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 μTL, which is the extension of temporal logic by fixpoint operators and past temporal connectives. We extend the automata-theoretic paradigm to μTL. That is, we show how, given an μTL formula φ, we can produce a finite-state Büchi automaton A_φ, whose size is at most exponentially bigger than the size of φ, such that A_φ accepts precisely the computations that satisfy φ. The result has immediate applications, e.g., an optimal decision procedure and a model-checking algorithm for μTL.

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 expressive power of TL was Wolper [Wo83] who observed that temporal logic cannot express regular events (in fact, TL can express

*Address: IBM Almaden Research K53-802, 650 Harry Rd., San Jose, CA 95120-6099. Part of this research was done while the author was visiting the Weizmann Institute of Science.

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 νTL, 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]. νTL 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 [GPSS80] 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 μTL, which is the extension of νTL with past connectives. We wish to extend the

automata-theoretic paradigm advocated in [VW85] to μTL. 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 φ, produce a finite-state Büchi automaton A_φ (which is an automaton on infinite words), whose size is at most exponentially bigger than the size of φ, such that A_φ accepts precisely the computations that satisfy φ [SVW85, VW86, WVS83]. The beauty of this result is that for many application A_φ is all that necessary. First, one can check whether A_φ accepts some word to determine whether φ is satisfiable [SVW85,WVS83]. This gives us a decision procedure for satisfiability that runs in polynomial space. Secondly, one can view A_φ as a synthesized program that implements the specification φ [MW84]. Thirdly, given a finite-state program P, one can combine P with A_φ and then check whether P satisfies φ by a reachability analysis of the combined program [Va85,VW86]. Finally, given a program P one can use A_φ to derive proof obligations that when checked will guarantee the correctness of P with respect to φ [AS85, AS87, MP87, Va87].

We first show that the automata-theoretic paradigm can be extended to νTL. This result requires two basic techniques. The first technique is borrowed from the analysis by Streett and Emerson of the μ-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 $\varphi(\mu X.\varphi(X))$ [1] (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 μTL we encounter another difficulty. Since μTL 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 φ.

result for μTL.

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 μTL are:

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

A *sentence* of μTL is a formula with no free propositional variables.

Intuitively, $\oplus\varphi$ (resp. $\ominus\varphi$) says that φ holds in the *next* (resp. *previous*) time moment. The formula $\ominus\varphi$ says that φ 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 (σ, α, i), where σ is a structure, α is a pseudo-structure, and $i \geq 0$ is a point in time. We now define what it means for a formula φ to *hold* in (σ, α, 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 $(\sigma, \alpha, i) \not\models \varphi$.
  $(\sigma, \alpha, i) \models \varphi \wedge \psi$ if $(\sigma, \alpha, i) \models \varphi$ and $(\sigma, \alpha, i) \models \psi$.
  $(\sigma, \alpha, i) \models \varphi \vee \psi$ if $(\sigma, \alpha, i) \models \varphi$ or $(\sigma, \alpha, i) \models \psi$.
- **Temporal Connectives**: $(\sigma, \alpha, i) \models \bigoplus \varphi$ if $(\sigma, \alpha, i+1) \models \varphi$.
  $(\sigma, \alpha, i) \models \bigominus \varphi$ if $i > 0$ and $(\sigma, \alpha, i-1) \models \varphi$.
  $(\sigma, \alpha, i) \models \bigoplus \varphi$ if either $i = 0$ or $i > 0$ and $(\sigma, \alpha, i-1) \models \varphi$.

We still have to deal with fixpoint quantifiers. Let α and β be pseudo-structures and $X \in \mathcal{V}$. We say that α and β *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 σ be a structure, and let α be a pseudo-structure. We say that a pseudo-structure α' is a *fixpoint* of $X \equiv \varphi(X)$ with respect to σ and α if α and α' agree except on X and we have that $(\sigma, \alpha', i) \models \varphi$ iff $(\sigma, \alpha', i) \models X$. α' is the *least* fixpoint solution if whenever β is also fixpoint solution then $(\sigma, \beta, i) \models X$ if $(\sigma, \alpha', i) \models X$. α' is the *greatest* fixpoint solution if whenever β 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 α' is the least fixpoint solution of $X \equiv \varphi(X)$ with respect to σ and α.
- $(\sigma, \alpha, i) \models \nu X.\varphi(X)$ if $(\sigma, \alpha', i) \models X$, where α' is the greatest fixpoint solution of $X \equiv \varphi(X)$ with respect to σ and α.

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

2.2 An Example

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

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

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* $l_0 \wedge$ *at* m_1 never holds. One can

prove using TL that this property holds, by proving that the property $at\ l_0 \equiv at\ m_0$ 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 μTL.

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

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

$$at\ l_0 \rightarrow even$$

$$at\ m_1 \rightarrow odd$$

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 $at\ l_0$ may hold only at even moments, we can say that it cannot hold at odd moments. This can be expressed by the formula

$$\oplus(\nu X.(\neg at\ l_0 \wedge \oplus\oplus X)).$$

The readers should convince themselves that if σ is a computation of P_1 and the above formulas holds in $(\sigma, 0)$, then σ satisfies the required safety property. Unlike $at\ l_0 \rightarrow even$, 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 $at\ m_1$ does not hold at even moments by the formula

$$\nu X.(\neg at\ m_1 \wedge \oplus\oplus X).$$

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 ψ is *positive* if it is not of the from $\neg\xi$. ψ 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 $\neg\oplus\psi \equiv \oplus\neg\psi$, $\neg\ominus\psi \equiv \ominus\neg\psi$, and $\neg\mu X.\psi(X) \equiv \nu X.\neg\psi(\neg X)$.) 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 μTL sentence φ, denoted $cl(\varphi)$, is defined as follows:

- $\varphi \in cl(\varphi)$,
- if $\neg\psi \in cl(\varphi)$, then $\psi \in cl(\varphi)$,
- if $\psi \wedge \xi \in cl(\varphi)$ or $\psi \vee \xi \in cl(\varphi)$, then $\psi \in cl(\varphi)$ and $\xi \in cl(\varphi)$,
- if $\bigoplus \psi \in cl(\varphi)$, $\bigominus \psi \in cl(\varphi)$, or $\bigcirc\!\!\!\!\!- \psi \in cl(\varphi)$, 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 φ.

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

- if $\psi \wedge \xi \in cl(\varphi)$, then $\psi \wedge \xi \in \mathbf{a}$ iff $\psi \in \mathbf{a}$ and $\xi \in \mathbf{a}$,
- if $\psi \vee \xi \in cl(\varphi)$, then $\psi \wedge \xi \in \mathbf{a}$ iff $\psi \in \mathbf{a}$ or $\xi \in \mathbf{a}$,
- if $\mu X.\psi(X) \in cl(\varphi)$, then $\mu X.\psi(X) \in \mathbf{a}$ iff $\psi(\mu X.\psi(X)) \in \mathbf{a}$,
- if $\nu X.\psi(X) \in cl(\varphi)$, then $\nu X.\psi(X) \in \mathbf{a}$ iff $\psi(\nu X.\psi(X)) \in \mathbf{a}$.

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

A *pre-model* π of φ 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 $\bigoplus \psi \in \pi(i)$, then $\psi \in \pi(i+1)$,
- if $\bigcirc\!\!\!\!\!- \psi \in \pi(i)$, then $i > 0$ and $\psi \in \pi(i-1)$.
- if $\bigominus \psi \in \pi(i)$ and $i > 0$, then $\psi \in \pi(i-1)$.

A pre-model of φ is almost a model of φ 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 $\mu X.(P \vee \bigoplus X)$ to hold at point i, the sentence $P \vee \bigoplus(\mu X.(P \vee \bigoplus X))$ 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 $\psi \vee \xi$ will be true because of ψ or because of ξ. To overcome this difficulty, Streett and Emerson introduced the technical notion of *choice function* [SE84].

A choice function ρ for φ is a partial function on $cl(\varphi)$ that is defined on all disjunctions and returns one of the disjuncts. Namely, either $\rho(\psi \vee \xi) = \psi$ or $\rho(\psi \vee \xi) = \xi$. An *adorned atom* is a pair $(\mathbf{a}, \rho)$ of an atom and a choice function such that if $\psi \vee \xi \in \mathbf{a}$, then $\psi \in \mathbf{a}$ whenever $\rho(\psi \vee \xi) = \psi$, and $\xi \in \mathbf{a}$ whenever $\rho(\psi \vee \xi) = \xi$. Let $adat(\varphi)$ be the set of adorned atoms of φ. It is easy to see that the number of adorned atoms is still at most exponential in the length of φ. An *adorned pre-model* for φ 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 π be an adorned pre-model of φ, with $\pi(i) = (\mathbf{a}_i, \rho_i)$. The derivation relation, denoted $\vdash$ is defined as follows:

- if $\psi \vee \xi \in \mathbf{a}_i$ and $\rho_i(\psi \vee \xi) = \psi$, then $(\psi \vee \xi, i) \vdash (\psi, i)$,
- if $\psi \vee \xi \in \mathbf{a}_i$ and $\rho_i(\psi \vee \xi) = \xi$, then $(\psi \vee \xi, i) \vdash (\xi, i)$,
- if $\psi \wedge \xi \in \mathbf{a}_i$, then $(\psi \wedge \xi, i) \vdash (\psi, i)$ and $(\psi \wedge \xi, i) \vdash (\xi, i)$,
- if $\bigoplus \psi \in \mathbf{a}_i$, then $(\bigoplus \psi, i) \vdash (\psi, i+1)$,
- if $\ominus \psi \in \mathbf{a}_i$, then $(\ominus \psi, i) \vdash (\psi, i-1)$,
- if $\bigominus \psi \in \mathbf{a}_i$ and $i > 0$, then $(\bigominus \psi, i) \vdash (\psi, i-1)$,
- if $\mu X.\psi(X) \in \mathbf{a}_i$, then $(\mu X.\psi(X), i) \vdash (\psi(\mu X.\psi(X)), i)$,
- if $\nu X.\psi(X) \in \mathbf{a}_i$, 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 θ_i's. We say that π is well-founded if there is no fixpoint sentence $\mu X.\psi(X) \in cl(\varphi)$ 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 φ of μTL is satisfiable if and only if it has a well-founded adorned pre-model.*

We want to build a finite-state Büchi automaton that accepts precisely the models of φ, viewed these models as infinite words over the alphabet $2^{\mathcal{P}}$. 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 Σ, 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 φ proceed in two steps. We first construct an automaton that accepts the adorned pre-models of φ. This automaton, denoted $A_1 = (adat(\varphi), S, S_0, R, F)$ is a very simple automaton. We have $S = \{\mathbf{start}\} \cup adat(\varphi) \times \{0, 1\}$ (i.e., a state is either a special starting state or an atom tagged with a 0 or 1), $S_0 = \{\mathbf{start}\}$, $F = adat(\varphi) \times \{1\}$ (i.e, all the states tagged with 1). It remains to define the transition function:

- $R(\mathbf{start}), (\mathbf{b}, \rho))$ is the singleton set $\{(\mathbf{b}, \rho, j)\}$, where $j = 1$ iff $\varphi \in \mathbf{b}$, if $\mathbf{b}$ does not contain any formula of the form $\ominus\!\!\!\!\ominus \psi$. If this condition does not hold, then $R(\mathbf{start}), (\mathbf{b}, \rho))$ is empty.
- $R((\mathbf{a}, \rho, i), (\mathbf{b}, \rho'))$ is the singleton set $\{(\mathbf{b}, \rho', j)\}$, where $j = 1$ iff either $i = 1$ or $\varphi \in \mathbf{b}$, if the following holds: if $\oplus \psi \in \mathbf{a}$, then $\psi \in \mathbf{b}$, if $\ominus\!\!\!\!\ominus \psi \in \mathbf{b}$, then $\psi \in \mathbf{a}$, and if $\ominus \psi \in \mathbf{b}$, then $\psi \in \mathbf{a}$, If this condition does not hold, then $R((\mathbf{a}, \rho, i), (\mathbf{b}, \rho'))$ is empty.

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

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 φ, seeks an infinite regeneration sequence for a least fixpoint sentence in $cl(\varphi)$. If φ is an νTL 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 φ. All that is left is to construct the intersection of A_1 and $\overline{A_2}$ and then project it on the alphabet $2^{\mathcal{P}}$, and *voilá* we have a Büchi automaton that accepts precisely the models of φ and whose size is at most exponential in the length of φ. The problem with μTL 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 Σ, a finite set of state S, a set of initial states $S_0 \subseteq S$, a transition function $R : S \times \Sigma \to 2^{S \times \{-1,0,1\}}$, 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 $S \times I\!N$, i.e., a pair consisting of a state and a "position". A *run* $r \in (S \times I\!N)^\omega$ 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 $r(i) \in F \times I\!N$. *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 $labels(A) = S^2 \times \{0, 1\}$, 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* $(l, m) \in (labels(A)^\omega)^2$ 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.

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 $(w_0, l_0, m_0), (w_1, l_1, m_1), \ldots \in \Sigma_A^\omega$. 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 Σ_A^ω, 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 $|\Sigma| \cdot 4^{n^2}$ 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 Σ yields Σ^ω. Thus, the projection of the language $L_\omega(A') \cap L_\omega(A'')$ on the alphabet Σ yield the language $L_\omega(A)$, and the projection of the language $L_\omega(A') \cap L_\omega(\overline{A''})$ on the alphabet Σ yield the language $\Sigma^\omega - L_\omega(A)$. Using Theorem 3.2 we obtain:

Theorem 4.3: *Let A be a 2-way Büchi automata 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 μTL formulas. We construct a 2-way automaton A_2 to detect infinite regeneration. A_2 nondeterministically select a sentence ψ in $cl(\varphi)$ and an occurence of ψ in the pre-model. A_2 then traces a derivation sequences and verifies that ψ regenerates itself infinitely often. Thus, A_2 needs to remember only the last step of the derivation in addition to ψ. Consequently, the number of its states is at most quadratic in the length of φ. 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 $2^{\mathcal{P}}$ we obtain a 1-way Büchi automaton that accepts precisely the models of φ.

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

Remark 4.5: Suppose that $|\varphi| = n$. Then, the automaton A_2 has $O(n^2)$ states. Thus, a straightforward construction of A_φ 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_φ 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 μTL is PSPACE-complete.*

Corollary 4.7: *There is an algorithm that verifies that finite-state programs satisfies their μTL 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

[**AS85**] Alpern, B., Schneider, F.B.: *Verifying temporal properties without using temporal logic.* Technical Report TR–85–723, Cornell University, Dec. 1985.

[**AS87**] Alpern, B., Schneider, F.B.: Proving Boolean combinations of deterministic properties. *Proc. 2nd IEEE Symp. on Logic in Computer Science*, Ithaca, June 1987.

[**Ba86**] Barringer, H.: Using temporal logic in the compositional specification of concurrent systems. Dept. of Computer Science, University of Manchester, Technical Report UMCS–86–10–1.

[**BB86**] Banieqbal, B. Barringer: A study of an extended temporal language and a temporal fixed point calculus. Dept. of Computer Science, University of Manchester, Technical Report UMCS–86–10–2.

[**BK85**] B. Barringer, Kuiper, R.: Hierarchical development of concurrent systems in a temporal logic framework. *Seminar in Concurrency*, eds. S.D. Brookes et al., Springer-Verlag, Lecture Notes in Computer Science 197, 1985, pp. 35–61.

[**BKP85**] Barringer, H., Kuiper, R., Pnueli, A.: A compositional temporal approach to a csp-like language. *Formal Models of Programming*, eds. E.J. Neuhold and G. Chroust, North Holland, 1985, pp. 207-227.

[**BKP86**] Barringer, H., Kuiper, R., Pnueli, A.: A really abstract concurrent model and its temporal logic. *Proc. 13th ACM Symp. on Principles of Programming Languages*, St. Petersburg, 1986.

[**Bu62**] Büchi, J.R.: On a decision method in restricted second–order arithmetics. *Proc. Int'l Congr. on Logic, Method. and Phil. of Sci. 1960*, Stanford University Press, 1962, pp. 1–12.

[**Ch74**] Choueka, Y.: Theories of automata on ω–tapes – a simplified approach. *J. Computer and System Science* 8(1974), pp. 117–141.

[**EC80**] Emerson, E.A., Clarke, E.C.: Characterizing correctness properties of parallel programs using fixpoints. *Proc. 7th Int'l Colloq. on Automata, Languages and Programming*, 1980, pp. 169–181.

[**EP87**] Emerson, E.A., Pnueli, A.: Temporal and Modal Logic. *Handbook of Theoretical Computer Science*, eds. J. van Leeuwen et. al., North-Holland Pub., in press, 1987.

[**FL79**] Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. *J. Computer and Systems Science*, 18(1979), pp. 194–211.

[**GPSS80**] Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. *Proc. 7th ACM Symp. on Principles of Programming Languages*, 1980, pp. 163–173.

[**Ko82**] Kozen, D.: Results on the propositional mu-calculus. *Proc. 9th Int'l Colloq. on Automata, Languages and Programming*, 1982, pp. 348–359.

[**KVR83**] Koymans, R., Vytopil, J., DeRoever, W.P.: Real-time programming and asynchronous message passing. *Proc. 2nd ACM Symp. on Principles of Distributed Computing*, Montreal, 1983, pp. 187–197.

[**LP85**] Lichtenstein, O., Pnueli, A.: Checking that finite–state concurrent programs satisfy their linear specification. *Proc. 12th ACM Symp. on Principles of programming Languages*, 1985, pp. 97–107.

[**LPZ85**] Lichtenstein, O., Pnueli, A., Zuck L.: The glory of the past. *Proc. Workshop on Logics of Programs*, Brooklyn, Springer-Verlag, Lecture Notes in Computer Science 193, 1985, pp. 97–107.

[**MP83**] Manna, Z., Pnueli, A.: How to cook a temporal proof system for your pet language. *Proc. 10th Symposium on Principles of Programming Languages*, Austin, Texas, 1983, pp. 141–154.

[**MP87**] Manna, Z., Pnueli, A.: Specification and verification of concurrent programs by ∀–automata. *Proc. 14 ACM Symp. on Principles of Programming Languages*, Munich, Jan. 1987, pp. 1–12.

[**MW84**] Manna, Z., Wolper, P.: Synthesis of communicating processes from temporal logic specifications. *ACM Trans. on Programming Languages*, 1984, pp. 68–93.

[**Pn77**] Pnueli, A.: The temporal logic of programs. *Proc. 18th Symp. on Foundations of Computer Science*, 1977, pp. 46–57.

[**Pn85**] Pnueli, A: In transition from global to modular reasoning about programs. *Logics and Models of Concurrent Systems*, ed. K.R. Apt, Springer-Verlag, 1985, pp. 123–144.

[**Pn86**] Pnueli, A: Applications of temporal logic to the specification and verification of reactive systems - a survey of current trends. *Current Trends in Concurrency*, eds. J.W. de Bakker et al., Springer-Verlag, Lecture Notes in Computer Science 224, 1986, pp. 510–584.

[**Pr51**] Prior, A.: *Past, Present, and Future*, Oxford Press, 1951.

[**Pr81**] Pratt, V.R.: Program Logic Without Binding is Decidable, *Proc. 8th ACM Symp. on Principles of Programming Languages*, 1981.

[**Pr82**] Pratt, V.R.: A decidable mu-calculus. *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, 1982, pp. 421–427.

[**RS59**] Rabin, M.O., Scott, D.: Finite automata and their decision problems. *IBM J. Research and Development*, 3(1959), pp. 114–125.

[**RU71**] Rescher, N, Urquhart, A.: *Temporal Logic*. Springer–Verlag, 1971.

[**SC85**] Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. *J. ACM* 32(1985), pp. 733–749.

[**SE84**] Streett, R.S., Emerson, E.A.: The propositional mu-calculus is elementary. *Proc. 11th Int'l Colloq. on Automata, Languages and Programming*, 1984, Springer-Verlag, Lecture Notes in Computer Science 172, pp. 465–472.

[**Sh59**] Shepherdson, J.C.: The reduction of two-way automata to one-way automata. *IBM J. Research and Development*, 3(1959), pp. 199–201.

[**St82**] Streett, R.S.: Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1982), pp. 121–141.

[**SVW85**] Sistla, A.P., Vardi, M.Y., Wolper, P.: The complementation problem for Büchi automata with applications to temporal logic. Theoretical Computer Science 49(1987), pp. 217-237.

[**To81**] Thomas, W.: A combinatorial approach to the theory of ω-automata. *Information and Control* 48(1981), pp. 261–283.

[**Va85**] Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. *Proc. 26th IEEE Symp. on Foundations of Computer Science*, Portland, 1985, pp. 327–338.

[**VW86**] Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. *Proc. IEEE Symp. on Logic in Computer Science*, Cambridge, 1986, pp. 332–344.

[**VW86**] Vardi, M.Y., Wolper, P.: Applications of temporal logic - an automata-theoretic prespective. Stanford University, CSLI, 1985.

[**Wo83**] Wolper, P.: Temporal logic can be more expressive. *Information and Control* 56(1983), pp. 72–99.

[**WVS83**] Wolper, P.L., Vardi, M.Y., Sistla, A.P.: Reasoning about infinite computation paths. *Proc. 24th IEEE Symp. on Foundation of Computer Science*, Tuscon, 1983, pp. 185–194.