A Temporal Fixpoint Calculus

Extended Abstract

Moshe Y. Vardi
IBM Almaden Research Center

Abstract

Two distinct extensions of temporal logic has been recently advocated in the literature. The first extension is the addition of fixpoint operators that enable the logic to make assertions about arbitrary regular events. The second extension is the addition of past temporal connectives that enables the logic to refer directly to the history of the computation. Both extensions are motivated by the desire to adapt temporal logic to modular, i.e., compositional, verification (as opposed to global verification). We introduce and study here the logic \( \mu TL \), which is the extension of temporal logic by fixpoint operators and past temporal connectives. We extend the automata-theoretic paradigm to \( \mu TL \). That is, we show how, given an \( \mu TL \) formula \( \varphi \), we can produce a finite-state Büchi automaton \( A_\varphi \), whose size is at most exponentially bigger than the size of \( \varphi \), such that \( A_\varphi \) accepts precisely the computations that satisfy \( \varphi \). The result has immediate applications, e.g., an optimal decision procedure and a model-checking algorithm for \( \mu TL \).

1 Introduction

Since the proposal by Pnueli in 1977 [Pn77] to apply Temporal Logic (TL) to the specification and verification of concurrent programs, the role of TL as a feasible approach to that task has been widely accepted. Over the last decade an extensive research has been carried out concerning both practical and theoretical aspects of using TL to specify and verify concurrent programs (see surveys in [EP87,Pn86]). One of the conclusions of this research is that the standard TL, which consists of the temporal connectives nexttime and until is not expressive enough for its task.

The first to complain about the expressible power of TL was Wolper [Wo83] who observed that temporal logic cannot express regular events (in fact, TL can express
precisely the star-free regular events [To81]). As was shown later [LPZ85], this makes TL inadequate for modular verification (as opposed to global verification). As a remedy Wolper suggested extending TL with temporal connectives that correspond to regular grammars. The extended TL, is called, naturally enough, ETL. ETL is indeed expressive enough [LPZ85, Pn86, Wo83]; the weakness, however, of this approach is that to get the full expressive power of ETL we need infinitely many temporal connectives!

An alternative approach, advocated originally by Emerson and Clarke [EC80] and then by Barringer et al. [BKP85,BKP86], is to extend TL with maximal and minimal fixpoint quantifiers, as was done for dynamic logic in [Ko83,Pr82]. The resulting logic, called $\nu TL$, is expressively equivalent to ETL, but unlike ETL it needs only one temporal connective (the nexttime connective). Furthermore, the requirement for fixpoint quantifiers arises naturally when we attempt to specify the temporal semantics of recursive procedures [Ba86]. $\nu TL$ has been studied in [BB86], where a decision procedure for satisfiability that runs in exponential time and space is given. (In contrast, for all variants of ETL there are decision procedures that run in polynomial space [SC85, SVW83, WVS85].)

Another problem with the standard TL is the lack of temporal connectives to describe past events. While such connectives are present in works on temporal logic by philosophers (e.g., [Pr51,RU71]), they have been purged by computer scientists motivated by a strive for minimality, following the observation in [GPS80] that in applications with infinite future but finite past, past connectives do not add expressive power. More recently, however, arguments were made for the restoration of the past in temporal logic. The first argument is that while past temporal connectives do not add any expressive power the price for eliminating them can be high. Many natural statements in program specification are much easier to express using past connectives [KVR83]. In fact, the best known procedure to eliminate past connectives may cause a nonelementary blow-up of the considered formulas [LPZ85].

A more important motivation for the restoration of the past is again the use of temporal logic in modular verification. In global verification one uses temporal formulas that refer to locations in the program text [MP83]. This is absolutely verboten in modular verification, since in specifying a module one can refer only to its external behavior. Since we cannot now refer to program location we have instead to refer to the history of the computation, and we can do that very easily with past connectives [BK85]. Lichtenstein et al. [LPZ85] studied the extension of TL with the past connectives. In particular they supplied a decision procedure for satisfiability that runs in polynomial space.

We can summarize the above arguments for the extension of TL with a quote from [Pn86]: “In order to perform compositional specification and verification, it is convenient to use the past operators but necessary to have the full power of ETL.”

Our goal in this paper is to study a temporal logic that has past temporal connectives and the expressive power of ETL. To this end we introduce a temporal logic, called $\mu TL$, which is the extension of $\nu TL$ with past connectives. We wish to extend the
automata-theoretic paradigm advocated in [VW85] to \( \mu TL \). According to that paradigm a temporal logic becomes more easily applicable once its connection to automata theory is established. Indeed for ETL and for TL with past connectives there are procedures that, given a temporal formula \( \varphi \), produce a finite-state Büchi automaton \( A_\varphi \) (which is an automaton on infinite words), whose size is at most exponentially bigger than the size of \( \varphi \), such that \( A_\varphi \) accepts precisely the computations that satisfy \( \varphi \) [SVW85, VW86, VW883]. The beauty of this result is that for many application \( A_\varphi \) is all that necessary. First, one can check whether \( A_\varphi \) accepts some word to determine whether \( \varphi \) is satisfiable [SVW85, VW883]. This gives us a decision procedure for satisfiability that runs in polynomial space. Secondly, one can view \( A_\varphi \) as a synthesized program that implements the specification \( \varphi \) [MW84]. Thirdly, given a finite-state program \( P \), one can combine \( P \) with \( A_\varphi \) and then check whether \( P \) satisfies \( \varphi \) by a reachability analysis of the combined program [Va85, VW86]. Finally, given a program \( P \) one can use \( A_\varphi \) to derive proof obligations that when checked will guarantee the correctness of \( P \) with respect to \( \varphi \) [AS85, AS87, MP87, Va87].

We first show that the automata-theoretic paradigm can be extended to \( \nu TL \). This result requires two basic techniques. The first technique is borrowed from the analysis by Streett and Emerson of the \( \mu \)-calculus [SE84]. A least fixpoint formula \( \mu X.\varphi(X) \) (resp., a greatest fixpoint formula \( \nu X.\varphi(X) \)) holds in a point in a temporal structure only if the reiterated formula \( \varphi(\mu X.\varphi(X)) \) (resp., \( \varphi(\nu X.\varphi(X)) \)) also holds at that point. Streett and Emerson formalized this notion of derivation between fixpoint formulas. Furthermore, they have shown that for least fixpoint formulas to hold it is required that the derivation relation be well-founded. We show that in our framework that means that a finite-state Büchi automaton can detect non-satisfaction of fixpoint formulas, by checking the non-well-foundedness of the derivation relation.

We need, however, a Büchi automaton that will detect satisfaction of formulas. To obtain that we use the technique of Sistla et al. for the exponential complementation of Büchi automata [SVW85]. We note that we are dealing here with nondeterministic automata on infinite words for which the classical subset construction does not work. In fact, unlike automata on finite words, nondeterministic Büchi automata, cannot be determinized [Ch75].

When we try to extend the above framework to \( \mu TL \) we encounter another difficulty. Since \( \mu TL \) includes past connectives, the derivation relation of fixpoint formulas involves both future and past points. Thus, in order for a finite-state automaton to check non-well-foundedness, it has to be a 2-way automaton! 2-way automata over finite words have been studied and shown not to have any expressive power beyond 1-way automata [RS59, Sh59]. We define and study here 2-way Büchi automata and show that they are not more expressive than 1-way Büchi automata. In fact, we show that given a 2-way Büchi automaton we can make it into a 1-way Büchi automaton and complement it with the total cost of one exponential blow-up. This suffices to establish the automata-theoretic

---

\(^1\)That is, the formula obtained by substituting \( \mu X.\varphi(X) \) for \( X \) in \( \varphi \).
result for \( \mu TL \).

# 2 Fixpoint Temporal Calculus

## 2.1 Syntax and Semantics

We focus here on the propositional version of the logic. Our results are easily extendible to the first-order case provided all individual variables are local as in [Pr81].

**Formulas of \( \mu TL \) are:**

1. Propositional constants from a set \( \mathcal{P} \),
2. Propositional variables from a set \( \mathcal{V} \),
3. \( \neg \varphi, \varphi \land \psi, \text{ and } \varphi \lor \psi \), where \( \varphi \) and \( \psi \) are formulas,
4. \( \oplus \varphi, \ominus \varphi, \text{ and } \ominus \varphi \), where \( \varphi \) is a formula,
5. \( \mu X.\varphi(X) \) and \( \nu X.\varphi(X) \), where \( X \) is a propositional variables in \( \mathcal{V} \) and \( \varphi \) is a formula where \( X \) occurs positively (i.e., under an even number of negations).

A **sentence** of \( \mu TL \) is a formula with no free propositional variables.

Intuitively, \( \oplus \varphi \) (resp. \( \ominus \varphi \)) says that \( \varphi \) holds in the *next* (resp. *previous*) time moment. The formula \( \ominus \varphi \) says that \( \varphi \) holds in the previous time point if such a point exist (it may not exist in the initial point). The formula \( \mu X.\varphi(X) \) (resp. \( \nu X.\varphi(X) \)) denotes the *least* (resp. *greatest*) fixpoint solution to the equation \( X \equiv \varphi(X) \). We now give the formal semantics.

A **state** is a member of \( 2^\mathcal{P} \), i.e., an assignment of truth values to the propositional constants. A **structure** is a member of \( (2^\mathcal{P})^\omega \), i.e., an infinite sequence of states. (For technical convenience we choose to view all computations as infinite. Finite computations can be viewed as looping forever in their terminal state.) To interpret formulas we also need to deal with the propositional variables. A **pseudo-state** is a member of \( 2^\mathcal{V} \), i.e., an assignment of truth values to the propositional variables. A **pseudo-structure** is a member of \( (2^\mathcal{V})^\omega \), i.e., an infinite sequence of pseudo-states. A formula is interpreted with respect to triples \( (\sigma, \alpha, i) \), where \( \sigma \) is a structure, \( \alpha \) is a pseudo-structure, and \( i \geq 0 \) is a point in time. We now define what it means for a formula \( \varphi \) to *hold* in \( (\sigma, \alpha, i) \), denoted \( (\sigma, \alpha, i) \models \varphi \).

- **Propositional Constants:** \( (\sigma, \alpha, i) \models P \) if \( P \in \sigma(i) \).
- **Propositional Variables:** \( (\sigma, \alpha, i) \models X \) if \( X \in \alpha(i) \).
• **Boolean Connectives:** \((\sigma, \alpha, i) \models \neg \varphi\) if \((\sigma, \alpha, i) \not\models \varphi\).

\((\sigma, \alpha, i) \models \varphi \land \psi\) if \((\sigma, \alpha, i) \models \varphi\) and \((\sigma, \alpha, i) \models \psi\).

\((\sigma, \alpha, i) \models \varphi \lor \psi\) if \((\sigma, \alpha, i) \models \varphi\) or \((\sigma, \alpha, i) \models \psi\).

• **Temporal Connectives:** \((\sigma, \alpha, i) \models \bigoplus \varphi\) if \((\sigma, \alpha, i + 1) \models \varphi\).

\((\sigma, \alpha, i) \models \bigoplus \varphi\) if \(i > 0\) and \((\sigma, \alpha, i - 1) \models \varphi\).

\((\sigma, \alpha, i) \models \bigominus \varphi\) if either \(i = 0\) or \(i > 0\) and \((\sigma, \alpha, i - 1) \models \varphi\).

We still have to deal with fixpoint quantifiers. Let \(\alpha\) and \(\beta\) be pseudo-structures and \(X \in \mathcal{V}\). We say that \(\alpha\) and \(\beta\) agree except on \(X\) if for all \(i \geq 0\) and \(Y \in \mathcal{V}\) different from \(X\) we have that \(Y \in \alpha(i)\) iff \(Y \in \beta(i)\). Let \(\varphi(X)\) be a formula, let \(\sigma\) be a structure, and let \(\alpha\) be a pseudo-structure. We say that a pseudo-structure \(\alpha'\) is a *fixpoint of \(X \equiv \varphi(X)\)* with respect to \(\sigma\) and \(\alpha\) if \(\alpha\) and \(\alpha'\) agree except on \(X\) and we have that \((\sigma, \alpha', i) \models \varphi\) iff \((\sigma, \alpha, i) \models \varphi\). \(\alpha'\) is the *least fixpoint solution* if whenever \(\beta\) is also fixpoint solution then \((\sigma, \beta, i) \models X\) if \((\sigma, \alpha', i) \models X\). \(\alpha'\) is the *greatest fixpoint solution* if whenever \(\beta\) is also fixpoint solution then \((\sigma, \beta, i) \models X\) only if \((\sigma, \alpha', i) \models X\). By the Tarski-Knaster Fixpoint Theorem, our positivity condition ensures that the least fixpoint and the greatest fixpoint do exist.

We can now define the semantics of fixpoint formulas

- \((\sigma, \alpha, i) \models \mu X. \varphi(X)\) if \((\sigma, \alpha', i) \models X\), where \(\alpha'\) is the least fixpoint solution of \(X \equiv \varphi(X)\) with respect to \(\sigma\) and \(\alpha\).

- \((\sigma, \alpha, i) \models \nu X. \varphi(X)\) if \((\sigma, \alpha', i) \models X\), where \(\alpha'\) is the greatest fixpoint solution of \(X \equiv \varphi(X)\) with respect to \(\sigma\) and \(\alpha\).

It is easy to see that if \(\varphi\) is a sentence, then the truth value of \(\varphi\) at \((\sigma, \alpha, i)\) does not depend on \(\alpha\). Thus, we can define the truth value of sentences with respect to a point in a structure. That is, if \(\varphi\) is a sentence, \(\sigma\) is a structure, and \(i \geq 0\), then we say that \(\varphi\) holds in \((\sigma, i)\), denoted \((\sigma, i) \models \varphi\) if there is some pseudo-structure \(\alpha\) such that \((\sigma, \alpha, i) \models \varphi\). \(\sigma\) is a *model* of \(\varphi\) if \((\sigma, i) \models \varphi\) for some \(i \geq 0\). A sentence \(\varphi\) is *satisfiable* if it has a model.

### 2.2 An Example

We demonstrate now the use of \(\mu TL\) in modular specification. The example, based on [Pn86], concerns a trivial CSP solution to the mutual exclusion problem.

\[
P_1 :: *[l_0 : \langle C_1; e! \rangle; l_1 : \langle N_1; e! \rangle] \parallel P_2 :: *[m_0 : \langle N_2; e? \rangle; m_1 : \langle C_2; e? \rangle]
\]

\(C_1\) and \(C_2\) are the critical sections of \(P_1\) and \(P_2\), respectively. The mutual exclusion property here is a safety property that says that at \(l_0 \land m_1\) never holds. One can
prove using TL that this property holds, by proving that the property at \( l_0 \equiv m_0 \) is an invariant of the program. Unfortunately, this proof refers to the location \( m_0 \), which is an internal location of \( P_2 \). Thus, a modular verification can only refer to \( l_0 \) and \( m_1 \).

The solution proposed in [LPZ85,Pn86] is to show that in \( P_1 \) we may visit \( l_0 \) only after an even number of communications, and in \( P_2 \) we may visit \( m_1 \) only after an odd number of communications. It was shown in [Wo83] that this property cannot be expressed in TL. We now show how these properties can be specified in \( \mu TL \).

We first introduce the formula \( \text{init} \), which is the abbreviation of \( \bigoplus \text{false} \). Note that if \( (\sigma,i) \models \text{init} \), then \( i = 0 \). Also, if \( (\sigma,i) \models \bigoplus \text{init} \), then \( i = 1 \). We now describe a formula \( \text{even} \) such that \( (\sigma,i) \models \text{even} \) iff \( i \) is even. \( \text{even} \) is the formula \( \mu X. (\text{init} \lor \bigoplus \bigoplus X) \). Analogously, \( \text{odd} \) is the formula \( \mu X. (\bigoplus \text{init} \lor \bigoplus \bigoplus X) \).

Now the safety properties that we want to specify are simply:

\[
\text{at } l_0 \rightarrow \text{even}
\]

\[
\text{at } m_1 \rightarrow \text{odd}
\]

We now show that it is possible to specify the same safety properties without the past connective, but with less natural sentences. Instead of saying that \( \text{at } l_0 \) may hold only at even moments, we can say that it cannot hold at odd moments. This can be expressed by the formula

\[
\bigoplus (\mu X. (\neg \text{at } l_0 \land \bigoplus \bigoplus X)).
\]

The readers should convince themselves that if \( \sigma \) is a computation of \( P_1 \) and the above formulas holds in \( (\sigma,0) \), then \( \sigma \) satisfies the required safety property. Unlike \( \text{at } l_0 \rightarrow \text{even} \), the above formula is not a safety property; it is a property that has to hold at the beginning of the computation. Analogously, we can say that \( \text{at } m_1 \) does not hold at even moments by the formula

\[
\nu X. (\neg \text{at } m_1 \land \bigoplus \bigoplus X).
\]

Note, however, that by [Wo83], these properties cannot be specified in TL. Thus, the use of past connectives is indeed for convenience, but the expressive power beyond TL is necessary.

### 3 Pre-Models

A formula \( \psi \) is positive if it is not of the from \( \neg \xi \). \( \psi \) is in positive normal form if all its subformulas, with the possible exception of propositional constants, are positive. Every formula can be transformed to positive normal form with a linear increase in size. (One has to use equivalences such as \( \neg \bigoplus \psi \equiv \bigoplus \neg \psi \), \( \neg \bigoplus \psi \equiv \bigoplus \neg \psi \), and \( \neg \mu X. \psi (X) \equiv \mu X. \neg \psi (\neg X) \). Thus, we can focus on formulas in positive normal form.

To determine the truth value of a Boolean formula it suffices to consider subformulas. For modal and temporal formulas, one has to consider a bigger collection of formulas,
the so called Fischer-Ladner closure [FL79]. The closure of an $\mu TL$ sentence $\varphi$, denoted $cl(\varphi)$, is defined as follows:

- $\varphi \in cl(\varphi)$,
- if $\neg \psi \in cl(\varphi)$, then $\psi \in cl(\varphi)$,
- if $\psi \land \xi \in cl(\varphi)$ or $\psi \lor \xi \in cl(\varphi)$, then $\psi \in cl(\varphi)$ and $\xi \in cl(\varphi)$,
- if $\oplus \psi \in cl(\varphi)$, $\ominus \psi \in cl(\varphi)$, or $\ominus \psi \in cl(\varphi)$, then $\psi \in cl(\varphi)$,
- if $\mu X. \psi(X) \in cl(\varphi)$, then $\psi(\mu X. \psi(X)) \in cl(\varphi)$,
- if $\nu X. \psi(X) \in cl(\varphi)$, then $\psi(\nu X. \psi(X)) \in cl(\varphi)$.

It is not hard to check that the size of $cl(\varphi)$ is linear in the length of $\varphi$.

An atom $a$ of $\varphi$ is a set of positive sentences in $cl(\varphi)$ that satisfies the following properties:

- if $\psi \land \xi \in cl(\varphi)$, then $\psi \land \xi \in a$ iff $\psi \in a$ and $\xi \in a$,
- if $\psi \lor \xi \in cl(\varphi)$, then $\psi \land \xi \in a$ iff $\psi \in a$ or $\xi \in a$,
- if $\mu X. \psi(X) \in cl(\varphi)$, then $\mu X. \psi(X) \in a$ iff $\psi(\mu X. \psi(X)) \in a$,
- if $\nu X. \psi(X) \in cl(\varphi)$, then $\nu X. \psi(X) \in a$ iff $\psi(\nu X. \psi(X)) \in a$.

Intuitively, an atom is a consistent subset of $cl(\varphi)$. The set of atoms of $\varphi$ is denoted $at(\varphi)$. Clearly, the size of $at(\varphi)$ is at most exponential in the length of $\varphi$.

A pre-model $\pi$ of $\varphi$ is a member of $at(\varphi)^\omega$, i.e., an infinite sequence of atoms, that satisfies the following properties:

- $\varphi \in \pi(i)$, for some $i \geq 0$,
- if $\oplus \psi \in \pi(i)$, then $\psi \in \pi(i + 1)$,
- if $\ominus \psi \in \pi(i)$, then $i > 0$ and $\psi \in \pi(i - 1)$,
- if $\ominus \psi \in \pi(i)$ and $i > 0$, then $\psi \in \pi(i - 1)$.

A pre-model of $\varphi$ is almost a model of $\varphi$ except for fixpoint formulas that do not necessarily get the right semantics (that is, fixpoints are arbitrary rather than minimal or maximal as needed).

Fixpoint sentences “trigger” evaluation of other fixpoint formulas. For example, for the formula $\mu X. (P \lor \oplus X)$ to hold at point $i$, the sentence $P \lor \oplus (\mu X. (P \lor \oplus X))$ has to hold at point $i$. The distinction between least and greatest fixpoint formulas is
in the presence or absence of nonterminating evaluation sequences. The problem with these evaluation sequences is that they are hard to trace in the presence of disjunctions. Intuitively, we do not a priori know whether a disjunction $\psi \lor \xi$ will be true because of $\psi$ or because of $\xi$. To overcome this difficulty, Streett and Emerson introduced the technical notion of choice function [SE84].

A choice function $\rho$ for $\varphi$ is a partial function on $\text{cl}(\varphi)$ that is defined on all disjunctions and returns one of the disjuncts. Namely, either $\rho(\psi \lor \xi) = \psi$ or $\rho(\psi \lor \xi) = \xi$. An adorned atom is a pair $(a, \rho)$ of an atom and a choice function such that if $\psi \lor \xi \in a$, then $\psi \in a$ whenever $\rho(\psi \lor \xi) = \psi$, and $\xi \in a$ whenever $\rho(\psi \lor \xi) = \xi$. Let $\text{adat}(\varphi)$ be the set of adorned atoms of $\varphi$. It is easy to see that the number of adorned atoms is still at most exponential in the length of $\varphi$. An adorned pre-model for $\varphi$ is defined as above but with adorned atoms instead of atoms.

We can now define formally the notion of derivation between occurrences of sentences in adorned pre-models. Let $\pi$ be an adorned pre-model of $\varphi$, with $\pi(i) = (a_i, \rho_i)$. The derivation relation, denoted $\vdash$ is defined as follows:

- if $\psi \lor \xi \in a$ and $\rho_i(\psi \lor \xi) = \psi$, then $(\psi \lor \xi, i) \vdash (\psi, i)$.
- if $\psi \lor \xi \in a$ and $\rho_i(\psi \lor \xi) = \xi$, then $(\psi \lor \xi, i) \vdash (\xi, i)$.
- if $\psi \land \xi \in a$, then $(\psi \land \xi, i) \vdash (\psi, i)$ and $(\psi \land \xi, i) \vdash (\xi, i)$.
- if $\ominus \psi \in a$, then $(\ominus \psi, i) \vdash (\psi, i + 1)$.
- if $\ominus \psi \in a$, then $(\ominus \psi, i) \vdash (\psi, i - 1)$.
- if $\ominus \psi \in a$, and $i > 0$, then $(\ominus \psi, i) \vdash (\psi, i - 1)$.
- if $\mu X.\psi(X) \in a$, then $(\mu X.\psi(X), i) \vdash (\psi(\mu X.\psi(X)), i)$.
- if $\nu X.\psi(X) \in a$, then $(\nu X.\psi(X), i) \vdash (\psi(\nu X.\psi(X)), i)$.

A least fixpoint sentence sentences $\mu X.\psi(X)$ is said to be regenerated from point $i$ to point $j$ ($i$ might be equal to $j$) if there is a sequence $(\theta_1, i_1), \ldots, (\theta_k, i_k)$, $k > 1$, such that $\theta_1 = \theta_k = \mu X.\psi(X)$, $i_i = i$, $i_k = j$, $(\theta_l, i_l) \vdash (\theta_{l+1}, i_{l+1})$, for $0 < l < k$, and $\mu X.\psi(X)$ is a subsentence of each of the $\theta_l$'s. We say that $\pi$ is well-founded if there is no fixpoint sentence $\mu X.\psi(X) \in \text{cl}(\varphi)$ and an infinite sequence $i_0, i_1, \ldots$ such that $\mu X.\psi(X)$ is regenerated from $i_j$ to $i_{j+1}$ for all $j \geq 0$.

**Theorem 3.1:** [SE84] A sentence $\varphi$ of $\mu TL$ is satisfiable if and only if it has a well-founded adorned pre-model.

We want to build a finite-state B"uchi automaton that accepts precisely the models of $\varphi$, viewed these models as infinite words over the alphabet $2^P$. As an intermediate
step we build an automaton that accepts precisely the well-founded adorned pre-models, viewed as infinite words over the alphabet $adat(\varphi)$.

A Büchi automaton $A = (\Sigma, S, S_0, R, F)$ consists of an alphabet $\Sigma$, a finite set of state $S$, a set of initial states $S_0 \subseteq S$, a transition function $R : S \times \Sigma \to 2^S$, and a set of accepting states $F \subseteq S$. A run $r \in S^\omega$ of $A$ over an infinite word $w \in \Sigma^\omega$ is an infinite sequence of states such that $r(0) \in S_0$, and $r(i + 1) \in R(r(i), w(i))$ for all $i \geq 0$. This run is accepting if for infinitely many $i$'s we have $r(i) \in F$. $A$ accepts $w$ if it has an accepting run over $w$. The infinitary language accepted by $A$ is denoted $L_\omega(A)$ [Bu62].

The construction of an automaton that accept the well-founded adorned pre-models of $\varphi$ proceed in two steps. We first construct an automaton that accepts the adorned pre-models of $\varphi$. This automaton, denoted $A_1 = (adat(\varphi), S, S_0, R, F)$ is a very simple automaton. We have $S = \{\text{start}\} \cup adat(\varphi) \times \{0, 1\}$ (i.e., a state is either a special starting state or an atom tagged with a 0 or 1), $S_0 = \{\text{start}\}$, $F = adat(\varphi) \times \{1\}$ (i.e, all the states tagged with 1). It remains to define the transition function:

- $R(\text{start}, (b, \rho))$ is the singleton set $\{(b, \rho, j)\}$, where $j = 1$ iff $\varphi \in b$, if $b$ does not contain any formula of the form $\exists \psi$. If this condition does not hold, then $R(\text{start}, (b, \rho))$ is empty.

- $R((a, \rho, i), (b, \rho'))$ is the singleton set $\{(b, \rho', j)\}$, where $j = 1$ iff either $i = 1$ or $\varphi \in b$, if the following holds: if $\exists \psi \in a$, then $\psi \in b$, if $\exists \psi \in b$, then $\psi \in a$, and if $\exists \psi \in b$, then $\psi \in a$. If this condition does not hold, then $R((a, \rho, i), (b, \rho'))$ is empty.

It is easy to check that $A_1$ accepts precisely the adorned pre-models of $\varphi$.

The second step is to construct an automaton that checks for non-well-foundedness. This automaton, whose number of states is quadratic in the length of $\varphi$, seeks an infinite regeneration sequence for a least fixpoint sentence in $cl(\varphi)$. If $\varphi$ is an $\nu TL$ formula, then the regeneration sequence is a nondecreasing sequence, so we can indeed construct an automaton to find it. Given such an automaton $A_2$ we can use the following theorem:

**Theorem 3.2:** [SVW85] Given a Büchi automaton $A$ with $n$ states, there is a Büchi automaton $\overline{A}$ with $O(16^n^2)$ states such that $L_\omega(\overline{A}) = \Sigma^\omega - L_\omega(A)$.

Thus, if $A_2$ is the automaton that finds an infinite regeneration sequence, then $L_\omega(A_1) \cap L_\omega(\overline{A}_2)$ is precisely the set of all well-founded adorned pre-models of $\varphi$. All that is left is to construct the intersection of $A_1$ and $\overline{A}_2$ and then project it on the alphabet $2^\varphi$, and voilá we have a Büchi automaton that accepts precisely the models of $\varphi$ and whose size is at most exponential in the length of $\varphi$. The problem with $\mu TL$ formulas is that the regeneration sequence is not non-decreasing, and we need a 2-way automaton to find it.
4 Two-Way Büchi Automata

A 2-way Büchi automaton $A = (\Sigma, S, S_0, R, F)$ consists of an alphabet $\Sigma$, a finite set of states $S$, a set of initial states $S_0 \subseteq S$, a transition function $R : S \times \Sigma \to 2^{S \times \{-1, 0, 1\}}$, and a set of accepting states $F \subseteq S$. Intuitively a transition indicates not only the new state of the automaton, but also whether the head should move left, right, or stay in place. A configuration of $A$ is a member of $S \times \mathbb{N}$, i.e., a pair consisting of a state and a “position”. A run $r \in (S \times \mathbb{N})^\omega$ of $A$ over an infinite word $w \in \Sigma^\omega$ is an infinite sequence of configurations such that $r(0) \in S_0 \times \{0\}$, and for all $i \geq 0$ if $r(i) = (s, j)$, then there is some $(t, k) \in R(s, w(j))$ such that $r(i + 1) = (t, j + k)$. This run is accepting if for infinitely many $i$’s we have $r(i) \in F \times \mathbb{N}$. $A$ accepts $w$ if it has an accepting run over $w$. The infinitary language accepted by $A$ is denoted $L_\omega(A)$.

In [St82], Streett has defined 2-way automata on infinite trees. When specialized to infinite words, 2-way Streett automata seems to be incomparable to 2-way Büchi automata. The results in [St82] and here imply, however, that both 2-way Street automata on infinite words and 2-way Büchi automata have the expressive power of 1-way Büchi automata. Note, however, that while our transformation is exponential, the transformation in [St82] is quadruply exponential!

We now show that a 2-way automaton can be converted to a 1-way automaton. Furthermore, we want to accomplish that and also to complement the automaton all with the total cost of at most an exponential blow-up. The key to this is the ability to keep in a finitary way all the information about “backwards” runs, which is essentially the idea in [Sh59].

Let $A = (\Sigma, S, S_0, R, F)$ be a 2-way Büchi automaton. An $A$-label is a subset of $\text{labels}(A) = S^2 \times \{0, 1\}$, i.e., pairs of states tagged by 0 or 1. Intuitively, a triple $(s, t, 0)$ denotes the fact that there is a backward run starting at a state $s$ and ending at a state $t$. The triple $(s, t, 1)$ denotes in addition that the run passes through a state in $F$. Let $w \in \Sigma$ be an infinite word. An $A$-labeling $(l, m) \in (\text{labels}(A)^\omega)^2$ for $w$ consists of two infinite sequences of labels that satisfy the following conditions:

- $(s, t, 0) \in l(i)$ iff either $(t, 0) \in R(s, w(i))$ or $i > 1$ and there are states $s', t' \in S$ such that $(s', t', j) \in m(i - 1)$, $(s', -1) \in R(s, w(i))$, and $(t, 1) \in R(t', w(i - 1))$.

- $(s, t, 1) \in l(i)$ iff either $(t, 0) \in R(s, w(i))$ and $t \in F$ or $i > 1$ and there are states $s', t' \in S$ such that $(s', t', j) \in m(i - 1)$, $(s', -1) \in R(s, w(i))$, and $(t, 1) \in R(t', w(i - 1))$, and in addition either $j = 1$ or one of $s', t', t$ is in $F$.

- $(s, t, 0) \in m(i)$ iff there is a sequence $s_0, ..., s_k, k > 0$, such that $s_0 = s$, $s_k = t$ and $(s_j, s_{j+1}, 0) \in l(i)$ for $0 \geq j > k$.

- $(s, t, 1) \in m(i)$ iff there is a sequence $s_0, ..., s_k, k > 0$, such that $s_0 = s$, $s_k = t$, $(s_j, s_{j+1}, k_j) \in l(i)$ for $0 \geq j > k$, and either one of the $k_j$’s is 1 or one of the $s_j$’s, $j > 0$, is in $F$. 

10
Intuitively, $m$ keeps the information about backward runs; $l$ is used just to simplify the definition. It is easy to see that every infinite word has a unique $A$-labeling.

We now consider words over the alphabet $\Sigma_A = \Sigma \times labels(A)^2$. Let $u$ be the word $(w_0, l_0, m_0), (w_1, l_1, m_1), \ldots \in \Sigma_A^n$. We say that the the word is $A$-legal if $(l, m)$ is an $A$-labeling of $w$, where $w = w_0, w_1, \ldots, l = l_0, l_1, \ldots$, and $m = m_0, m_1, \ldots$. We abuse notation and denote $u$ by the triple $(w, l, m)$.

It is now straightforward to construct a Büchi automaton that accepts the legal words. Given a word $(w, l, m)$ over $\Sigma_A^n$, the automaton just has to check that $(l, m)$ is an $A$-labeling of $w$.

**Lemma 4.1:** Let $A$ be a 2-way Büchi automaton with $n$ states. There is a 1-way Büchi automaton $A'$ with at most $|\Sigma| \cdot 4^n$ states such that $A'$ accepts precisely all the $A$-legal words.

The usefulness of $A$-labeling is that it supplies enough information to check in a “1-way sweep” whether a word is accepted by the 2-way automaton $A$.

**Lemma 4.2:** Let $A$ be a 2-way Büchi automaton with $n$ states. There is a 1-way Büchi automaton $A''$ with $2n$ states such that a given legal word $(w, l, m)$ is accepted by $A''$ if and only if $w$ is accepted by $A$.

Essentially the automaton $A''$ uses the information supplied in the labelling to avoid going backwards.

We now note that the projection of the legal words on the alphabet $\Sigma$ yields $\Sigma^\omega$. Thus, the projection of the language $L_\omega(A') \cap L_\omega(A'')$ on the alphabet $\Sigma$ yield the language $L_\omega(A)$, and the projection of the language $L_\omega(A') \cap L_\omega(A''')$ on the alphabet $\Sigma$ yield the language $\Sigma^\omega - L_\omega(A)$. Using Theorem 3.2 we obtain:

**Theorem 4.3:** Let $A$ be a 2-way Büchi automaton with $n$ states. Then there are 1-way Büchi automata $B_1$ and $B_2$ with $O(\exp(n^2))$ states such that $L_\omega(B_1) = L_\omega(A)$ and $L_\omega(B_2) = \Sigma^\omega - L_\omega(A)$.

We can now complete the desired construction for $\mu$TL formulas. We construct a 2-way automaton $A_2$ to detect infinite regeneration. $A_2$ nondeterministically select a sentence $\psi$ in $cl(\varphi)$ and an occurrence of $\psi$ in the pre-model. $A_2$ then traces a derivation sequences and verifies that $\psi$ regenerates itself infinitely often. Thus, $A_2$ needs to remember only the last step of the derivation in addition to $\psi$. Consequently, the number of its states is at most quadratic in the length of $\varphi$. Using Theorem 4.3 we obtain the a 1-way automaton $\overline{A_2}$ that is the complement of $A_2$. By taking the intersection of $A_1$ and $\overline{A_2}$ and projecting the result on the alphabet $2^p$ we obtain a 1-way Büchi automaton that accepts precisely the models of $\varphi$.

**Theorem 4.4:** Given an $\mu$TL sentence $\varphi$, we can construct a Büchi automaton $A_\varphi$ whose number of states is at most exponential in the length of $\varphi$ such that $L_\omega(A_\varphi)$ is the set of models of $\varphi$. 

11
Remark 4.5: Suppose that $|\varphi| = n$. Then, the automaton $A_2$ has $O(n^2)$ states. Thus, a straightforward construction of $A_2$ would yield an automaton with $O(exp(n^4))$ states. In the full paper we shall show that by being more careful we can construct $A_2$ with only $O(exp(n^3))$ states. □

As mentioned in the introduction, Theorem 4.4 is basic result that has many application. In particular we have:

Corollary 4.6: The satisfiability problem for $\mu$TL is PSPACE-complete.

Corollary 4.7: There is an algorithm that verifies that finite-state programs satisfies their $\mu$TL specification, whose complexity in the size of the programs is co-NLOGSPACE-complete and whose complexity in the size of the specifications is PSPACE-complete.

5 Concluding Remarks

We have extended the automata-theoretic paradigm to a temporal logic that includes both fixpoint quantifiers and past connectives. We believe that our technical result is an important step in the development of modular temporal specification and verification methodology for concurrent programs.

Acknowledgments. I am grateful to B. Banieqbal and H. Barringer for stimulating discussions that inspired me to study temporal fixpoint calculi. I'd like also to thank O. Lichtenstein, R. Rosner, and an anonymous member of the program committee for helpful comments on a previous draft of this paper.

6 References


