see that a translation for $\Diamond q = \mathbf{true}\,\mathsf{U}\,q$ and $\Box q = \mathbf{false}\,\mathsf{R}\,q$ can be obtained from the above by simply substituting p for **true** and **false** respectively, yielding $\mu X\,.\,[\cdot]X \vee q$ and $\nu X\,.\,[\cdot]X \wedge q$ respectively. However, $\Diamond\Box q$ cannot be obtained by the same simple syntactic replacing, as that would result in the formula $\mu X\,.\,[\cdot]X \vee \nu Y\,.\,[\cdot]Y \wedge q$, which expresses the CTL formula $\mathsf{AFAG}\,q$. The following μ-calculus formula is the proper translation of $\Diamond\Box q$.

$$\mu X\,.\,\nu Y\,.\,[\cdot]X \vee (q \wedge [\cdot]Y)$$

We use a variant of this formula to translate LTL to the first order modal μ-calculus. Notice that this formula expresses the absence of an accepting path in a Büchi automaton if we label all non-accepting states of that automaton with q. Because a translation to Büchi automata is already known, it seems natural to use the above as a framework for our translation.

We note that the above formula can be replaced by the formula $\mu X\,.\,\nu Y\,.\,(\neg q \wedge [\cdot]X) \vee (q \wedge [\cdot]Y)$, which appears stronger at first sight. This alteration does not change the meaning of the formula, because both fixpoints must identify the same set of nodes, and therefore in particular $[\cdot]Y \Rightarrow [\cdot]X$. We use a similar construction in this paper to make the complexity analysis easier, even though we do not need this alternative formulation for our proof of correctness.

The introduction of data allows us to formulate certain properties more concisely, by exploiting repetitive structures in the formula. Consider for instance the following formula.

$$\mu X\,.\,\langle\cdot\rangle X \vee (p(0) \wedge \mu Y\,.\,\langle\cdot\rangle Y \vee (p(1) \wedge \mu Z\,.\,\langle\cdot\rangle Z \vee p(2)))$$

This formula expresses that first a state in which $p(0)$ holds is reachable, then a state in which $p(1)$ holds and finally one in which $p(2)$ holds. This formula (and any extension thereof) can also be expressed as follows:

$$\mu X(i : N = 0) \,.\, \langle\cdot\rangle X(i) \vee (p(i) \wedge \langle\cdot\rangle X(i+1)) \vee i \approx 3$$

In the above, $i \approx 3$ has the standard arithmetic meaning of 'i equals 3'. Note that the formula has collapsed the fixpoints into a single one, and that the number of boolean operators has also diminished.

Another example is the following formula, which expresses that out of the first $2k$ states visited, k states must be labelled with p:

$$\begin{aligned}\mu X(n : N = 0, m : N = 0) \,.\;\; & (n \approx k \wedge m \approx k) \vee \\ & ([\cdot]X(n+1, m) \wedge p) \vee \\ & ([\cdot]X(n, m+1) \wedge \neg p)\end{aligned}$$

The size of this formula is $O(\log k)$—because we have to write down k—where the equivalent in the normal μ-calculus would take $O(2^k)$ space (or $O(k^2)$ in the equational μ-calculus).

We now return to the problem of translating LTL to the first-order μ-calculus. We base our translation on Büchi automaton representations of LTL formulae as referred to in section 2.3. This representation is encoded into a data structure consisting of booleans and natural numbers. We express a μ-calculus property that in a sense 'synchronizes' steps in the Büchi automaton (utilizing the data structure) with steps that are made in the transition system against which the formula is checked. Keeping this synchrony intact, we can use the standard translation of $\Diamond\Box q$ to express that this Büchi automaton does not accept any path of the transition system.

Formally speaking, we assume that we are given some Büchi automaton $\mathcal{A}$, and construct a μ-calculus formula that accepts an initial state of the transition system M it is interpreted on if and only if $\mathcal{L}(M \otimes \mathcal{A}) = \emptyset$, i.e. the accepted language of the product of the Kripke structure and the Büchi automaton is empty.

Definition 1 $(T, T', \mathbf{Tr})$**.** *We define a translation function* **Tr** *that generates a μ-calculus formula from a Büchi automaton. Let* $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ *be a Büchi automaton.*

$$\mathbf{Tr}(\mathcal{A}) = \forall_{q \in Q, p \in \Sigma}\big((p \wedge \exists_{q_0 \in Q_0} q \in \delta(q_0, p)) \Rightarrow T(q)\big)$$

with $T(q)$ *defined as*

$$\begin{aligned} T(q_0) =& \mu X(q'' : Q = q_0) \,.\, T'(q'') \\ T'(q'') =& \nu Y(q : Q = q'') \,. \\ & \forall_{p, q' : q' \in \delta(q,p)}[\cdot]\Big(p \Rightarrow \big((X(q') \wedge q \in F) \vee (Y(q') \wedge q \notin F)\big)\Big)\end{aligned}$$

In the above, note that the quantifier selects those q' and p that form a single step in the Büchi automaton from state q. The implication $(p \Rightarrow \ldots)$ ensures that the required property is only checked along paths realising such steps. In this manner, the quantifier and implication realise the aforementioned synchrony. In effect the formula $\mu X \,.\, \nu Y \,.\, [\cdot]X \vee (q \notin F \wedge [\cdot]Y)$ is checked on the paths of the Büchi automaton that have a corresponding path in the Kripke structure (i.e. exactly the paths in $M \otimes \mathcal{A}$).

Before we look at the properties of this translation, we introduce a notational convention that will make the syntax less hairy, and we introduce definitions for $\hat{X}^n$ and $\hat{Y}^n$, which are used in the proofs of lemmata 1 and 2.

The semantics $[\![T(q_0)]\!]^{\rho\varepsilon}$ of this formula is $(\mu\Phi)(q_0)$, where

$$\Phi = \lambda\hat{X} : Q \to 2^S . \lambda\hat{q} : Q . [\![T'(q'')]\!]^{\rho[X \mapsto \hat{X}]\varepsilon[q'' \mapsto \hat{q}]}$$

As explained in section 3, we can calculate this fixpoint by starting with an initial approximation $\hat{X}^0$ that is the minimal element of the lattice $\langle Q \to 2^S, \sqsubseteq \rangle$, and then choosing the next approximation $\hat{X}^{m+1} = \Phi(\hat{X}^m)$.

Because $\hat{X}^0 = \lambda \hat{q} : Q.\emptyset$ can be written as $\lambda \hat{q} : Q.[\![\mathbf{false}]\!]^{\rho[X \mapsto \hat{X}^{-1}]\varepsilon[q'' \mapsto \hat{q}]}$ (regardless of how we define $\hat{X}^{-1}$), every approximation $\hat{X}^{m+1}$ can be rewritten to the form $\lambda \hat{q} : Q.[\![\phi]\!]^{\rho[X \mapsto \hat{X}^m]\varepsilon[q'' \mapsto \hat{q}]}$. To increase legibility, we omit the interpretation function and abbreviate this to $\lambda q'' : Q.\phi[\hat{X}^m/X]$, where $\phi[\hat{X}^m/X]$ is the formula ϕ with all occurrences of X replaced by $\hat{X}^m$.

We can now inductively define the approximations for T as follows:

$$\begin{array}{lcl} \hat{X}^0 & = & \lambda q'' : Q.\mathbf{false} \\ \hat{X}^{m+1} & = & \lambda q'' : Q.\nu Y(q : Q = q'')\,. \\ & & \quad \forall_{p,q' : q' \in \delta(q,p)}[\cdot]\Big(p \Rightarrow \big((\hat{X}^m(q') \wedge q \in F) \vee (Y(q') \wedge q \notin F)\big)\Big) \end{array}$$

Note that T is equal to $\hat{X}^\alpha$ for some sufficiently large α. Similarly, we can approximate T', given an approximation $\hat{X}^m$ of T:

$$\begin{array}{lcl} \hat{Y}^0_m & = & \lambda q : Q.\mathbf{true} \\ \hat{Y}^{n+1}_m & = & \lambda q : Q.\forall_{p,q' : q' \in \delta(q,p)}[\cdot]\Big(p \Rightarrow \big((\hat{X}^m(q') \wedge q \in F) \vee (\hat{Y}^n_m(q') \wedge q \notin F)\big)\Big) \end{array}$$

Using these definitions, we show the relationship between the μ-calculus formula of definition 1 and the Büchi automaton it was generated from.

Lemma 1 ($\Rightarrow$)**.** *Given are a Büchi automaton $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, F \rangle$ and a Kripke structure $M = \langle S, \to, I, AP, L \rangle$. If $s_0 \models T(q_0)$, then there is no accepting run in $M \otimes \mathcal{A}$ from state $\langle s_0, q_0 \rangle$.*

Proof. We give a proof by contraposition. Suppose that there is a run $\pi = \langle s_0, q_0 \rangle, \langle s_1, q_1 \rangle, \ldots$ in $M \otimes \mathcal{A}$ that is accepting. We prove that $s_0 \not\models T(q_0)$ by showing that $\forall_{m \in \mathbb{N}} \forall_{i \in \mathbb{N}}\ s_i \not\models \hat{X}^m(q_i)$ by using induction on m. It then follows that $\forall_{i \in \mathbb{N}}\ s_i \not\models T(q_i)$. The induction hypothesis is the following.

$$\forall_{i \in \mathbb{N}}\ s_i \not\models \hat{X}^m(q_i) \tag{1}$$

For $m = 0$, this trivially holds. For $m > 0$ we have to show that the greatest fixpoint T' of Y does not contain any of these s_i either. We show that there is some n for which $\forall_{i \in \mathbb{N}}\ s_i \not\models \hat{Y}^n_m(q_i)$ and therefore $\forall_{i \in \mathbb{N}}\ s_i \not\models T'(q_i)$.

Observe that, because of the definition of the transition function of $M \otimes \mathcal{A}$, $s_i \to s_{i+1}$ and $q_{i+1} \in \delta(q_i, L(s_{i+1}))$ for all $i \in \mathbb{N}$. The definition of $\hat{Y}^{n+1}_m$ therefore implies that if $s_i \models \hat{Y}^{n+1}_m(q_i)$, then we must also have $s_{i+1} \models L(s_{i+1}) \Rightarrow ((\hat{X}^m(q_{i+1}) \wedge q_i \in F) \vee (\hat{Y}^n_m(q_{i+1}) \wedge q_i \notin F))$. Because by definition $s_{i+1} \models L(s_{i+1})$, and because of (1), we have an even stronger implication:

$$\text{For } n \in \mathbb{N}, \text{ if } s_i \models \hat{Y}^{n+1}_m(q_i), \text{ then } s_{i+1} \models \hat{Y}^n_m(q_{i+1}) \text{ and } q_i \notin F \tag{2}$$

Suppose that $q_i \in F$ for some i, then $s_{i+1} \not\models \hat{Y}^n_m(q_{i+1}) \wedge (q_i \notin F)$ for any n, and it follows immediately that also $s_i \not\models \hat{Y}^1_m(q_i)$. Now suppose $q_i \notin F$. Because π is an accepting run, there must be some $k \in \mathbb{N}$ for which $q_{i+k} \in F$, in which case $s_{i+k} \not\models \hat{Y}^n_m(q_{i+k})$ for any n. In particular this holds for $n = 1$ and therefore we have, by transitivity of implication (2), $s_i \not\models \hat{Y}^{k+1}_m(q_i)$. □

Lemma 2 ($\Leftarrow$)**.** *Given are a Büchi automaton $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, F \rangle$ and a Kripke structure $M = \langle S, \to, I, AP, L \rangle$. If $s_0 \not\models T(q_0)$, then there is an accepting run in $M \otimes \mathcal{A}$ from state $\langle s_0, q_0 \rangle$.*

Proof. Let $s_i \in S$ and assume that $s_i \not\models T(q_i)$. Let $\rightarrowtail$ be the transition relation of $M \otimes \mathcal{A}$. We show that $s_i \not\models T(q_i) \Rightarrow \exists_{n \in \mathbb{N}}\ n > 0 \wedge G(s_i, q_i, n)$, where G is defined as follows.

$$G(s, q, n) = \begin{cases} q \in F \wedge s \not\models T(q), & n = 0 \\ \exists_{s' \in S, q' \in Q}\ \langle s, q \rangle \rightarrowtail \langle s', q' \rangle \wedge G(s', q', n-1), & n > 0 \end{cases}$$

In other words, either $\langle s_i, q_i \rangle$ is an accepting state of $M \otimes \mathcal{A}$ or there is a finite path from $\langle s_i, q_i \rangle$ to an accepting state of $M \otimes \mathcal{A}$ that again does not satisfy $T(q_i)$.

Choose an arbitrary $m \in \mathbb{N}$. Notice that because $s_i \not\models T(q_i)$, also $s_i \not\models \hat{X}^m(q_i)$. For this reason there must be some $k > 0$ for which $s_i \not\models \hat{Y}_m^k(q_i)$. Filling in the definition of $\hat{Y}_m^k(q_i)$, we get

$$s_i \not\models \forall_{p,q':q' \in \delta(q,p)}[\cdot]\Big(p \Rightarrow \big((\hat{X}^m(q') \wedge q_i \in F) \vee (\hat{Y}_m^{k-1}(q') \wedge q_i \notin F)\big)\Big)$$

In particular, this means that there exist an s_{i+1} such that $s_i \to s_{i+1}$ and q_{i+1} such that $q_{i+1} \in \delta(q, L(s_{i+1}))$ for which the following holds:

$$s_{i+1} \models L(s_{i+1}) \wedge \neg\Big((\hat{X}^m(q_{i+1}) \wedge q_i \in F) \vee (\hat{Y}_m^{k-1}(q_{i+1}) \wedge q_i \notin F)\Big) \tag{3}$$

Because m was chosen arbitrarily, this also implies that $s_{i+1} \not\models T(q_{i+1})$. We therefore prove that $G(s_i, q_i, k-1)$. As the only assumption on s_i was that $s_i \not\models T(q_i)$, we then also have $G(s_{i+1}, q_{i+1}, k'-1)$ for some $k' > 0$. But then, by definition of G and the fact that $\langle s_i, q_i\rangle \rightarrowtail \langle s_{i+1}, q_{i+1}\rangle$, we also have $G(s_i, q_i, k'')$ for $k'' > 0$.

Suppose $q_i \in F$. Then $G(s_i, q_i, 0)$ holds trivially. Now suppose $q_i \notin F$. We prove by induction on k that $s_i \not\models \hat{Y}_m^k(q_i) \Rightarrow G(s_i, q_i, k-1)$. For the base case we fill in $k = 1$ in (3) and obtain $q_i \in F$. We had already assumed that $s_i \not\models T(q_i)$, so we have found that $G(s_i, q_i, k-1)$ holds.

In the case that $k = n+1$, the assumption that $q_i \notin F$ in combination with (3) yields $s_{i+1} \not\models \hat{Y}_m^n(q_{i+1})$. The induction hypothesis then yields $G(s_{i+1}, q_{i+1}, n-1)$, and because $\langle s_i, q_i\rangle \rightarrowtail \langle s_{i+1}, q_{i+1}\rangle$ also $G(s_i, q_i, k-1)$. □

Theorem 1. *Let $\mathcal{A} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ be a Büchi automaton and $M = \langle S, \to, I, AP, L\rangle$ be a Kripke structure. The language of their product is empty, i.e. $\mathcal{L}(M \otimes \mathcal{A}) = \emptyset$ if and only if $M \models \mathbf{Tr}(\mathcal{A})$.*

Proof. Note that $s_0 \models p \wedge \exists_{q_0 \in Q_0} q \in \delta(q_0, p)$ for exactly those $s_0 \in S$ and $q \in Q$ for which $\langle s_0, q\rangle$ is in the set of initial states of $M \otimes \mathcal{A}$. The language of $M \otimes \mathcal{A}$ is empty if and only if there is no accepting run starting in any of these states. $\mathbf{Tr}(\mathcal{A})$ demands that in these states $T(q)$ must hold. Lemmas 1 and 2 show that $s_0 \models T(q)$ is true if and only if there is no accepting run in $M \otimes \mathcal{A}$ starting in $\langle s_0, q\rangle$. □

Finally, we note that the given translation is also correct for Kripke structures with deadlock states, since paths ending in deadlock are never accepting and deadlock states make the $[\cdot]$ modality in the μ-calculus formula hold trivially.

4.1 Data specifications

We have formulated our translation in such a way that it uses a Büchi automaton directly (Q, δ, Q_0 and F occur in our formula). In order to use existing techniques [7] to be able to automatically check the μ-calculus formula against a Kripke structure, and also to exploit the structured manner in which we can build a Büchi automaton from an LTL formula, we encode Q into a datatype which consists of only Booleans and natural numbers.

Let φ be an LTL formula consisting of subformulae Ψ, and let $\mathcal{A}$ be a Büchi automaton constructed for φ as described in section 2.4. Because of the way $\mathcal{A}$ was constructed, it may be described using only Booleans and natural numbers. Recall that a state in $\mathcal{A}$ is represented by a set of subformulae $q \in 2^\Psi$ and a counter $c \in \{0, \ldots, k-1\}$, with k the number of until operators in φ.

Now we use the fact that, given some mapping from Ψ to $\{0, \ldots, |\Psi|\}$, we can substitute 2^Ψ by the isomorphic domain $\mathbb{B}^{|\Psi|}$. The counter can be represented by a value from $\mathbb{N}$, and so we may represent states by an element from $\mathbb{B}^n \times \mathbb{N}$.

By $P(q)$ we denote the set of atomic propositions of which the corresponding bit in q is set, i.e. if q represents a set $\Phi \subseteq 2^\Psi$, then $P(q) = \Phi \cap AP$.

We proceed by giving an encoding of the Büchi automaton corresponding to an LTL formula φ over atomic propositions AP, consisting of arbitrarily ordered subformulae $\psi_0 \ldots \psi_n$. We fix a

datatype $D = B^n \times N$ and we define the following four mappings:

$$\mathsf{inQ} : D \to B \qquad \mathsf{inQ_0} : D \to B$$
$$\mathsf{inF} : D \to B \qquad \mathsf{trans} : D \times D \to B$$

Intuitively, these mappings represent predicates on states from the Büchi automaton. For instance, if a data element $d{:}D$ represents some state q in a Büchi automaton with states Q and acceptance set F, then $\mathsf{inF}(d)$ will have the same truth value as the predicate $q \in F$. The inQ mapping is needed to identify those elements in D that represent a valid state in the Büchi automaton (there are subsets of the closure of φ that are not in Q, see section 2.4).

We now give the definitions of these mappings. Let $\mathfrak{U}$ be a set of indices, and let L and R be mappings from indices to indices. We have $i \in \mathfrak{U}$, $L(i) = j$ and $R(i) = k$ if and only if $\psi_i = \psi_j \,\mathsf{U}\, \psi_k$ for some i, j and k. Let $U : \mathbb{N} \to \mathfrak{U}$ enumerate $\mathfrak{U}$ in an arbitrary way.

Similarly, let $\mathfrak{X}$ be another set of indices, such that $i \in \mathfrak{X}$ and $R(i) = j$ if and only if $\psi_i = \bigcirc\psi_j$ for some i and j.

$$\begin{aligned}
\mathsf{inQ}(\langle b_0, \ldots, b_n, c\rangle) &= \bigwedge_{i\in\mathfrak{U}} (b_{R(i)} \Rightarrow b_i) \wedge (b_i \Rightarrow (b_{L(i)} \vee b_{R(i)})) \\
\mathsf{inQ_0}(\langle b_0, \ldots, b_n, c\rangle) &= \mathsf{inQ}(\langle b_0, \ldots, b_n, c\rangle) \wedge b_0 \\
\mathsf{inF}(\langle b_0, \ldots, b_n, c\rangle) &= \mathsf{inQ}(\langle b_0, \ldots, b_n, c\rangle) \wedge (\neg b_{U(c)} \vee b_{R(U(c))}) \\
\mathsf{trans}(\ \langle b_0, \ldots, b_n, c\rangle, \ &= \bigwedge_{i\in\mathfrak{X}} (b_i \Leftrightarrow b'_{R(i)}) \wedge \\
\quad \langle b'_0, \ldots, b'_n, c'\rangle) \ &\quad \bigwedge_{i\in\mathfrak{U}} (b_i \Leftrightarrow (b_{R(i)} \vee (b_{L(i)} \wedge b'_i))) \wedge \\
&\quad \mathsf{inF}(\langle b_0, \ldots, b_n, c\rangle) \Leftrightarrow (c' = (c+1) \bmod |\mathfrak{U}|)
\end{aligned}$$

Clearly, this specification is linear in the number of subformulae of φ.[1]

We use the fact that $q' \in \delta(\langle b_0, \ldots, b_n, c\rangle, p)$ implies that both $\mathsf{trans}(\langle b_0, \ldots, b_n, c\rangle, q')$ and $p = P(\langle b_0, \ldots, b_n, c\rangle) = \{a \in AP \mid b_{I(a)}\}$, where I maps a subformula ψ_i to its index i. The μ-calculus formula in definition 1 can be rewritten to the following formula using only quantifiers over D and using the previously defined mappings (i.e. $q \in F$ is replaced by $\mathsf{inF}(q)$, $q' \in \delta(q, p)$ by $\mathsf{trans}(\langle b_0, \ldots, b_n, c\rangle, q')$ while replacing all occurrences of p by $P(q)$, etc.).

$$\begin{aligned}
\psi = &\forall_{q_0\in D}\big(\mathsf{inQ}(q_0) \wedge \exists_{q'_0\in D}(\mathsf{inQ_0}(q'_0) \wedge P(q'_0) \wedge \mathsf{trans}(q'_0, q_0))\big) \Rightarrow \\
&\mu X(q'' : D = q_0)\,.\,\nu Y(q : D = q'')\,. \\
&\forall_{q'\in D}(\mathsf{inQ}(q') \wedge \mathsf{trans}(q, q')) \Rightarrow [\cdot]\Big(P(q) \Rightarrow \big(\ (X(q') \wedge \mathsf{inF}(q)) \vee \\
&\qquad\qquad (Y(q') \wedge \neg\mathsf{inF}(q))\big)\Big)
\end{aligned}$$

The formula may grow to a size linear in n due to the expansion of $P(q)$ to $\bigwedge_{a\in AP}(b_{I(a)} \Leftrightarrow a)$.

4.2 Complexity

We have given a translation from an LTL formula to a first-order μ-calculus formula over a data structure. We now show that model checking the resulting formula against a Kripke structure has the same time complexity as other LTL model checking methods. In particular, we establish the same worst-case time complexity as Bhat et al. [5].

Theorem 2. *Let ψ be the μ-calculus formula that is the result of the above translation for an LTL formula φ. Verifying ψ on a Kripke structure M can be done in $O(|M|) \cdot 2^{O(|\varphi|)}$ time.*

[1] We assume that the size of an identifier for a subformula can be seen as a constant. If the number of subformulae is extremely high, it would be fairer to say that this description has size $n \log(n)$.

Proof. Using the Bekič principle [4] and the fact that we can transform a μ-calculus formula into an equational equivalent [2], we transform the first-order modal μ-calculus formula into the equational modal μ-calculus. Because D is a finite data type, we can transform—in linear time—the formula to the following system, where $N = |D|$ and $h : \{1, \ldots, N\} \to D$ enumerates D. The inverse mapping is denoted by h^{-1}.

$$
\begin{aligned}
\mu X_\psi &= \bigwedge_{q_0 \in D} \left(\mathsf{inQ}(q_0) \wedge \bigvee_{q_0' \in D} (\mathsf{inQ_0}(q_0') \wedge P(q_0') \wedge \mathsf{trans}(q_0', q_0)) \right) \Rightarrow X_{h^{-1}(q_0)} \\
\mu X_0 &= Y_0 \\
&\vdots \\
\mu X_N &= Y_N \\
\nu Y_0 &= \bigwedge_{q' \in D} (\mathsf{inQ}(q') \wedge \mathsf{trans}(h(0), q')) \Rightarrow \\
&\qquad [\cdot] \left(P(h(0)) \Rightarrow (X_{h^{-1}(q')} \wedge \mathsf{inF}(h(0))) \vee (Y_{h^{-1}(q')} \wedge \neg \mathsf{inF}(h(0))) \right) \\
&\vdots \\
\nu Y_N &= \bigwedge_{q' \in D} (\mathsf{inQ}(q') \wedge \mathsf{trans}(h(N), q')) \Rightarrow \\
&\qquad [\cdot] \left(P(h(N)) \Rightarrow (X_{h^{-1}(q')} \wedge \mathsf{inF}(h(N))) \vee (Y_{h^{-1}(q')} \wedge \neg \mathsf{inF}(h(N))) \right)
\end{aligned}
$$

The structure of the above expression becomes more apparent after computing the truth values of all expressions that are only dependent on data terms. This computation takes $O(|D|^2 \cdot \log |D|)$ time, as $\mathsf{trans}(q, q')$ has to be calculated for every pair q, q' and every such calculation costs $\log |D|$ time. Note that because $|D| = 2^{|\varphi|}$, the time complexity for this computation is also $2^{O(|\varphi|)}$. After computation, the system can be written as follows, where sets $S, S', S_0, \ldots, S_N$ contain indices between 0 and N for which certain data expressions evaluated to **true**.

$$
\begin{aligned}
\mu X_\psi &= \bigwedge_{i \in S} \Big(\bigvee_{j \in S'} P(h(j)) \Big) \Rightarrow X_i \\
\mu X_0 &= Y_0 \\
&\vdots \\
\mu X_N &= Y_N \\
\nu Y_0 &= \bigwedge_{i \in S_0} [\cdot] \left(P(h(0)) \Rightarrow R_i \right), && R \in \{X, Y\} \\
&\vdots \\
\nu Y_N &= \bigwedge_{i \in S_N} [\cdot] \left(P(h(N)) \Rightarrow R_i \right), && R \in \{X, Y\}
\end{aligned}
$$

From this system it is easy to see that when it is checked against a Kripke structure M, the disjuncts (including implications) disappear, as all $P(\ldots)$ terms are substituted by a truth value. Furthermore, the $[\cdot]$ operators change into conjuncts.

The solution of the resulting system $\mathcal{E}$ can therefore be found as the solution of a conjunctive boolean equation system [11]. Such a solution can be found in $O(|\mathcal{E}| \cdot |M|)$ time. The complexity of checking an LTL formula φ through the first-order modal μ-calculus is therefore $O(|M|) \cdot 2^{O(|\varphi|)}$, as the size of D is exponential in the size of the LTL formula. □

5 Conclusion

In this paper we presented a translation from LTL formulae to first order μ-calculus formulae. By using this specific variant of the μ-calculus, we are able to give a translation that is succinct, but that does not introduce performance penalties when checking the formula against a Kripke structure. Indeed, the time complexity of LTL model checking via the first order modal μ-calculus is no worse than that of LTL model checking using the best existing direct method.

It is expected that the translation of LTL formulae can be lifted to a translation of CTL* formulae much in the same manner as described in [5]. The intuition here is that in a given CTL* formula, the path quantifiers are state formulae and can in a sense be treated in the same way as atomic propositions. This leads to an approach where for every path quantifier in a CTL* formula, a data structure and a μ-calculus formula are generated, which are composed using the structure of the CTL* formula.

Acknowledgements We would like to thank Tim Willemse for many valuable comments and discussions.

References

[1] mCRL2 web site. http://www.mcrl2.org.

[2] A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in logic and the foundations of mathematics*. North-Holland, 2001.

[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[4] H. Bekič. Definable operation in general algebras, and the theory of automata and flowcharts. In *Programming Languages and Their Definition*, volume 177 of *LNCS*, pages 30–55. Springer, 1984.

[5] G. Bhat and R. Cleaveland. Efficient model checking via the equational μ-calculus. In *Logic in Computer Science (LICS '96)*, pages 304–312. IEEE Computer Society, 1996.

[6] J. Bradfield and C. Stirling. Modal μ-calculi. *Handbook of Modal Logic*, pages 721–756, 2006.

[7] A. van Dam, B. Ploeger, and T.A.C. Willemse. Instantiation for parameterised boolean equation systems. In *Theoretical Aspects of Computing (ICTAC 2008)*, volume 5160 of *LNCS*, pages 440–454. Springer, 2008.

[8] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. In *17th Colloquium on Trees in Algebra and Programming (CAAP '92)*, volume 581 of *LNCS*, pages 145–164. Springer, 1992.

[9] E. Allen Emerson. Model checking and the mu-calculus. In *DIMACS Series in Discrete Mathematics*, pages 185–214. American Mathematical Society, 1997.

[10] E.A. Emerson and C.L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Logic in Computer Science (LICS '86)*, pages 267–278. IEEE Computer Society Press, 1986.

[11] J.F. Groote and M. Keinänen. A sub-quadratic algorithm for conjunctive and disjunctive boolean equation systems. In *Theoretical Aspects of Computing (ICTAC 2005)*, volume 3722 of *LNCS*, pages 532–545. Springer, 2005.

[12] J.F. Groote and R. Mateescu. Verification of temporal properties of processes in a setting with data. In *Algebraic Methodology and Software Technology*, volume 1548 of *LNCS*, pages 74–90. Springer, 1998.

[13] J.F. Groote, A.H.J. Mathijssen, M.A. Reniers, Y.S. Usenko, and M.J. van Weerdenburg. Analysis of distributed systems with mCRL2. In *M. Alexander, W. Gardner, editors, Process Algebra for Parallel and Distributed Processing*, pages 99–128. Chapman Hall, 2009.

[14] J.F. Groote and T.A.C. Willemse. Model-checking processes with data. *Science of Computer Programming*, 56(3):251–273, 2005.

[15] J.F. Groote and T.A.C. Willemse. Parameterised boolean equation systems. *Theoretical Computer Science*, 343(3):332–369, 2005.

[16] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *Automata, Languages and Programming*, volume 85 of *LNCS*, pages 299–309. Springer, 1980.

[17] D. Kozen. Results on the propositional μ-calculus. In *Automata, Languages and Programming*, volume 140 of *LNCS*, pages 348–359, 1982.

[18] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.