Extended Temporal Logic Revisited

Orna Kupferman[*]
Hebrew University

Nir Piterman[†]
Weizmann Institute of Science

Moshe Y. Vardi[‡]
Rice University

March 26, 2001

Abstract

A key issue in the design of a model-checking tool is the choice of the formal language with which properties are specified. It is now recognized that a good language should extend linear temporal logic with the ability to specify all ω-regular properties. Also, familiar with finite-state machines, designers prefer extensions based on automata than these based on fixed points or propositional quantification. Early extensions of linear temporal logic with automata use nondeterministic Büchi automata. Their drawback has been inability to refer to the past and the asymmetrical structure of nondeterministic automata.

In this work we study an extension of linear temporal logic, called ETL_{2a}, that uses two-way alternating automata as temporal connectives. Two-way automata can traverse the input word back and forth and they are exponentially more succinct than one-way automata. Alternating automata combine existential and universal branching and they are exponentially more succinct than nondeterministic automata. The rich structure of two-way alternating automata makes ETL_{2a} a very powerful and convenient logic. We show that ETL_{2a} formulas can be translated to nondeterministic Büchi automata with an exponential blow up. It follows that the satisfiability and model-checking problems for ETL_{2a} are PSPACE-complete, as are the ones for LTL and its earlier extensions with automata. So, in spite of the succinctness of two-way and alternating automata, the advantages of ETL_{2a} are obtained essentially for free. The recent acceptance of alternating automata by the industry and the development of symbolic procedures for handling them make us optimistic about the practicality of ETL_{2a}.

[*] Address: School of Computer Science and Engineering, Hebrew University, Jerusalem 91904, Israel. Email: orna@cs.huji.ac.il

[†] Department of Computer Science and Applied Mathematics, Weizmann institute, Rehovot 76100, Israel. Email: nirp@wisdom.weizmann.ac.il

[‡] Address: Department of Computer Science, Rice University, Houston TX 77005-1892, U.S.A. Email: vardi@cs.rice.edu

1 Introduction

In *formal verification*, we check that a system is correct with respect to a desired behavior by checking that a mathematical model of the system satisfies a formal specification of the behavior. Early formal-verification efforts considered terminating systems. There, the specification relates an initial condition about the system with a condition that is guaranteed to be satisfied upon its termination [Fra92]. In 1977, Pnueli suggested to use temporal logic in order to describe nonterminating and reactive systems [Pnu81]. Temporal logics augment propositional logics with *temporal modalities*, making it possible to describe a sequence of events in time. For example, using the temporal modalities *always* ($\Box$) and *eventually* ($\Diamond$), we can specify the behavior "if p holds in all future moments then there is a future moment in which q holds" ($\Box p \rightarrow \Diamond q$). Temporal logic has led to the development of algorithmic methods for reasoning about reactive systems. In particular, temporal logic *model checking* enjoys a substantial and growing use in industrial applications [BBG$^+$94].

A key issue in the design of a model-checking tool is the choice of the formal language with which behaviors are specified. Almost two decades ago, Wolper argued that some very basic behaviors cannot be expressed by the linear temporal logic LTL. For example, he showed that the behavior "$\Box^2 p$", stating that an atomic proposition p is true in all even positions, cannot be expressed in LTL [Wol83]. Wolper suggested to extend linear temporal logic by grammar operators. It is more convenient to think about Wolper's extension in terms of ω-regular languages, as was later suggested in [VW94]. Intuitively, if the system is defined over a set AP of atomic propositions, then an infinite behavior of the system can be viewed as a word over the alphabet 2^{AP}, and a set of allowed behaviors can be described by an ω-regular automaton whose alphabet consists of Boolean formulas over AP. For example, the behavior $\Box^2 p$ can be described by an automaton whose language is $(\mathbf{true} \cdot p)^\omega$, and the behavior $\Box p \rightarrow \Diamond q$ can be described by an automaton whose language is $\mathbf{true}^* \cdot ((\neg p) \vee q) \cdot \mathbf{true}^\omega$.

It turned out that LTL can express precisely the star-free ω-regular behaviors [Tho81], and that its inability to express all ω-regular expressions makes LTL inadequate for numerous important tasks. For example, in *compositional model checking*, we verify a system by checking assume-guarantee specifications on its components. The specification $\langle\psi\rangle M \langle\varphi\rangle$ states that a composition that contains M and satisfies ψ, also satisfies φ. The assumption ψ can refer only to propositions observed by M, and LTL is not expressive enough to specify it [LPZ85][1]. Similarly, full ω-regularity is required in [HT97] in order to specify a network of communicating agents. The recognition that the specification language should be able to specify all ω-regular properties has led to several other extensions of LTL. This includes augmenting LTL with quantification over atomic propositions, resulting in *QLTL* [LPZ85, SVW87, MP92], and augmenting LTL with fixed-point operators, resulting in the linear μ-calculus [BB87, Var88]. These suggestions, however, are not very appealing in practice: formulas of QLTL and the linear μ-calculus are very hard to understand, and the satisfiability problem for QLTL is non-elementary [SVW87, Mey75].

Recall that Vardi and Wolper suggested to use ω-automata as temporal connectives [VW94]. They survey different types of automata. In particular, the logic ETL$_r$ uses nondeterministic Büchi automata, and it enables the specification of all ω-regular properties. ETL$_r$ combines two

[1] The reason is that the assumption needs to refer to locations in the interaction between M and its environment, which cannot be done by a star-free ω-regular expression.

perspectives of system specification: the operational perspective (finite-state machines) and the behavioral perspective (temporal operators). This makes ETL_r, as well as related logics, appealing in practice (cf. [BBL98, AFG$^+$01]). Moreover, unlike QLTL, the satisfiability problem for ETL_r is PSPACE-complete.

The logic ETL_r still suffers from two limitations. First, it lacks temporal operators that can refer to the past. While past temporal operators do not add expressive power to LTL, they make the specification of many behaviors much more convenient [LPZ85] [2]. This convenience is reflected in the fact that the best known translation of PLTL, which extends LTL with past temporal operators, to LTL involves a non-elementary blow up [Gab87]. Also, as mentioned above, in assume-guarantee reasoning in compositional model checking, the assumptions refer only to propositions observed by the component. In PLTL we can refer to the history of the computation, which is as strong as referring to locations in the interaction between the component and its environment [BK85, Pnu85, LPZ85]. To quote from Pnueli: "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_r" [Pnu85]. The second limitation of ETL_r follows from the limited structure of nondeterministic automata. Unlike LTL, whose syntax contains both disjunctions and conjunctions, runs of nondeterministic automata are treated purely disjunctively. Modelling of conjunctions by nondeterministic automata involves a blow up of the state space and results in automata whose structure is different from the structure of equivalent LTL formulas. We would like to use automata that preserve as much as possible the structure of the formulas.

In this paper we describe and study the logic ETL_{2a}, which removes both limitations. The extension of temporal logic with past is analogous to an extension of automata with bidirectional movement. *Two-way* automata can traverse the input word back and forth (technically, the transition function of two-way automata maps a state and a letter to a set of pairs, where each pair specifies both the next state and the direction to which the reading head of the automaton proceeds). Just like PLTL is not more expressive than LTL, two-way automata are not more expressive than conventional one-way automata. Also, as in the temporal-logic paradigm, it is often more convenient to define languages using two-way automata, and the convenience is reflected in their succinctness. For example, the translation of nondeterministic two-way Büchi automata to nondeterministic one-way Büchi automata involves an exponential blow-up [GH96]. So, our ETL_{2a} is going to have two-way Büchi automata as its temporal operators. In addition, the automata are going to be *alternating*[3].

A deterministic automaton has a single run over an input word. A nondeterministic automaton may have many runs, and it accepts the word if one of them is accepting. This can be viewed as if the automaton operates in an existential mode. Dually, in a universal mode, a word is accepted if all the runs of the automaton on it are accepting. In an alternating automaton [BL80, CKS81], both existential and universal modes are allowed. The richer combinatorial structure of alternating automata makes them a convenient specification language. Formally, alternating Büchi automata are exponentially more succinct than nondeterministic Büchi automata [DH94]. In addition, the complementation of alternating Büchi automata is quadratic and simpler [KV97].

[2] For example, it is easy to specify the fact that grants are issued only upon requests using past temporal operators: $\Box(grant \rightarrow \ominus(\neg grant \mathcal{S} req))$, where $\ominus$ ("Yesterday") and $\mathcal{S}$ ("Since") are the past-time counterparts of "Next" and "Until"). The reader is encouraged to try and specify the behavior without past temporal operators.

[3] An earlier attempt to extend ETL with alternating automata is reported in [VW94]. That attempt, however, was somewhat ad-hoc and could not handle alternating Büchi automata.

Our interest in alternating automata is not merely theoretical. Alternating automata have recently been used in industrial projects. The Intel ForSpec compiler uses an intermediate language called *SPIF*, which is essentially a variant of ETL_a, using alteranting automata as temporal connectives. The ForSpec compiler translates *ForSpec Temporal Logic* (FTL) formulas [AFG$^+$01] into SPIF, and from SPIF into nondeterministic Büchi automata [AFF$^+$01]. We note, however, that the ability of FTL to refer to past events is very limited, because of the limitations of ETL_a. Using ETL_{2a}, it would be possible to extend FTL and SPIF to include reference to past. We also note that it has recently been shown how nondeterministic Boolean decision diagrams (BDDs) can be used for maintaining sets of states in order to reason about alternating automata [Fin01]. Thus, we believe that ETL_{2a} is interesting both theoretically and practically.

We note that the succinctness of two-way automata stays valid in the framework of alternating automata: Birget has shown that two-way alternating automata on finite words are exponentially more succinct than one-way alternating automata on finite words [Bir93], and it is not hard to extend his result to Büchi automata [Pit00]. Also, the succinctness of alternating automata is valid in the framework of two-way automata: two-way alternating Büchi automata are exponentially more succinct than two-way nondeterministic Büchi automata [GH96]. So, ETL_{2a} extends ETL_r in two important aspects. On the other hand, the two succinctness results are not additive: there is an exponential translation of two-way alternating Büchi automata to one-way nondeterministic Büchi automata [Var98].

In the automata-theoretic approach to verification, we reduce questions about systems and their behavior to questions about automata [VW94]. Given a formal specification ψ, we construct a nondeterministic Büchi automaton $\mathcal{A}_\psi$ that accepts exactly the set of words that satisfy ψ. In order to check if ψ is satisfiable, we check whether the language of $\mathcal{A}_\psi$ is nonempty. In order to verify a system with respect to ψ, we check that the language of the system is contained in the language of $\mathcal{A}_\psi$. Following this approach, we would like to construct, given an ETL_{2a} formula ψ, a nondeterministic Büchi automaton that accepts exactly the set of words that satisfy ψ.

The construction proceeds in tho stages. We first translate an ETL_{2a} formula ψ to a two-way alternating *hesitant* automaton. Alternating hesitant automata are an extension of alternating weak automata [MSS86], and they combine the Büchi and its dual co-Büchi acceptance condition. Recall that the complementation problem for alternating Büchi automata is quadratic. On the other hand, complementing an alternating Büchi automaton to a co-Büchi alternating automaton can be done by dualizing the transition function and the acceptance condition. Consequently, the combination of both conditions leads to a linear translation of ETL_{2a} to two-way alternating hesitant automata. In the second stage we translate the two-way alternating hesitant automaton to a one-way nondeterministic Büchi automaton. For that, we first remove the hesitation and get a Büchi automaton, and then combine techniques for removing alternation [MS95] with techniques for removing bidirectionality [Var88]. The fact we deal with hesitant word automata makes the procedure much simpler than the one required for the translation of two-way alternating parity tree automata to one-way nondeterministic parity tree automata [Var98]. All in all, given an ETL_{2a} formula ψ, the nondeterministic Büchi automaton $\mathcal{A}_\psi$ has $2^{O(|\psi|^2)}$ states. It follows that the model-checking and the satisfiability problems for ETL_{2a} can be solved in polynomial space. Matching lower bounds are easy to show, hence the problems are PSPACE-complete, as are the ones for ETL_r or LTL [SC85]. It follows that the in spite of the succinctness of two-way and alternating automata, the advantages of ETL_{2a} are obtained essentially for free.

2 Definitions

2.1 Automata

For a finite alphabet Σ, a *word* $w \in \Sigma^\omega$ is an infinite sequence of letters from Σ. We denote by w_i the *i-th* letter of w.

Nondeterministic automata A *nondeterministic automaton* is $A = \langle \Sigma, Q, Q_0, \rho, F \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $\rho : Q \times \Sigma \to 2^Q$ is a transition function, and $F \subseteq Q$ is an acceptance condition. A *run* of A on a word $w \in \Sigma^\omega$ is an infinite sequence $r = q_0, q_1, \ldots$, where $q_0 \in Q_0$ and for all $i \geq 0$, we have $q_{i+1} \in \rho(q_i, w_i)$. Let $inf(r)$ denote the set of all states occurring infinitely often in r. Formally, $inf(r) = \{s \mid s = q_i$ for infinitely many i's$\}$. We consider two types of acceptance conditions *Büchi* and *co-Büchi*. A run of a Büchi automaton is *accepting* if it visits states from F infinitely often; i.e., $inf(r) \cap F \neq \emptyset$. A run of a co-Büchi automaton is *accepting* if it visits states from F only finitely often; i.e., $inf(r) \cap F = \emptyset$.

Hesitant automata combine the Büchi and the co-Büchi acceptance conditions. They extend *weak automata* [MSS86] by a richer acceptance condition. A hesitant automaton $A = \langle \Sigma, B, C, Q_0, \rho, F \rangle$ is a nondeterministic automaton such that the set of states $Q = B \cup C$ is the disjoint union of a set B of *Büchi states* and a set C of *co-Büchi states*. In addition, there is a partition of Q into disjoint sets, such that for each set S in the partition, either $S \subseteq B$, in which case S is a *Büchi set*, or $S \subseteq C$, in which case S is a *co-Büchi set*. For a state $q \in Q$, let $[q]$ denote the set of states in q's set in this partition. There exists a partial order $\leq$ on the collection of the sets such that for every two states q and q' for which q' occurs in $\delta(q, \sigma)$, for some $\sigma \in \Sigma$, we have $[q'] \leq [q]$. Thus, transitions from a state in a set S lead to states in either the same set or a lower one. It follows that a run r of a hesitant automaton ultimately gets trapped within some set S in the partition. The run r is *accepting* iff either $S \subseteq B$ is a Büchi set and $inf(r) \cap F \neq \emptyset$ or $S \subseteq C$ is a co-Büchi set and $inf(r) \cap F = \emptyset$. Thus, a run of a nondeterminisitic hesitant automaton may switch between Büchi and co-Büchi sets, yet eventually it stays forever in some set, and acceptance is determined according to the classification of this set. Note that if $C = \emptyset$, then A is a Büchi automaton, and that if $B = \emptyset$, then A is a co-Büchi automaton.

An automaton A *accepts* a word w if there exists an accepting run of A on w. Otherwise, A *rejects* w. The *language* of A, denoted $L(A)$, is the set of all words accepted by A. The *complementary language* of A is the set $\Sigma^\omega \setminus L(A)$ of all words w rejected by A.

Alternating automata For a set Q, we denote by $B^+(Q)$ the set of all positive Boolean formulas over Q, where we also allow **true** and **false**. We say that a set $Q' \subseteq Q$ *satisfies* a formula $\theta \in B^+(Q)$ (denoted $Q' \models \theta$) if by assigning true to all members of Q' and false to all members of $Q \setminus Q'$, the formula θ evaluates to true. Note that the formula **true** is satisfied by the empty set and the formula **false** cannot be satisfied. Given a formula $\theta \in B^+(Q)$, the *dual* of θ, denoted by $\widetilde{\theta}$, is obtained from θ by switching $\wedge$ and $\vee$, and switching **true** and **false**.

A *tree* is a set $T \subseteq \mathbb{N}^*$ such that if $x \cdot c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ϵ is the *root* of T. For every $x \in T$, the nodes $x \cdot c$ where $c \in \mathbb{N}$ are the *children* of x, the nodes $x \cdot y$ where $y \in \mathbb{N}^*$ are the *successors* of x. A node is a *leaf* if it has no children. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\epsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. Given an alphabet Σ, a Σ-*labeled tree* is a pair $\langle T, r\rangle$, where T is a tree and $r : T \rightarrow \Sigma$ maps each node of T to a letter in Σ.

An *alternating automaton* is $A = \langle \Sigma, Q, q_0, \rho, F\rangle$, where Σ, Q, and F are as in nondeterministic automata, q_0 is a unique initial state, and $\rho : Q \times \Sigma \rightarrow B^+(Q)$ is the transition function. We can say that a nondeterministic automaton accepts a word $w = w_0 \cdot w_1 \cdot w_2 \cdots$ from state s if it accepts the suffix $w_1 \cdot w_2 \cdots$ from one of the states in $\rho(s, w_0)$. In alternating automata, we allow posing both existential and universal demands on the suffix of the word. For example, if $\rho(s, a) = s_1 \wedge s_2 \vee s_3$, then A accepts a word starting with a from state s if it accepts the suffix of the word from both s_1 and s_2, or it accepts the suffix from s_3. For that, A sends to the suffix either two copies of itself, in states s_1 and s_2, or a single copy, in state s_3.

A *run* of an alternating automaton on a word $w \in \Sigma^\omega$ is a Q-labeled tree $\langle T, r\rangle$, where $r(\epsilon) = q_0$ and for all $x \in T$ the (possibly empty) set $\{r(x \cdot c) \mid c \in \mathbb{N} \text{ and } x \cdot c \in T\}$ satisfies the formula $\rho(r(x), w_{|x|})$. For a path π in the tree T, let $inf(r|\pi)$ denote the set of all states occurring infinitely often along that path in r, formally $inf(r|\pi) = \{s \mid s = r(x) \text{ for infinitely many } x \text{ in } \pi\}$. We consider alternating Büchi and co-Büchi automata. A run of an alternating Büchi automaton is *accepting* if for all infinite paths π in T, we have $inf(r|\pi) \cap F \neq \emptyset$. A run of an alternating co-Büchi automaton is *accepting* if for all infinite paths π in T we have $inf(r|\pi) \cap F = \emptyset$.

Two-way alternating automata A *2-way alternating automaton* is $A = \langle \Sigma, Q, q_0, \rho, F\rangle$, where Σ, Q, q_0, and F are as in alternating automata, and the transition function is $\rho : Q \times \Sigma \rightarrow B^+(\{-1, 0, 1\} \times Q)$. Alternating automata allowed us to pose both existential and universal demands on the suffix of the word. Two-way automata allow us to pose demands also on the prefix of the word. Technically, when the reading head of A is on the i-th position of w, it can move to locations $i-1$, i, and $i+1$. For example, $\rho(s_0, a) = (-1, s_1) \wedge (1, s_2) \vee (0, s_3)$ means that when the automaton is in state s_0 reading the letter a in location i, it can either send a copy in state s_1 to location $i-1$ and a copy in state s_2 to location $i+1$, or stay in location i in the state s_3. If $i = 0$, the automaton must choose the second option.

A *run* of A on a word $w \in \Sigma^\omega$ is a $(Q \times \mathbb{N})$-labeled tree $\langle T, r\rangle$, where $r(\epsilon) = (q_0, 0)$ and for all $x \in T$ with $r(x) = (r, k)$, the set $\{(q, \Delta) \mid c \in \mathbb{N},\ x \cdot c \in T, \text{ and } r(x \cdot c) = (q, k + \Delta\}$ satisfies the formula $\rho(r, w_k)$. For a path π, the set $inf(r|\pi)$ is defined as in alternating automata, thus $inf(r|\pi) = \{s \mid \text{there are infinitely many nodes } x \in \pi \text{ with } r(x) \in \{s\} \times \mathbb{N}\}$. A run of a 2-way alternating Büchi automaton is *accepting* if for all infinite paths π in T we have $inf(r|\pi) \cap F \neq \emptyset$ and a run of a *2-way alternating co-Büchi automaton* is accepting if for all infinite paths π in T we have $inf(r|\pi) \cap F = \emptyset$.

A 2-way alternating hesitant automaton $A = \langle \Sigma, B, C, q_0, \rho, F\rangle$ obeys the same restrictions as a nondeterministic hesitant automaton. Namely, the state set $Q = B \cup C$ is the union of Büchi and co-Büchi sets, there is a partition of the state set and a partial order that restricts the transition function. It follows that every infinite path in a run tree of a 2-way alternating hesitant automaton ultimately gets trapped within some $S \times \mathbb{N}$, for a set S in the partition. The run $\langle T, r\rangle$ is accepting if for every infinite path π in T, either $S \subseteq B$ and $inf(r|\pi) \cap F \neq \emptyset$, or $S \subseteq C$ and $inf(r|\pi) \cap F = \emptyset$.

Note that a 1-way alternating automaton can be viewed as a 2-way alternating automaton whose transition function is restricted to formulas from $B^+(\{1\} \times Q)$. Also, a nondeterministic automaton can be viewed as an alternating automaton whose transitions are restricted to disjunctions over the set Q. Given an automaton $A = \langle \Sigma, Q, q_0, \rho, F\rangle$ and a state $q \in Q$, we denote by A^q the automaton with initial state q; i.e, is $A^q = \langle \Sigma, Q, q, \rho, F \rangle$.

Given a 2-way alternating Büchi automaton $A = \langle \Sigma, Q, q_0, \rho, F \rangle$, the dual of A is the co-Büchi automaton $\widetilde{A} = \langle \Sigma, Q, q_0, \widetilde{\rho}, F \rangle$, where $\widetilde{\rho}(s,a)$ is the dual of $\rho(s,a)$. The automata A and $\widetilde{A}$ accept complementary languages [MS87]; i.e. $L(\widetilde{A}) = \Sigma^\omega \setminus L(A)$. Given an alternating hesitant automaton $A = \langle \Sigma, B, C, q_0, \rho, F \rangle$, the dual of A is the alternating hesitant automaton $\widetilde{A} = \langle \Sigma, C, B, q_0, \widetilde{\rho}, F \rangle$, where the set of Büchi states and the set of co-Büchi states switch roles. Again, $\widetilde{A}$ accepts the complementary language of A. Clearly, $\widetilde{\widetilde{A}}$ is A again.

We denote the different types of automata by four-symbol acronyms in $\{1,2\} \times \{D,N,A\} \times \{B,C,H\} \times \{W\}$, where the first symbol describes whether the automaton is 2-way or 1-way, (for 1-way automata we often omit the 1), the second symbol describes the branching mode of the automaton (deterministic, nondeterministic, or alternating), the third symbol describes the type of acceptance condition (Büchi, co-Büchi or hesitant), and the last symbol indicates that the automaton runs over words. For example, 1DBW denotes 1-way deterministic Büchi automata, as well as the set of ω-regular languages that can be recognized by a deterministic Büchi word automaton.

2.2 Temporal logic

Linear Temporal Logic The linear temporal logic LTL extends propositional logic by temporal operators like always ($\Box$), eventually ($\Diamond$), until (U), and next-time ($\bigcirc$) [Pnu81]. The semantics of LTL is defined with respect to infinite words in $(2^{AP})^\omega$, for the set AP of atomic propositions. For example, the formula $\Box p$ (always p) is satisfied over words in all of whose letters contain the atom p. For full syntax and semantics see [Eme90].

Extended Temporal Logic As mentioned above, Vardi and Wolper suggested to increase the expressive power of LTL by using 1NBW as temporal connectives [VW94]. Suppose the alphabet of an 1NBW A is the set 2^{AP}. The 1NBW A defines a set of sequences of truth assignments to the propositions. We can view A as a formula that is satisfied by exactly all the words accepted by A. The formal definition is a bit more complex, as automata are allowed to use other formulas as part of their alphabet and not only propositions. Below we describe the definition of ETL$_r$ as defined in [VW94].

We start with the syntax. Formulas are defined with respect to a set AP of atomic propositions as follows.

- Every proposition $p \in AP$ is a formula.
- If φ_1 and φ_2 are formulas, then $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, and $\varphi_1 \wedge \varphi_2$ are formulas.
- For every 1NBW $A = \langle \Sigma, Q, \rho, Q_0, F \rangle$ with $\Sigma = \{a_1, \ldots, a_n\}$, if $\varphi_1, \ldots, \varphi_n$ are formulas, then $A(\varphi_1, \ldots, \varphi_n)$ is a formula.

The *semantics* of ETL$_r$ is defined with respect to pairs $(\pi, i) \in (2^{AP})^\omega \times \mathbb{N}$, of words and locations. Consider an 1NBW $A = \langle \Sigma, Q, Q_0, \rho, F \rangle$. A *run* of a formula $A(\varphi_1, \ldots, \varphi_n)$ over a word π starting at point i, is an infinite sequence $\sigma = q_0, q_1, \ldots$ of states from Q, such that $q_0 \in Q_0$ and for all $k \geq 0$, there is some $a_j \in \Sigma$ such that $(\pi, i+k) \models \varphi_j$ and $q_{k+1} \in \rho(q_k, a_j)$. The run is *accepting* if $inf(r) \cap F \neq \emptyset$.

We use $(\pi, i) \models \psi$ to indicate that the word π in the location i satisfies the formula ψ. For a word $\pi \in (2^{AP})^\omega$ and a location $i \in \mathbb{N}$, the relation $\models$ is defined as follows.

- For a proposition $p \in AP$, we have $(\pi, i) \models p$ iff $p \in \pi_i$.
- $(\pi, i) \models \neg\varphi_1$ iff not $(\pi, i) \models \varphi_1$.
- $(\pi, i) \models \varphi_1 \vee \varphi_2$ iff $(\pi, i) \models \varphi_1$ or $(\pi, i) \models \varphi_2$.
- $(\pi, i) \models \varphi_1 \wedge \varphi_2$ iff $(\pi, i) \models \varphi_1$ and $(\pi, i) \models \varphi_2$.
- $(\pi, i) \models A(\varphi_1, \ldots, \varphi_n)$ iff there is an accepting run of $A(\varphi_1, \ldots, \varphi_n)$ over π starting at i.

Consider for example the 1NBW $A = \langle \Sigma, Q, Q_0, \rho, F \rangle$, where $\Sigma = \{a, b\}$, $Q = \{q_0, q_1\}$, $\rho(q_0, a) = \rho(q_1, a) = \{q_0\}$, $\rho(q_0, b) = \rho(q_1, b) = \{q_1\}$, and $Q_0 = F = \{q_1\}$. The state q_1 is visited exactly when A reads the letter b. A run of A is accepting if it visits state q_1 infinitely often. Hence, the ETL$_r$ connective $A(\neg p, p)$, where p is a proposition, is true iff p is true infinitely often. It is equal to the LTL formula $\Box\Diamond p$. As another example, consider the formula $\psi = \Box(\mathit{grant} \rightarrow \ominus\, (\neg \mathit{grant} \mathcal{S} \mathit{req}))$ stating that grants are issued only upon requests. We describe an equivalent ETL$_r$ formula for it. Consider the 1NBW $A = \langle \Sigma, Q, Q_0, \rho, F \rangle$, where $\Sigma = \{a, b, c, d\}$, $Q = \{q_0, q_1\}$, $\rho(q_0, a) = \{q_0\}$, $\rho(q_0, b) = \{q_1\}$, $\rho(q_1, c) = \{q_1\}$, $\rho(q_1, d) = \{q_0\}$, $Q_0 = \{q_0\}$, and $F = \{q_0, q_1\}$. Note that all the infinite runs of A are accepting. The state q_0 corresponds to a configuration in which no requests are pending. The state q_1 corresponds to a configuration in which there is at least one request pending. Accordingly, the ETL$_r$ formula $A(\neg \mathit{req} \wedge \neg \mathit{grant}, \mathit{req} \wedge \neg \mathit{grant}, \mathit{grant} \rightarrow \mathit{req}, \mathit{grant} \wedge \neg \mathit{req})$ is equivalent to ψ.

Extending temporal logic with 2-way alternating automata We now define formally the logic ETL$_{2a}$. The logic ETL$_{2a}$ extends ETL$_r$ by having 2-way alternating automata as its temporal connectives. Complementing the transition function of alternating automata is very simple. Hence, by allowing both Büchi and co-Büchi acceptance conditions, we can make the complementation of the temporal connectives simple. Accordingly, ETL$_{2a}$, uses both 2ABW and 2ACW as automata connectives. Runs of formulas that are automata connectives are defined as follows.

Consider a 2-way alternating automaton $A = \langle \Sigma, Q, q_0, \rho, F \rangle$. A *run* of the formula $A(\varphi_1, \ldots, \varphi_n)$ over a word w starting at point i, is a finite or infinite $(Q \times \mathbb{N})$-labeled tree $\langle T, r \rangle$ such that $r(\epsilon) = (q_0, i)$ and for all $x \in T$ with $r(x) = (q, k)$, there is some $a_j \in \Sigma$ such that $(\pi, k) \models \varphi_j$ and the (possibly empty) set $P = \{(q', \Delta) \mid \text{There is a child } y \text{ of } x \text{ in } T \text{ such that } r(y) = (q', k + \Delta)\}$ satisfies the transition $\rho(r(x), a_j)$. Intuitively, the children of x are labeled by the states of A and the locations in w from which the copies of the automaton should start running. Note that as $\varphi_1, \ldots, \varphi_n$ are not mutually exclusive, different copies may choose different formulas. If the automaton is a 2ABW, the run is *accepting* if for all infinite paths π of T we have $\mathit{inf}(r|\pi) \cap F \neq \emptyset$. If the automaton is a 2ACW, the run is *accepting* if for all paths π of T we have $\mathit{inf}(r|\pi) \cap F = \emptyset$. When not important or clear from the context, we often write the formula $A(\varphi_1, \ldots, \varphi_n)$ as A.

Recall the formula ψ stating that grants are issued only upon requests. We now describe an ETL$_{2a}$ formula for it. Consider the 2ABW $A = \langle \Sigma, Q, q_0, \rho, F \rangle$, where $\Sigma = \{a, b, c, d\}$, $Q = \{q_0, q_1\}$, $\rho(q_0, a) = (q_0, 1)$, $\rho(q_0, b) = (q_0, 1) \wedge (q_1, -1)$, $\rho(q_1, b) = \textbf{false}$, $\rho(q_1, c) = \textbf{true}$, $\rho(q_1, d) = (q_1, -1)$, and $F = \{q_0\}$. The formula $A(\neg \mathit{grant}, \mathit{grant}, \mathit{req}, \neg \mathit{req} \wedge \neg \mathit{grant})$ is equivalent to ψ. To see that, note that whenever A visits the state q_0 and reads a letter containing a grant, it sends a copy that goes backwards, expecting a request before it comes across a grant. Also, as $q_1 \notin F$, a request has to be eventually found. The ETL$_{2a}$ formula has very much the same structure as the PLTL formula.

Similar to other logics, handling ETL$_{2a}$ is easier in positive normal form, where negations are pushed inward using De-Morgan laws. In an ETL$_{2a}$ formula in positive normal form, negations apply to automata and atomic propositions only. For a formula ψ, let $\widetilde{\psi}$ denote $\neg\psi$ in positive normal form[4].

Given an ETL$_{2a}$ formula ψ in a positive normal form, the *closure* of ψ, denoted $cl(\psi)$ includes all the subformulas of ψ and their complements. This includes formulas of the form A^q, for an automata connective A and a state q of it. For simplicity, we assume that the state sets of the automata connectives in ψ are pairwise disjoint, thus we can denote the subformula $A(\psi_1, \ldots, \psi_n)$ by $q_0(\psi_1, \ldots, \psi_n)$, for the initial state q_0 of A. Similarly, we denote $A^q(\psi_1, \ldots, \psi_n)$ by $q(\psi_1, \ldots, \psi_n)$. When $\psi_1, \ldots, \psi_n$ are clear from the context, we write just q_0 or q, respectively. Formally, the set $cl(\psi)$ is the minimal set satisfying all the following.

- $\psi \in cl(\psi)$,
- If $\psi_1 \in cl(\psi)$, then $\widetilde{\psi_1} \in cl(\psi)$.
- if $\psi_1 \wedge \psi_2 \in cl(\psi)$ then $\{\psi_1, \psi_2\} \subseteq cl(\psi)$,
- if $\psi_1 \vee \psi_2 \in cl(\psi)$ then $\{\psi_1, \psi_2\} \subseteq cl(\psi)$, and
- if $q_0(\psi_1, \ldots, \psi_n) \in cl(\psi)$, for a 2ABW or a 2ACW $A = \langle \Sigma, Q, q_0, \rho, F \rangle$, then $\{\psi_1, \ldots, \psi_n\} \subseteq cl(\psi)$, and for all $q \in Q$, we have $q(\psi_1, \ldots, \psi_n) \in cl(\psi)$.

Note that the formulas in $cl(\psi)$ are in positive normal form. Thus, negation applies to atomic propositions and automata connectives only.

Consider again the formula $\varphi = A(\neg grant, grant, req, \neg req \wedge \neg grant)$ discussed above. The closure of φ is

$$cl(\varphi) = \{q_0, q_1, \neg q_0, \neg q_1, grant, \neg grant, req, \neg req, \neg req \wedge \neg grant, req \vee grant\}$$

For a formula ψ, the *models* of the formula is the set $L(\psi)$ of all infinite words $w \in (2^{AP})^\omega$ that satisfy the formula.

3 Decision procedures for ETL$_{2a}$

In this section we solve the satisfiability and model-checking problems for ETL$_{2a}$. Given an ETL$_{2a}$ formula ψ, we construct a 1NBW $\mathcal{A}_\psi$ such that $\mathcal{A}_\psi$ accepts exactly all the words satisfying ψ.

[4] Consider an automaton A. Note that the formula $\psi = \neg A(\varphi_1, \ldots, \varphi_n)$ is not equivalent to the formula $\tilde{A}(\varphi_1, \ldots, \varphi_n)$, where $\tilde{A}$ is the dual of A. This is because both A and $\tilde{A}$ treat the formulas $\varphi_1, \ldots, \varphi_n$ existentially. Indeed, for both automata, the transition from a state to its successor involves a choice of some φ_i. In order for ψ to be false, all the runs of A should be rejected, thus $\tilde{A}$ should treat the formulas $\varphi_1, \ldots, \varphi_n$ universally. Universally in this case means that if φ_i holds then $\tilde{A}$ should take the transition corresponding to the letter a_i. This is why the positive normal form for ETL$_{2a}$ allows the application of negation to automata connectives.

The size of $\mathcal{A}_\psi$ is $2^{O(|\psi|^2)}$. Using $\mathcal{A}_\psi$, we show that both the satisfiability and the model-checking problems for ETL$_{2a}$ are PSPACE-complete. The construction of $\mathcal{A}_\psi$ proceeds in two stages, with 2AHW serving as an intermediate formalism.

3.1 From ETL$_{2a}$ to 2AHW

In this section we describe how to construct the intermediate 2AHW.

Theorem 3.1 *Given an ETL$_{2a}$ formula ψ of length n, there is a 2AHW $\mathcal{H}_\psi$ such that $L(\mathcal{H}_\psi) = L(\psi)$ and $\mathcal{H}_\psi$ has $O(n)$ states.*

Proof: Given a set Q of states, let $\neg Q = \{\neg q \mid q \in Q\}$. We define the function *dual* : $B^+(\{-1,0,1\} \times Q) \to B^+(\{-1,0,1\} \times \neg Q)$ as follows. For a formula $\theta \in B^+(\{-1,0,1\} \times Q)$, the formula $dual(\theta)$ is obtained from $\tilde{\theta}$ (the dual of θ) by replacing every atom of the form $(\Delta, q) \in \{-1,0,1\} \times Q$ by the atom $(\Delta, \neg q)$. So, $dual(\theta)$ switches $\vee$ and $\wedge$, and **true** and **false**, and also adds negationa in front of states in Q. For example, $dual(((-1,s) \wedge (0,t)) \vee (1,q)) = ((-1,\neg s) \vee (0,\neg t)) \wedge (1,\neg q)$.

Now, given an ETL$_{2a}$ formula ψ, we define $\mathcal{H}_\psi = \langle 2^{AP}, B, C, \psi, \eta, \alpha\rangle$, where $B \cup C = cl(\psi)$, and

- The set of Büchi states is

$$B = \begin{array}{l}\{q \mid A \text{ is a 2ABW connective in } \psi \text{ and } q \text{ is a state of } A\} \cup \\ \{\neg q \mid A \text{ is a 2ACW connective in } \psi \text{ and } q \text{ is a state of } A\}.\end{array}$$

  The set of co-Büchi states is $C = cl(\psi) \setminus B$. We note that the decision to include elements of $cl(\psi)$ that are not states of automata in C is arbitrary. Indeed, the transition from such states are to strict subformulas, and the automaton is not going to get trapped in a set associated with them.

  The partition of $cl(\psi)$ is as follows. For every state $s \in cl(\psi)$ such that s is not a state of an automaton, the Singleton $\{s\}$ is a set of the partition. For an automaton $A = \langle \Sigma, Q, q_0, \rho, F\rangle \in cl(\psi)$, all the states $\{q \mid q \in Q\}$ form a set in the partition, and all the states $\{\neg q \mid q \in Q\}$ form a set in the partition. The partial order $\leq$ on the sets is such that $[s'] \leq [s]$ iff $s' \in cl(s)$.

- The transition function $\eta : cl(\psi) \times 2^{AP} \to B^+(\{-1,0,1\} \times cl(\psi))$ is defined for every formula in $cl(\psi)$ and letter $a \in 2^{AP}$ as follows.

  - For a proposition $p \in AP$, we have $\eta(p,a) = \begin{cases} \textbf{true} & \text{if } p \in a \\ \textbf{false} & \text{if } p \notin a \end{cases}$
  - For a proposition $p \in AP$ we have $\eta(\neg p,a) = \begin{cases} \textbf{true} & \text{if } p \notin a \\ \textbf{false} & \text{if } p \in a \end{cases}$
  - $\eta(\psi_1 \wedge \psi_2, a) = (\psi_1, 0) \wedge (\psi_2, 0)$
  - $\eta(\psi_1 \vee \psi_2, a) = (\psi_1, 0) \vee (\psi_2, 0)$

- Consider a 2ABW or 2ACW $A(\psi_1,\ldots,\psi_n) \in cl(\psi)$, with $A = \langle\{a_1,\ldots,a_n\},Q,q_0,\rho,F\rangle$. For every $q \in Q$ we have $\eta(q(\psi_1,\ldots,\psi_n),a) = \bigvee_{i=1}^{n}[(\psi_i,0) \wedge \rho(q,a_i)]$
- Consider a 2ABW or 2ACW $A(\psi_1,\ldots,\psi_n) \in cl(\psi)$, with $A = \langle\{a_1,\ldots,a_n\},Q,q_0,\rho,F\rangle$. For every $q \in Q$ we have $\eta(\neg q(\psi_1,\ldots,\psi_n),a) = \bigwedge_{i=1}^{n}[(\widetilde{\psi_i},0) \vee dual(\rho(q,a_i))]$

The transition from states associated with an automaton $A(\psi_1,\ldots,\psi_n)$ makes sure that there is indeed an accepting run of the formula. For that, when the automaton is in state q of A, it checks that there is a formula ψ_i in $\psi_1,\ldots,\psi_n$ such that ψ_i holds in the current location (checked by sending the copy $(\psi_i,0)$), and that the formula A^q has an accepting run starting with the transition taken by reading a_i (checked by the copies sent by $\rho(q,a_i)$). In a dual manner, the transition from states associated with a formula $\neg A(\psi_1,\ldots,\psi_n)$ makes sure that there is no accepting run of A. For that, when the automaton is in state $\neg q$, it checks that for all formulas $\psi_1,\ldots,\psi_n$, either ψ_i is false (checked by sending the copy $(\widetilde{\psi_i},0)$), or that the formula A^q has no accepting run starting with the transition taken by reading a_i (checked by the copies sent by $dual(\rho(q,a_i))$).

It is easy to see that η respects the partial order on the partition of $B \cup C$.

- The acceptance condition is

$$\alpha = \{q, \neg q \mid q \in F, \text{ for an automaton connective } A = \langle\Sigma,Q,q_0,\rho,F\rangle \text{ in } \psi\}.$$

For a 2ABW A and state $q \in F$, we would like to visit q infinitely often, and indeed q is a Büchi state in α. On the othe hand, the transition from the state $\neg q$ is obtained by dualizing the transitions from q, we would like to visit it finitely often, and indeed it is a co-Büchi state in α. So, both q and $\neg q$ are members of α and the restriction as to whether they should be visited finitely or infinitely often is determined by their classification as Büchi and co-Büchi states, respectively. Similarly, when A is a 2ACW and $q \in F$, the state q is a co-Büchi state in α, and should be visited finitely often, and the state $\neg q$ is a Büchi state state in α and should be visited infinitely often.

It is left to prove that $L(\mathcal{H}_\psi) = L(\psi)$. The proof proceeds We prove that $L(\mathcal{H}_\psi) = L(\psi)$. The proof proceeds by induction on the structure of ψ. We assume that for a subformula $\varphi \in cl(\psi)$, if we start the 2NHW $\mathcal{H}_\psi$ running from state φ in location i in the word w, it accepts iff $(w,i) \models \varphi$.

The proof for propositions and Boolean connectives is immediate. We give the details for the case φ is an automaton or its negation.

We first need a notation. Given a labeled tree $\langle T,r\rangle$, a node $x \in T$, and a labeled tree $\langle T',r'\rangle$ by *attaching* $\langle T',r'\rangle$ *under* x we get the labeled tree $\langle T'',r''\rangle$ as follows. The tree $T'' = T \cup \{x \cdot a \cdot y \mid y \in T'\}$, where $a \in \mathbb{N}$ such that $x \cdot a \notin T$. The labeling function $r''(z) = r(z)$ for z in T and $r''(x \cdot a \cdot y) = r'(y)$ for $y \in T'$.

- Let $\varphi = A(\psi_1,\ldots,\psi_n)$, for $A = \langle\Sigma,Q,q_0,\rho,F\rangle$. Consider a word $w \in (2^{AP})^\omega$ and a location $k \in \mathbb{N}$.

  If $(w,k) \models A(\psi_1,\ldots,\psi_n)$, we know that there exists an accepting run tree of A on w. The labels of the tree T are from the set $Q \times \mathbb{N}$. We convert this run tree into a run tree of $\mathcal{H}_\psi$ starting at the letter w_k. For that, we attach run trees of $\mathcal{H}_{\psi_i}$.

Given an accepting run $\langle T, r\rangle$ of the formula ψ on the word w starting from letter w_k we modify it into a run tree $\langle T', r'\rangle$ of $\mathcal{H}_\psi$. Given a node $x \in T$ with label $r(x) = (q, l)$, where q is a state of A and l a location in the word w, we know that there is some i such that $(w, l) \models \psi_i$ and the set $\{(t, \Delta) \mid x \cdot a \in T \text{ and } r(x \cdot a) = (t, l + \Delta)\}$ satisfies the transition $\rho(q, a_i)$. So we attach under x the run tree of $\mathcal{H}_{\psi_i}$. It is easy to see that the resulting run tree is a valid and accepting run tree of $\mathcal{H}_\psi$.

The other direction is similar. Consider an accepting run tree of $\mathcal{A}_\psi$ starting from the letter w_k. From the induction hypothesis an accepting run of $\mathcal{H}_{\psi_i}$ starting from letter w_j exists iff $(w, j) \models \psi_i$. We can prune the tree to serve as a run tree of the formula ψ on the word w starting from k.

- Let $\varphi = \neg A(\psi_1, \ldots, \psi_n)$, for $A = \langle \Sigma, Q, q_0, \rho, F\rangle$. By the construction of $\mathcal{H}_\psi$, the 2NHW $\mathcal{H}_{\neg A(\psi_1,\ldots,\psi_n)}$ is equivalent to the dual $\widetilde{\mathcal{H}}_{A(\psi_1,\ldots,\psi_n)}$ of the 2NHW $\mathcal{H}_{A(\psi_1,\ldots,\psi_n)}$. By the induction hypothesis, $\mathcal{H}_{A(\psi_1,\ldots,\psi_n)}$ accepts all words that satisfy the formula $A(\psi_1, \ldots, \psi_n)$. Since $\widetilde{\mathcal{H}}_{A(\psi_1,\ldots,\psi_n)}$ and $\mathcal{H}_{A(\psi_1,\ldots,\psi_n)}$ accept complementary languages, we are done.

□

3.2 From 2AHW to 1NBW

In this section we describe how to transform the intermediate 2AHW to 1NBW. In [Var98], Vardi translates 2-way alternating parity tree automata to 1-way nondeterministic parity tree automata. Since $\mathcal{H}_\psi$ can be defined as a parity automaton, and since words are a special case of trees, one could use the transformation in [Var98]. We describe here a simpler and more direct construction. We first need some notations.

Consider a 2AHW $A = \langle \Sigma, Q, q_0, \eta, F\rangle$. A *restriction* of A is a set $\xi \in 2^{Q \times \{-1,0,1\} \times Q}$. For a restriction $\xi \subseteq Q \times \{-1, 0, 1\} \times Q$, we define $\mathit{state}(\xi) = \{u : (u, \Delta, u') \in \xi\}$. A *strategy* for A is an infinite sequence $\tau = \xi_0, \xi_1, \ldots$ of restrictions. We sometimes denote to ξ_i by $\tau(i)$. We say that the strategy τ is *on* a word w if $q_0 \in \mathit{state}(\xi_0)$, for all $i \in \mathbb{N}$ and states $q \in \mathit{state}(\xi_i)$, the set $\{(\Delta, q') | (q, \Delta, q') \in \xi_i\}$ satisfies $\eta(q, w_i)$, and for all $i \in \mathbb{N}$ and $(q, \Delta, q') \in \xi_i$ we have $q' \in \mathit{state}(\xi_{i+\Delta})$ (or $\eta(q', w_{i+\Delta}) = \mathit{true}$). Intuitively, a strategy suggests at each location i, a possible way for satisfying the transition function.

Lemma 3.2 *Consider a 2AHW $A = \langle \Sigma, Q, q_0, \eta, F\rangle$ with n states. There is a 1DBW A' over the alphabet $\Sigma \times 2^{Q \times \{-1,0,1\} \times Q}$ such that A' has $2^{O(n)}$ states and it accepts a word $(w_0, \xi_0) \cdot (w_1, \xi_1) \cdots$ iff $\xi_0, \xi_1, \cdots$ is a strategy for A on $w_0, w_1, \cdots$.*

Proof: The intuition is quite simple. When reading (w_i, ξ_i), the automaton A' has to remember two sets. The set of states that ξ_i restricts ($\mathit{state}(\xi_i)$) and the set of states that ξ_i promises that have a strategy from ξ_{i+1} (if $(q, 1, q') \in \xi_i$ then ξ_i promises that q' has a strategy from ξ_{i+1}). It then checks that the states that are promised by ξ_{i+1} that have a strategy from ξ_i are indeed restricted by ξ_i and that all promises of ξ_i are indeed fulfilled. The local requirements, that the strategy fulfills the transition of A and that states that should be restricted by ξ_i are indeed restricted need no memory in order to be checked.

Formally, $A' = \langle \Sigma \times 2^{Q\times\{-1,0,1\}\times Q}, S, s_0, \rho, S\rangle$ where $S = 2^Q \times 2^Q \cup \{s_0\}$. The states of A' are pairs of sets of states of A and the acceptance condition is trivial. Thus, every infinite run of A' is accepting.

We use the following notation. For every letter $w \in \Sigma$ and restriction $\xi \in 2^{Q\times\{-1,0,1\}\times Q}$ the set $appear(w,\xi) = states(\xi) \cup \{q \mid \eta(q,w) = true\}$ is the set of states restricted by ξ, and for $\Delta \in \{-1,0,1\}$ the set $promise(\Delta,\xi) = \{q' \mid (q,\Delta,q') \in \xi \text{ for some } q \in Q\}$ is the set of states that should have a restriction in the direction Δ. The transition of A' is defined as follows.

$$\rho(s_0,(w,\xi)) = \left\{ (appear(w,\xi), promise(1,\xi)) \;\middle|\; \begin{array}{l} q_0 \in appear(w,\xi) \\ promise(0,\xi) \subseteq appear(w,\xi) \\ promise(-1,\xi) = \emptyset \\ \forall\, q \in state(\xi),\ \{(\Delta',q') \mid (q,\Delta',q') \in \xi\} \models \eta(q,w) \end{array} \right\}$$

$$\rho((A,P),(w,\xi)) = \left\{ (appear(w,\xi), promise(1,\xi)) \;\middle|\; \begin{array}{l} P \subseteq appear(w,\xi) \\ promise(0,\xi) \subseteq appear(w,\xi) \\ promise(-1,\xi) \subseteq A \\ \forall\, q \in state(\xi),\ \{(\Delta',q') \mid (q,\Delta',q') \in \xi\} \models \eta(q,w) \end{array} \right\}$$

Clearly, A' is deterministic and it accepts a word $(w_0,\xi_0)\cdot(w_1,\xi_1)\cdots$ iff $\xi_0,\xi_1,\ldots$ is a strategy of A on $w_0,w_1,\ldots$. □

A *path* in a strategy τ is a finite or infinite sequence $(0,q_0),(i_1,q_1),(i_2,q_2),\ldots$ of pairs from $\mathbb{N}\times Q$ such that either the path is infinite, in which case for all $j \geq 0$, there is $\Delta_j \in \{-1,0,1\}$ such that $(q_j,\Delta_j,q_{j+1}) \in \tau(i_j)$ and $i_{j+1} = i_j + \Delta_j$, or the path is finite $(0,q_0),\ldots,(i_m,q_m)$, in which case for all $0 \leq j < m$, there is $\Delta_j \in \{-1,0,1\}$ such that $(q_j,\Delta_j,q_{j+1}) \in \tau(i_j)$, $i_{j+1} = i_j + \Delta_j$, and $\eta(q_m,w_{i_m}) = true$. An infinite path is *accepting* if it gets trapped in B and visits $\mathbb{N} \times F$ infinitely often or if it gets trapped in C and visits $\mathbb{N} \times F$ finitely often. A finite path is always *accepting*. We say that τ is *winning* if all infinite paths in τ are accepting. Otherwise, τ is *losing*. It is not hard to see that a 2AHW accepts a word iff it has a winning strategy on the word (c.f., [MS95, Var98, KV00]).

Lemma 3.3 *Consider a 2AHW $A = \langle \Sigma, Q, q_0, \eta, F\rangle$ with n states. There is a 2NBW A' over the alphabet $2^{Q\times\{-1,0,1\}\times Q}$ such that A' has $O(n)$ states and it accepts exactly all the losing strategies for A.*

Proof: We first define a 2NHW A'' that accepts exactly all the losing strategies of A. The automaton $A'' = \langle 2^{Q\times\{-1,0,1\}\times Q}, Q, q_0, \eta', F\rangle$, where the co-Büchi set of A'' is the Büchi set of A, the Büchi set of A'' is the co-Büchi set of A, and η' is defined for all $q \in Q$ and $\xi \in 2^{Q\times\{-1,0,1\}\times Q}$ as follows. Let $options(q,\xi) = \{(\Delta,q') \mid (q,\Delta,q') \in \xi\}$. Then,

$$\eta'(q,\xi) = \begin{cases} \textbf{false} & \text{If } options(q,\xi) = \emptyset \\ options(q,\xi) & \text{Otherwise.} \end{cases}$$

Intuitively, when the automaton A'' reads a strategy τ, it guesses a path in τ that is not accepting. Accordingly, A'' rejects when the strategy reaches a location in which the set of restrictions is empty (this corresponds to the case where the candidate path is finite). When the candidate path is infinite, it is not accepting in τ iff it does not satisfy the acceptance condition of A, which is why A'' dualizes A.

Now, it is easy to translate the 2NHW A'' to a 2NBW A' with a linear blow up: whenever we are in a co-Büchi set S, we can nondeterministically move to a copy of the set in which only states from $S \setminus F$ are present. □

The automaton A' in Lemma 3.3 uses its bidirectionality in order to follow the strategy τ. This enables us by remove alternation, but still leaves us with bidirectionality. To remove the latter, we have to blow up the state space:

Lemma 3.4 [Var88] *Given a 2NBW A with n states, we can construct a 1NBW A' with $2^{O(n^2)}$ states such that $L(A') = \Sigma^\omega \setminus L(A)$.*

Intuitively, the $2^{O(n^2)}$ blow up follows from the fact we have to remember, for each pair of states $\langle q, q' \rangle$, the set of states visited between subsequent visits of the automaton in q and q'. We can now combine Lemmas 3.2, 3.3, and 3.4 to obtain our goal.

Theorem 3.5 *Given a 2AHW $\mathcal{H}$ over Σ, we can construct a 1NBW $\mathcal{A}$ with $2^{O(n^2)}$ states such that $L(\mathcal{H}) = L(\mathcal{A})$.*

Proof: Let $\mathcal{H} = \langle \Sigma, Q, q_0, \eta, F \rangle$. By Lemma 3.2, we can construct a 1DBW A_1 over the alphabet $\Sigma \times 2^{Q \times \{-1,0,1\} \times Q}$ such that A_1 has $2^{O(n)}$ states and it accepts a word $(w_0, \xi_0) \cdot (w_1, \xi_1) \cdots$ iff $\xi_0, \xi_1, \ldots$ is on $w_0, w_1, \ldots$. Also, by Lemmas 3.3 and 3.4, we can construct a 1NBW A_2 such that A_2 accepts a word over the alphabet $\Sigma \times 2^{Q \times \{-1,0,1\} \times Q}$ iff its projection on $2^{Q \times \{-1,0,1\} \times Q}$ is an accepting strategy. The automaton $\mathcal{A}$ is the intersection of A_1 and A_2, projected on Σ. □

3.3 Automata-based decision procedures for ETL$_{2a}$

In this section we combine the constructions described in Sections 3.1 and 3.2 and use the resulting 1NBW for solving the satisfiability and model-checking problems for ETL$_{2a}$. First, Theorems 3.1 and 3.5 immediately imply the following.

Theorem 3.6 *Given an ETL$_{2a}$ formula ψ of length n, there is a 1NBW $\mathcal{A}_\psi$ such that $L(\mathcal{A}_\psi) = L(\psi)$ and $\mathcal{A}_\psi$ has $2^{O(n^2)}$ states.*

Once we construct $\mathcal{A}_\psi$, we can reduce satisfiability of ψ to nonemptiness of $\mathcal{A}_\psi$, and we can reduce model checking of a system S with respect to ψ to the language inclusion problem $L(S) \subseteq L(\mathcal{A}_\psi)$. (The system S is given as a finite state-transition graph, $L(S)$ is the set of all the words that S generates, and we say that S satisfies ψ if $(\pi, 0) \models \psi$ for all the words π that S generate.) As in LTL, we can use the fact that ETL$_{2a}$ is closed under negation and check the latter by checking the emptiness of the intersection $S \times \mathcal{A}_{\neg\psi}$ [VW94]. Since the nonemptiness problem for Büchi automata can be solved in NLOGSPACE, we have the following (the lower bounds follow immediately from the lower bounds on LTL [SC85] and the linear translation of LTL to alternating automata [Var96]).

Theorem 3.7 *The satisfiability and the model-checking problems for ETL_{2a} are PSPACE-complete.*

It follows that in spite of the succinctness of two-way and alternating automata, the advantages of ETL_{2a} are obtained essentially for free.

4 Discussion

We studied an extension of linear temporal logic with two-way alternating automata. The resulting logic ETL_{2a}, is as expressive as previous extensions of linear temporal logic with ω-regular automata, but the added strength of bidirectionality and alternation makes the logic substantially more convenient. The satisfiability and model-checking problems for ETL_{2a} are PSPACE-complete, as is the case with LTL or weaker extensions of LTL with automata. There have been two recent developments that make us optimistic about the practicality of ETL_{2a}: the development of symbolic procedures for handling alternating automata [Fin01], and the usage of alternating automata as an intermediate formalism at Intel [AFF$^+$01]. Using ETL_{2a}, it would be possible to extend this intermediate formalism to include convenient reference to past.

In this paper we considered the linear framework to verification. Branching temporal logic extends linear temporal logic with the path quantifiers A ("for all path") and E ("there exists a path"), and its formulas describe computation trees. The same limitation of LTL applies to its branching-time extension CTL*. Similar suggestions to extend the expressiveness of CTL* are studied in the literature. This includes both the extensions of the path formulas of CTL* with ω-regular word automata [VW84, CGK92], and the extension of the state formulas with ω-regular tree automata [MS85]. As in the linear framework, one can strengthen these extensions by using more powerful automata, in particular two-way and alternating ones. Since it is possible to remove bidirectionality and alternation also in the branching framework [Var98], our treatment of ETL_{2a} should work here as well. Its implementation, however, is going to be much more complicated in the branching framework.

References

[AFF$^+$01] R. Armoni, L. Fix, A. Flaisher, R. Gerth, T. Kanza, A. Landver, S. Mador-Haim, A. Tiemeyer, M.Y. Vardi, and Y. Zber. The ForSpec compiler. Submitted, 2001.

[AFG$^+$01] R. Armoni, L. Fix, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, A. Tiemeyer, E. Singerman, and M.Y. Vardi. The ForSpec temporal logic: A new temporal property-specification logic. Submitted, 2001.

[BB87] B. Banieqbal and H. Barringer. Temporal logic with fixed points. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, LNCS 398, pages 62–74. Springer-Verlag, 1987.

[BBG$^+$94] I. Beer, S. Ben-David, D. Geist, R. Gewirtzman, and M. Yoeli. Methodology and system for practical formal verification of reactive hardware. In *6th CAV*, LNCS 818, pages 182–193, Stanford, June 1994.

[BBL98] I. Beer, S. Ben-David, and A. Landver. On-the-fly model checking of RCTL formulas. In *10th CAV*, LNCS 1427, pages 184–194. Springer-Verlag, 1998.

[Bir93] J.C. Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical Systems Theory*, 26(3):237–269, 1993.

[BK85] H. Barringer and R. Kuiper. Hierarchical development of concurrent systems in a framework. In S.D. Brookes et al., editor, *Seminar in Concurrency*, Lecture Notes in Computer Science, Vol. 197, pages 35–61. Springer-Verlag, Berlin/New York, 1985.

[BL80] J.A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[CGK92] E.M. Clarke, O. Grumberg, and R.P. Kurshan. A synthesis of two approaches for verifying finite state concurrent systes. *Logic and Computation*, 2(5):605–618, 1992.

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[DH94] D. Drusinsky and D. Harel. On the power of bounded concurrency I: Finite automata. *Journal of the ACM*, 41(3):517–539, 1994.

[Eme90] E.A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, pages 997–1072, 1990.

[Fin01] B. Finkbeiner. Symbolic refinement checking with nondeterministic BDDs. In *TACAS*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[Fra92] N. Francez. *Program verification.* International Computer Science. Addison-Weflay, 1992.

[Gab87] D. Gabbay. The declarative past and imperative future. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, LNCS 398, pages 407–448. Springer-Verlag, 1987.

[GH96] N. Globerman and D. Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Computer Science*, 143:161–184, 1996.

[HT97] J. Henriksen and P. Thiagarajan. A product version of dynamic linear time temporal logic. In *Proc. 8th Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 45–58, Warsaw, 1997. Springer-Verlag.

[KV97] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symp. on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.

[KV00] O. Kupferman and M.Y. Vardi. μ-calculus synthesis. In *25th MFCS*, LNCS 1893, pages 497–507. Springer-Verlag, 2000.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218, Brooklyn, June 1985. Springer-Verlag.

[Mey75] A. R. Meyer. Weak monadic second order theory of successor is not elementary recursive. In *Proc. Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. Springer-Verlag, 1975.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, Berlin, January 1992.

[MS85] D.E. Muller and P.E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MS95] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[MSS86] D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *13th ICALP*, LNCS 226, Springer-Verlag, 1986.

[Pit00] N. Piterman. Extending temporal logic with ω-automata. M.Sc. Thesis, The Weizmann Institute of Science, Israel, http://www.wisdom.weizmann.ac.il/home/nirp/public_html/publications/msc_thesis.ps, 2000.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[Pnu85] A. Pnueli. In transition from global to modular temporal reasoning about programs. In K. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 123–144. Springer-Verlag, 1985.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *Journal ACM*, 32:733–749, 1985.

[SVW87] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[Tho81] W. Thomas. A combinatorial approach to the theory of ω-automata. *Information and Computation*, 48:261–283, 1981.

[Var88] M.Y. Vardi. A temporal fixpoint calculus. *15th POPL*, pages 250–259, San Diego, January 1988.

[Var96] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, Berlin, 1996.

[Var98] M.Y. Vardi. Reasoning about the past with two-way automata. In *25th ICALP*, LNCS 1443, pages 628–641. Springer-Verlag, Berlin, July 1998.

[VW84] M.Y. Vardi and P. Wolper. Yet another process logic. In *Logics of Programs*, volume 164, pages 501–512. Lecture Notes in Computer Science, Springer-Verlag, 1984.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.