REASONING ABOUT INFINITE COMPUTATION PATHS
(extended abstract)

Pierre Wolper†
Bell Laboratories

Moshe Y. Vardi‡
Stanford University

*A. Prasad Sistla**
Harvard University

Abstract

We investigate extensions of temporal logic by finite automata on infinite words. There are three different types of acceptance conditions (finite, looping and repeating) that one can give for these finite automata. This gives rise to three different logics. It turns out, however, that these logics have the same expressive power but differ in the complexity of their decision problem. We also investigate the addition of alternation and show that it does not increase the complexity of the decision problem.

1. Introduction

For many years, logics of programs were tools for reasoning about the input/output behavior of programs. When dealing with concurrent or non-terminating processes (like operating systems) there is, however, a need to reason about infinite computation paths. These are the sequences of states that the computation goes through. In the propositional case they can be viewed as infinite sequences of propositional truth assignments. In [Pn77], temporal logic was proposed to reason about such sequences. Later it was incorporated into the process logics of [Ni80] and [HKP80].

For reasoning about propositional truth assignments, propositional logic is a descriptively complete language, i.e., it can specify any set of propositional truth assignments. However, for reasoning about computation paths there is no a priori robust notion of descriptive completeness. In [GPSS80] propositional temporal logic (*PTL*) was shown to be expressively equivalent to the monadic first-order theory of $(N,<)$, the natural numbers with the less-than relation. This was taken as an indication that *PTL* is "descriptively complete", and that so are the process logics based on it [Ni80,HKP80].

The above claim is based on the assumption that first-order notions are all we need to reason about computation paths. But, the very fundamental notion of regular sequences of events is not first-order. And, it turns out that regular sequences are a natural way of describing concurrent processes [Sh79, Mi80]. Moreover, even a simple property like "the proposition p holds at least in every other state on a path" is not expressible in *PTL* [Wo81].

In view of the need to extend the expressive power of *PTL*, different approaches can be taken. The mathematician would probably choose to add some non-first-order construct, like least-fixpoint or second-order quantification. The computer scientist, on the other hand, would probably choose to add a mechanism to specify regular events as was done in [Wo81]. There, *PTL* is extended with a temporal connective corresponding to every nondeterministic finite automaton. For

† Address: Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974.
‡ Research supported by a Weizmann Post-Doctoral Fellowship and AFOSR grant 80-12907. Address: IBM Research Laboratory, 5600 Cottle Rd., San Jose, CA 95193.
* Address: Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.

0272-5428/83/0000/0185$01.00 © 1983 IEEE

example, if $\Sigma=\{a,b\}$ and A is the automaton accepting all words over Σ having a in every other position, then A is also a binary temporal connective, and the formula $A(p,true)$ is satisfied by the paths where p holds in at least every other state. Note that automata connectives can be nested within each other.

An important point that was not considered in [Wo81] is that, while the notion of regularity for finite paths is quite robust, this is not the case for infinite paths. One can think of three ways of defining acceptance of an infinite word by a nondeterministic finite automaton. We can say that the automaton accepts the word if (1) it accepts some prefix of it by the standard notion of acceptance for finite words, (2) if it has some infinite run over the word (this is the notion used in [Wo81]), or (3) it has some infinite run over the word where the set of states that repeat infinitely often in this run satisfies some additional constraint, e.g., it contains some specific state. We call these notions of acceptance *finite acceptance*, *looping acceptance*, and *repeating acceptance*, respectively. Repeating acceptance is the notion used in Büchi automata [Bu62]. The languages defined by Büchi automata are the same as those definable by ω-regular expressions [McN66]. The two other notions of acceptance are incomparable and define strictly smaller classes of languages. (For example, the sequence $(ab^*)^\omega$ can be defined by repeating acceptance but not by finite or looping acceptance.) Thus, depending on which acceptance conditions one chooses, different logics can be defined and we could expect them to have different expressive powers.

The main result of the paper is that all these logics are expressively equivalent. They all have the expressive power of ω-regular expressions, which by [Bu62] and [McN66] is the same as that of the monadic second-order theory of $(N,<)$, denoted S1S. The complexity of the decision problem for these logics ranges from polynomial space to exponential space, whereas S1S is non-elementary [Me75]. The principal technique is the maximal model technique [Pr79] viewed in a new light. While it was previously seen as a model building technique, we use it as a technique to synthesize automata that recognize models. We see this as yet another indication to the relevance of automata theory to the logic of programs (see for example [St81]).

Furthermore, our results have an interesting interpretation from a purely automata-theoretic point of view. The ability to have an automata operator nested within another automata operator is essentially equivalent to the ability of an automaton to consult an oracle. That is, when the automaton reaches certain states it asks the oracle whether it accepts the rest of the word, and its next move depends on the answer. Consider now the following hierarchy. Let A_0 be the class of nondeterministic finite automata, let A_i be the class of nondeterministic finite automata with oracles from A_{i-1}, and let A be $\bigcup_{i\geq 0} A_i$. If C is a class of automata, then $\overline{C}$ denotes the class of languages defined by automata in C.

Now we have three hierarchies A^f, A^l, and A^r, depending whether we use finite, looping, or repeating acceptance, respectively. Our results show that not only do these hierarchies collapse but that they are also equivalent: $\overline{A^f}=\overline{A^l}=\overline{A^r}=\overline{A_1^l}=\overline{A_2^l}=\overline{A_0^l}$. However, since each complementation causes at least an exponential blow-up, we might have expected the complexity of the emptiness problem for automata in A^f, A^l, and A^r to be non-elementary. Our results show that the problem is in PSPACE for A^f and A^l and is in EXPSPACE for A^r (compare this with the non-elementariness of the emptiness problem for regular expressions with complement [MS73]). From an automata-theoretic point of view, our main technique, the maximal technique, is a construction that unifies the classical *subset construction* for determinizing finite automata and the *flag construction* of [Ch74] for running several automata in parallel under central control.

Finally, to explore the full power of our technique, we introduce *alternating* finite automata connectives. These can be exponentially more succinct than nondeterministic automata connectives, and one may expect their introduction to push the complexity of the logic up. Surprisingly, this is not the case. In fact, we view alternating automata as capturing the quintessence of the maximal model technique.

Though we have chosen temporal logic as the framework for this investigation, our results can also be stated in the framework of propositional dynamic logic [FL79]. We discuss this in the concluding section of the paper.

2. Temporal Logic with Automata Connectives

We consider propositional temporal logic where the temporal operators are defined by finite automata, similarly to the extended temporal logic (*ETL*) of [Wo81]. More precisely, we consider formulas built from a set P of atomic propositions by means of:

- Boolean connectives
- Automata connectives. That is, every nondeterministic finite automaton $A=(\Sigma,S,R,s_0,F)$, where Σ is the input alphabet $\{a_1,\ldots,a_n\}$, S is the set of states, $R:\Sigma\times S\rightarrow 2^S$ is the transition relation, $s_0\in S$ is the initial state, and $F\subseteq S$ is a set of accepting states (or a set of repeating states, see below), is considered as an n-ary temporal connective. That is, if $f_1,\ldots,f_n$ are formulas, then so is $A(f_1,\ldots,f_n)$.

A structure for our logic is an infinite sequence of truth assignments, i.e., a function $\pi:N\rightarrow 2^P$ that assigns truth values to the atomic propositions in each state. We use π^i to denote the i-th "tail" of π, i.e., $\pi^i(k)=\pi(k+i)$. We now define satisfaction of formulas and runs of formulas $A(f_1,\ldots,f_n)$ over sequences by mutual induction. Satisfaction of a formula f by a structure π is denoted $\pi\models f$.

- for an atomic proposition p, $\pi\models p$ if $p\in\pi(0)$.
- $\pi\models f_1\wedge f_2$ if $\pi\models f_1$ and $\pi\models f_2$.
- $\pi\models\neg f$ if not $\pi\models f$.
- $\pi\models A(f_1,\ldots,f_n)$, where $A=(\Sigma,S,R,s_0,F)$, if there is an accepting run $s=s_0,s_1,\cdots$ of $A(f_1,\ldots,f_n)$ over π.
- A run of a formula $A(f_1,\ldots,f_n)$, where $A=(\Sigma,S,R,s_0,F)$, over a structure π is a sequence $s=s_0,s_1,\cdots$ of states from S where for all i, $0\leq i<|s|$, there is some $a_j\in\Sigma$ such that $\pi^i\models f_j$ and $s_{i+1}\in R(a_j,s_i)$.

Depending on how we define accepting runs, we get three different versions of the logic:

- ETL_f: A run s is accepting iff some state $s\in F$ occurs in s.
- ETL_l: A run s is accepting iff it is infinite.
- ETL_r: A run s is accepting iff some state $s\in F$ occurs infinitely often in s (Büchi acceptance).

Every formula defines a set of sequences, the set of sequences that satisfies it. Our yardstick for measuring the expressive power of all these logics is their ability to define sets of sequences.

Example: Consider the automaton

$$A_1=(\{s_0,s_1\},s_0,\{s_0\times a\rightarrow s_1,s_1\times b\rightarrow s_0\},\emptyset)$$

defined over the alphabet $\Sigma=\{a,b\}$. If we consider looping acceptance, it only accepts one word: $w=ababababab\cdots$. It thus defines an ETL_l connective such that $A_1(f_1,f_2)$ is true of a sequence iff f_1 is true in every even state and f_2 is true in every odd state of that sequence.

3. Translations to Automata and Decision Procedure.

Sequences of truth assignments to the propositional variables in P can be viewed as infinite words over the alphabet 2^P. It is not hard to show that our logics are translatable into S1S, and hence, by [Bu62] and [McN66], they define ω-regular sets of words. That is, there is a translation from each of the logics ETL_f, ETL_l, and ETL_r to Büchi automata. However, since negation in front of automata connectives causes an exponential blow-up, we might have expected the complexity of the translation to be non-elementary, as is the translation from S1S to Büchi automata [Bu62]. Not so.

Theorem 3.1: Given an ETL_f or an ETL_l formula f of length l, one can construct a Büchi automaton of size $exp(l)$ that accepts exactly those sequences that satisfy f.

Sketch of Proof: We will give the proof for ETL_f. The proof for ETL_l is similar. Given an ETL_f formula f, the construction of the Büchi automaton proceeds in three steps. First, we construct the *local automaton* for the formula. This automaton checks for "local inconsistencies" in the model, i.e., it checks for inconsistencies between consecutive states. For a formula $A(f_1,\ldots,f_n)$ to be satisfied, the automaton A has to reach an accepting state. This condition, which we call an *eventuality*, is not checked by the local automaton. Thus we construct a second automaton, the *eventuality automaton* whose purpose is to impose these eventuality conditions. Finally, the construction combines the local and eventuality automata.

Before giving the construction, we need to define

the notion of the closure of an ETL_f formula f, denoted $cl(f)$. It is similar in nature to the closure defined for PDL in [FL79]. Given an automaton $A=(\Sigma,S,R,s_0,F)$, for each $s\in S$ we define A_s to be the automaton (Σ,S,R,s,F). The closure is then defined as follows:

- $f\in cl(f)$
- $f_1 \wedge f_2 \in cl(f) \rightarrow f_1, f_2 \in cl(f)$
- $\neg f_1 \in cl(f) \rightarrow f_1 \in cl(f)$
- $f_1 \in cl(f)$ not of the form $\neg f_2 \rightarrow \neg f_1 \in cl(f)$
- $A(f_1,\ldots,f_n)\in cl(f) \rightarrow f_1,\ldots,f_n \in cl(f)$
- $A(f_1,\ldots,f_n)\in cl(f) \rightarrow A_s(f_1,\ldots,f_n)\in cl(f)$, for all $s\in S$.

If we define the size of an automata connective to be equal to the number of states of the automaton, then for an ETL_f formula f, the size of $cl(f)$ can easily be seen to be at most $2l$ where l is the length of f.

Constructing the Local Automaton

The local automaton is $L=(2^{cl(f)},N_L,\rho_L,N_f,N_L)$. The state set N_L will be the set of all sets X of formulas in $cl(f)$ that do not have any propositional inconsistency. Namely they must satisfy the following conditions (we identify a formula g with $\neg\neg g$):

- $g\in X$ iff $\neg g \notin X$.
- $g_1 \wedge g_2 \in X$ iff $g_1\in X$ and $g_2\in X$.

For the transition relation ρ_L, we have that $Y\in\rho_L(a,X)$ iff $a=X$ and:

- for all $A(f_1,\ldots,f_n)\in X$, where $A=(\Sigma,S,R,s,F)$, either $s\in F$ or there are an f_j and an s' such that $f_j\in X$, $s'\in R(a_j,s)$, and $A_s(f_1,\ldots,f_n)\in Y$.
- for all $\neg A(f_1,\ldots,f_n)\in X$, where $A=(\Sigma,S,R,s,F)$, $s\notin F$, and for all f_j such that $f_j\in X$, if $s'\in R(a_j,s)$ then $\neg A_s(f_1,\ldots,f_n)\in Y$.

Finally, the set of starting states N_f consists of all sets X such that $f\in X$. The local automaton does not impose any acceptance conditions. However, the infinite sequences that correspond to paths through the local automaton satisfy the formula f except possibly for eventualities. That is, if we have a formula of the form $A(f_1,\ldots,f_n)$, nothing requires that an accepting state of the automaton A will ever be reached. We now build the eventuality automaton that will ensure that the eventuality formulas are indeed satisfied.

The Eventuality Automaton

Given an ETL_f formula f, we defined the set $e(f)$ of its eventualities as the subset of $cl(f)$ that contains all formulas of the form $A(f_1,\ldots,f_n)$. The eventuality automaton is $E=(2^{cl(f)},2^{e(f)},\rho_E,\{\emptyset\},\{\emptyset\})$, where for the transition relation ρ_E, we have that $Y\in\rho_E(a,X)$ iff:

- $X=\emptyset$ and for all $A(f_1,\ldots,f_n)\in a$, where $A=(\Sigma,S,R,s,F)$, either $s\in F$ or there are an f_j and an s' such that $f_j\in a$, $s'\in R(a_j,s)$, and $A_s(f_1,\ldots,f_n)\in Y$.
- $X\neq\emptyset$ and for all $A(f_1,\ldots,f_n)\in X$, where $A=(\Sigma,S,R,s,F)$, either $s\in F$ or there are an f_j and an s' such that $f_j\in a$, $s'\in R(a_j,s)$, and $A_s(f_1,\ldots,f_n)\in Y$.

Intuitively, the eventuality automaton tries to satisfies the eventualities in the model. When the current state is $\emptyset$, it looks at the model to see which eventualities have to be satisfied. Then, the current state says which eventualities have yet to be satisfied. Observe that E does not try to satisfy the eventualities of each state of the model. Since, however, the model is infinite, it suffices to satisfy the eventualities infinitely often. Note that as opposed to what happens in PTL, the satisfaction of eventualities can not be imposed by requiring the appearance of certains states, but rather sequences of states have to be considered.

Combining the Automata

To ensure that paths through the local automaton are actual models of the ETL_f formula, we combine it with the eventuality automaton to get the *model automaton*. The model automaton $M=(2^{cl(f)},N_M,\rho_M,N_{M0},F_M)$ is obtained by taking the cross product of L and E. Its sets of states is $N_M=N_T\times 2^{e(f)}$. The transition relation ρ_M is defined as follows: $(Y,Z)\in\rho_M(a,(W,X))$ iff $Y\in\rho_L(a,W)$ and $Z\in\rho_E(a,X)$. The set of starting states is $N_{M0}=N_f\times\{\emptyset\}$, and the set of repeating states is $F_M=N_L\times\{\emptyset\}$.

The automaton M is a Büchi automaton. That is, it is an automaton on infinite strings and the strings accepted are those for which the intersection between the states visited infinitely often and F_M is nonempty.

The automaton we have constructed, accepts strings over $2^{cl(f)}$. However, the models of f are defined

by strings over 2^P. So, the last step of our construction is to take the projection of our automaton on 2^P. This is done by mapping each element $b \in 2^{cl(f)}$ into all elements $a \in 2^P$ such that $b \cap P \subseteq a$.

The size of both the local and eventuality automata is at most $2^{cl(f)}$. Thus for a formula of length l, the size of the model automaton constructed is at most 2^{4l}. ■

The construction described above simultaneously takes care of running automata in parallel, when we have an automata connective nested within another automata connective, and complementing automata, when we have a complemented automata connective. Thus the construction can be viewed as combining the classical subset construction and Choueka's "flag construction" in [Ch74]. It is interesting the compare this technique to Pratt's model construction technique [Pr79]. There one starts by building a maximal model, and then one eliminates states whose eventualities are not satisfied. Our local automata correspond to those maximal models. However, instead of eliminating states, we combine the local automata with eventuality automata that check for satisfaction of eventualities. This construction always yields automata whose size is exponential in the size of the formula. We could also construct our automata using the *tableau* technique of [Pr80]. This technique can sometimes be more efficient than the maximal model technique.

Theorem 3.2: Given an ETL_r formula f of length l, one can construct a Büchi automaton of size $\exp^3(l)$ that accepts exactly those sequences that satisfy f.

Idea of Proof: The maximal technique is not applicable here, because the subset construction is not applicable to Büchi automata. Instead, we use McNaughton's construction [McN66] for complementing Büchi automata and the flag construction [Ch74] to combine all the automata that occur in f. ■.

The constructions of Theorems 3.1-3.2 also give us decision procedures for the various logics, since to check whether a formula f is satisfiable it suffices to check that the synthesized automaton accepts some word (the so called *emptiness problem*). This will cost exponential time for ETL_f and ETL_l and triply exponential time for ETL_r. We can, however, do better by realizing that it is not necessary to synthesize the Büchi automaton given by the theorems. Rather, it suffices to non-deterministically guess a path through the automaton and check whether this path leads to acceptance. This is quite simple for ETL_f and ETL_l, but doing it for ETL_r requires a thorough understanding of McNaughton's construction. Combining it with the fact that *PTL* is PSPACE-hard [SC82], we get:

Theorem 3.3: The satisfiability problem for ETL_f and ETL_l is logspace complete in PSPACE†, and the satisfiability problem for ETL_r is in EXPSPACE. ■

The applicability of the maximal model method to ETL_f and ETL_l also enables us to get a sound and complete axiomatization for these logics.

4. Translations Among the logics

The results of the previous section show that ETL_r has the same expressive power as Büchi automata. Since the notions of finite and looping acceptance are weaker than the notions of repeating acceptance, it would have been conceivable for ETL_f and ETL_l to be less expressive than ETL_r. We show, however, that they have the same expressive power.

Theorem 4.1: Given an ETL_r formula f of length l, one can construct an ETL_f formula f' and an ETL_l formula f'', both of length $O(exp^3(l))$, that are satisfied by exactly the same sequences as f.

Proof: We give the proof for ETL_f, the proof for ETL_l is similar. The proof will show how an ETL_r formula $A(f_1,...,f_n)$ can be translated into ETL_f. The main difficulty is to express in ETL_f the condition imposed by the repetition set of the nondeterministic Büchi automaton. This can be done by using a deterministic version of the formula.

Determinizing the Formula

We want to express $A(f_1, \ldots, f_n)$ in terms of automata connectives that are deterministic in the sense that, given a structure, there will be at most one run of the formula over that structure. The nondeterminism we have to deal with is double. First, one can have several transitions labeled by the same letter coming from the same state. Secondly, a given state in a given structure may satisfy more than one f_i.

† Satisfiability of ETL_l was originally shown to be PSPACE-complete by a different technique in [Wo82] and, though with a flaw in the proof, in [SC82].

To overcome the second type of nondeterminism, we replace $A=(\Sigma,S,R,s,F)$ by an automaton over 2^{Σ}. the automaton is $A'=(2^{\Sigma},S,R',s,F)$, where the transition relation R' is defined as follows: $s_i \in R'(X,s_j)$ iff $s_i \in R(a,s_j)$ for some $a \in X$. Now, $A(f_1,\ldots,f_n)$ is equivalent to $A'(g_1,\ldots,g_{2^n})$, where the formula g_i corresponding to a set $X_i \subseteq \Sigma$ is $(\bigwedge_{a_j \in X_i} f_j) \wedge (\bigwedge_{a_j \notin X_i} \neg f_j)$. Clearly, in a given state exactly one g_i is satisfied.

We now have to deal with the nondeterminism of the first kind. Unlike finite automata over finite words, nondeterministic Büchi automata are more powerful than deterministic Büchi automata. In order to be able to determinize our automata, we need the more general notion of acceptance of infinite words due to Muller [Mu63]. This acceptance condition is specified by a collection F of subsets of S. An automaton whose acceptance condition is specified by F accepts an infinite word w if it has a run over w where the set of states that repeat infinitely often belongs to F. It is shown in [McN66] that every n-state Büchi automaton is equivalent to a $O(exp^2(n))$-state deterministic Muller automaton. We can extend the definition of ETL_r in the obvious way to include Muller automata connectives. Furthermore, if $F=\{F_1,\ldots,F_n\}$ and A_i is the Muller automaton $(\Sigma,S,R,s,\{F_i\})$, then

$$A(g_1,\ldots,g_m)=\bigvee_{i=1}^{n} A_i(g_1,\ldots,g_m)$$

So, we only need to show how a formula $A(g_1,\ldots,g_m)$, where A is a Muller automaton $(\Sigma,S,R,s,\{F\})$ can be translated into ETL_f. The automaton $A(g_1,\ldots,g_m)$ has exactly one infinite run over any given structure.

Expressing Repetition

Let s' be any state in F. The formula $A(g_1,\ldots,g_m)$ is satisfied by a given structure, if the run over the structure reaches the state s', and from that point, $A(g_1,\ldots,g_m)$ has an infinite run starting with s' and going only through states from F. This last property is expressed by the ETL_f formula $\varphi=\neg A'(g_1,\ldots,g_m)$, where $A'=(\Sigma,S,R,s',S-F)$. Thus $A(g_1,\ldots,g_m)$ is satisfied if the run reaches s' and at that point φ is satisfied. To express this we add to S a new accepting state e, we add to Σ a new letter a, and we extend the transition relation by $R(a,s')=e$. Let the extended automaton be $B=(\Sigma \cup \{a\},S \cup \{e\},R \cup (a,s',e),s,\{e\})$. Now, the ETL_f formula $B(g_1,\ldots,g_m,\varphi)$ is satisfied by a structure iff the formula $A(g_1,\ldots,g_m)$ is satisfied by the structure.

Complexity of the translation

The deterministic automaton is of size two exponentials in the size of the Büchi automaton. But, the set F of sets of repeating states in the deterministic version can contain a number of elements exponential in the size of that automaton. The whole translation is thus three exponentials. ■

Theorems 3.1 and 4.1 together provide a translation between ETL_f and ETL_l. That translation, however, is quadruply exponential. We can do much better.

Theorem 4.2: There is a one exponential translation between ETL_f and ETL_l.

Idea of Proof: For an automaton A, let A' be the automaton constructed from A by the subset construction. The key idea is that looping rejection of an infinite word by A means that A' reaches $\emptyset$ on this word, and finite rejection means that A' loops on sets that do not contain accepting states. Thus, looping and finite acceptance are in some sense dual notions. The translation between the logics proceeds by applying the subset construction to the automata connectives and then complementing them. ■

It is also interesting to note that our logics are expressively equivalent to a quantified version of temporal logic. However quantified temporal logic is, like S1S, of non-elementary complexity (the number of exponentials depends on the alternation of quantifiers).

5. Alternating Temporal Logic

The results of the preceding sections show that the maximal model technique is applicable to automata connectives and to the negation of automata connectives. This suggests that this technique can also deal with alternation.

Given a set S of states, let us denote by B_S the set of all Boolean formulas that use the states in S as variables. Members of B_S can be viewed as Boolean-valued functions on 2^S. Let $\varphi \in B_S$ and $S' \subseteq S$. Then $\varphi(S')$ is

the Boolean value of φ when the states in S' are assigned 1 and the states in $S-S'$ are assigned 0. An *alternating finite automata* [BL80,CKS81] (abbr. afa) A is a quintuple $A=(\Sigma,S,g,\varphi_0,F)$, where Σ is the input alphabet, S is the set of states $\{s_1,\ldots,s_m\}$, $g:\Sigma\times S\rightarrow B_S$ is the transition function that associates with each state and letter a Boolean formula in B_S, $\varphi_0\in B_S$ is the start formula, and $F\subseteq S$ is the set of accepting states. We can extend g to $\Sigma\times B_S$: $g(a,\varphi)$ is obtained by substituting $g(a,s_j)$ in φ for each s_j, $1\leq j\leq m$. The run of A on a word $w=a_1,\ldots,a_l$ is the sequence $\varphi_0,\ldots,\varphi_l$ of formulas from B_S, where $\varphi_i=g(a_i,\varphi_{i-1})$. A accepts w if $\varphi_l(F)=1$.

Afa's define regular sets. Nevertheless, it follows from the results in [Le81,CKS81] that they can be exponentially more succinct than nfa's. That is, given any n-state afa, one can construct an 2^n-state nfa that accepts the same language. Furthermore, for each n there is an n-states afa A, such that the language defined by A is not definable by any nfa with less than 2^n states.

For simplicity we deal here only with the alternating analog of ETL_f. ATL_f is defined analogously to ETL_f, with afa connectives replacing nfa connectives. We require, however, that the start formulas of the afa connectives be either states or negation of states. For an afa $A=(\Sigma,S,g,\varphi_0,F)$ and a formula $\varphi\in B_S$, we define A_φ to be the afa (Σ,S,g,φ,F). The semantics of ATL_f are defined as follows:

- for an atomic proposition p, $\pi\models p$ iff $p\in\pi(0)$.
- $\pi\models f_1\wedge f_2$ iff $\pi\models f_1$ and $\pi\models f_2$.
- $\pi\models\neg f$ iff not $\pi\models f$.

For an automata connective $A=(\Sigma,S,g,\varphi_0,F)$ we have:

- $\pi\models A(f_1,\ldots,f_n)$ if and only if $\varphi_0(F)=1$ or there are an f_j and a set $S'\subseteq S$ such that $i\models_\pi f_j$, $g(a,\varphi_0)(S')=1$, for all $s\in S'$ we have $\pi^1\models A_s(f_1,\ldots,f_n)$, and for all $s\in S-S'$ we have $\pi^1\models A_{\neg s}(f_1,\ldots,f_n)$.

In §2 satisfaction is defined via some accepting run of the formulas over the structure. This kind of definition here would not correspond to our intuitive notion of automata running in parallel over the structure π, while the above definition does. Clearly, ATL_f is at least as expressive as ETL_f. Indeed, if we restrict the formulas in B_S to be positive disjunctions then ATL_f reduces to ETL_f. Also, there is an exponential translation from ATL_f to ETL_f. (Note that the translation must involve more than translating afa to nfa, since the semantics of ATL_f is not analogous to that of ETL_f.) The interest in this logic is twofold. First, since the translation from ATL_f to ETL_f causes a one exponential blow-up, one may expect ATL_f to be of higher complexity then ETL_f. Surprisingly, it has the same complexity. Furthermore, ATL_f is the natural "homeland" of the maximal technique, and it is theoretically interesting to pursue the technique to its limit.

Theorem 5.1:

1. Given an ATL_f formula f of length l, one can construct a Büchi automaton of size $exp(l)$ that accepts exactly those sequences that satisfy f.
2. The satisfiability problem for ATL_f is complete in PSPACE.

Sketch of Proof: We prove here the first claim, and the second claim follows as in §2. The proof parallels the proof of Theorem 3.1. We are given a formula f, and we construct a model automaton, which is the cross product of the local automaton and the eventuality automaton. The notion of closure is similar to that in Theorem 3.1. For an ATL_f formula f, $cl(f)$ is defined as follows:

- $f\in cl(f)$
- $f_1\wedge f_2\in cl(f)\rightarrow f_1,f_2\in cl(f)$
- $\neg f_1\in cl(f)\rightarrow f_1\in cl(f)$
- $f_1\in cl(f)$ not of the form $\neg f_2\rightarrow\neg f_1\in cl(f)$
- $A(f_1,\ldots,f_n)\in cl(f)\rightarrow f_1,\ldots,f_n\in cl(f)$
- $A(f_1,\ldots,f_n)\in cl(f)\rightarrow A_s(f_1,\ldots,f_n)\in cl(f)$, for all $s\in S$.
- $A(f_1,\ldots,f_n)\in cl(f)\rightarrow A_{\neg s}(f_1,\ldots,f_n)\in cl(f)$, for all $s\in S$.

Constructing the Local Automaton

The local automaton is $L=(2^{cl(f)},N_L,\rho_L,N_f,N_L)$. The state set N_L will be the set of all sets X of formulas in $cl(f)$ that do not have any propositional inconsistency as in Theorem 3.1. For the transition relation ρ_L, we have that $Y\in\rho_L(a,X)$ iff $a=X$ and:

- For all $A(f_1,\ldots,f_n)\in X$, where

$A=(\Sigma,S,g,\varphi_0,F)$, either $\varphi_0(F)=1$ or there are an f_j and a set of states $S'\subseteq S$ such that $f_j\in X$, $g(a_j,\varphi_0)(S')=1$, for all $s\in S'$ we have $A_s(f_1,\ldots,f_n)\in Y$, and for all $s\in S-S'$ we have $A_{\neg s}(f_1,\ldots,f_n)\in Y$.

- For all $\neg A(f_1,\ldots,f_n)\in X$, where $A=(\Sigma,S,g,\varphi_0,F)$, we have $\varphi_0(F)=0$ and for all f_j such that $f_j\in X$ and all sets $S'\subseteq S$ such that $g(a_j,\varphi_0)(S')=1$ either for some $s\in S'$ we have $\neg A_{\neg s}(f_1,\ldots,f_n)\in Y$ or for some $s\in S-S'$ we have $\neg A_s(f_1,\ldots,f_n)\in Y$.

Finally, the set of starting states N_f consists of all sets X such that $f\in X$.

The Eventuality Automaton

Given an ETL_f formula f, we defined the set $e(f)$ of its eventualities as the subset of $cl(f)$ that contains all formulas of the form $A(f_1,\ldots,f_n)$. The eventuality automaton is $E=(2^{cl(f)},2^{e(f)},\rho_E,\{\emptyset\},\{\emptyset\})$, where for the transition relation ρ_E, we have that $Y\in\rho_E(a,X)$ iff:

- $X=\emptyset$ and for all $A(f_1,\ldots,f_n)\in a$, where $A=(\Sigma,S,g,\varphi_0,F)$, either $\varphi_0(F)=1$ or there are an f_j and a set of states $S'\subseteq S$ such that $f_j\in X$, $g(a_j,\varphi_0)(S')=1$, for all $s\in S'$ we have $A_s(f_1,\ldots,f_n)\in Y$, and for all $s\in S-S'$ we have $A_{\neg s}(f_1,\ldots,f_n)\in Y$
- $X\neq\emptyset$ and for all $A(f_1,\ldots,f_n)\in X$, where $A=(\Sigma,S,g,\varphi_0,F)$, either $\varphi_0(F)=1$ or there are an f_j and a set of states $S'\subseteq S$ such that $f_j\in X$, $g(a_j,\varphi_0)(S')=1$, for all $s\in S'$ we have $A_s(f_1,\ldots,f_n)\in Y$, for all $s\in S-S'$ we have $A_{\neg s}(f_1,\ldots,f_n)\in Y$

Combining the Automata

The model automaton is taken to be the cross product of the local automaton and the eventuality automaton. The model automaton is of size at most 2^{4l}. Finally, we project the model automaton on 2^P. ■

6. Concluding Remarks

Another approach to extend *PTL* was taken in [HP82]. They chose to extend the language by regular operators corresponding to concatenation and the Kleene star. This, however, pushes the decision problem for their language to non-elementary complexity. Furthermore, it does not show the fine interplay between the different acceptance conditions that we have considered. These conditions can also be presented within the framework of propositional dynamic logic (*PDL*) [FL79]. Since we are reasoning here about computation paths, we consider *PDL* with one deterministic atomic program (1*DPDL*), where the structures are infinite sequences. In this framework, the finite acceptance condition corresponds to the *diamond* construct [FL79], the looping acceptance condition corresponds to the *loop* construct [HP79], and the repeating acceptance condition corresponds the the *repeat* construct [HP79]. It follows that adding to 1*DPDL* looping, repeating, fixpoint, and even quantification over propositions does not change the expressive power of the language and does not render it undecidable. This in sharp contrast with what happens for propositional dynamic logic in general. It is known that *PDL* is less expressive from *PDL* + *loop* (Pratt), which is less expressive than *PDL* + *repeat* [HS83], which is less expressive than *PDL* + *fixpoint* [Ko82], which can be shown to be less expressive than *PDL* + *quantification*. Also, the last language can even be shown to be highly undecidable (Π^1_1-complete).

The maximal model technique that we are using is also applicable to propositional dynamic logic, extending the results in [Pr81] for *flowchart* − *PDL*. Consider the logic $LPDL_{alt}$, which is *PDL* augmented by the *loop* construct and with programs described by alternating finite automata. We can decide satisfiability for this logic by translating it into *PDL* + *repeat* and using the decision procedure of [St81], but that would take quintuply exponential time. By using the maximal model technique, we can not only give an exponential decision procedure, but also give a sound and complete axiomatization.

Another consequence concerns the relative expressive power of S1S, and WS1S (WS1S is S1S where quantification is only over finite sets). From the fact that we can simulate automata with repeating acceptance by automata with finite acceptance, it follows that quantification over arbitrary sets can be simulated by quantification over finite sets. That means that S1S and WS1S have exactly the same expressive power. This observation generalizes results appearing in [Bu62].

Acknowledgements. We'd like to thank R. Fagin, J. Halpern, and L. Stockmeyer for helpful comments.

References

[BL80] J.A. Brzozowski, E. Leiss, "On Equations for Regular Languages, Finite Automata, and Sequential Networks. *Theoretical Computer Science* 10(1980), pp. 19-35.

[Bu62] J. R. Büchi, "On a Decision Method in Restricted Second Order Arithmetic", *Proc. Internat. Congr. Logic, Method and Philos. Sci. 1960,* Stanford University Press, 1962, pp. 1-12.

[Ch74] Y. Choueka, "Theories of Automata on ω-Tapes: A Simplified Approach", *J. Computer and System Sciences,* 8 (1974), pp. 117-141.

[CKS81] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, "Alternation", *J. ACM* 28 (1981), pp. 114-133.

[FL79] M. Fisher, R. Ladner, "Propositional Dynamic Logic of Regular Programs", *J. of Computer and System Sciences,* 18(2), 1979, pp. 194-211.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, "The Temporal Analysis of Fairness", *Proc. 7th ACM Symp. on Principles of Programming Languages,* Las Vegas, 1980, pp. 163-173.

[HP79] D. Harel, V. Pratt, "Nondeterminism in logics of programs", *Proc. 5th ACM Symp. on Principles of Programming Languages,* Tuscon, 1978, pp. 203-213.

[HKP80] D. Harel, D. Kozen, R. Parikh, "Process Logic: Expressiveness, Decidability, Completeness", *J. Computer and System Science* 25(2), 1982, pp. 144-170

[HP82] D. Harel, D. Peleg, "Process logic with Regular Formulas", Technical Report, Bar-Ilan University, Ramat-Gan, Israel, 1982.

[HS83] D. Harel, R. Sherman, "Looping vs. Repeating in Dynamic Logic", Technical Report CS83-02, Weizmann Institute, Rehovot, Israel, 1983.

[Ko82] D. Kozen, "Results on the Propositional μ-Calculus", *Proc. 9th Int. Colloq. on Automata, Languages and Programming,* Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1982, pp. 348-359.

[Le81] E. Leiss, "Succint Representation of Regular Languages by Boolean Automata", *Theoretical Computer Science* 13(1981), pp. 323-330.

[McN66] R. McNaughton, "Testing and Generating Infinite Sequences by a Finite Automata", *Information and Control* 9 (1966), pp. 521-530

[Me75] A. R. Meyer, "Weak Monadic Second Order Theory of Successor is not Elementary Recursive", *Proc. Logic Colloquium,* Lecture Notes in Mathematics, v. 453, Springer-Verlag, 1975, pp. 132-154.

[Mi80] R. Milner, "A Calculus of Communicating Systems", Lecture Notes in Computer Science, v. 92, Springer-Verlag, 1980.

[MS73] A. R. Meyer, L. J. Stockmeyer, "Nonelementary Word Problems in Automata and Logic", *Proc. AMS Symp. on Complexity of Computation,* April 1973.

[Mu63] D.E. Muller, "Infinite sequences and finite machines", *Proc. 4th IEEE Ann. Symp. on Switching Circuit Theory and Logical Design,* New York, 1963, pp. 3-16.

[Ni80] H. Nishimura, "Descriptively Complete Process Logic", *Acta Informatica,* 14 (1980), pp. 359-369.

[Pn77] A. Pnueli, "The Temporal Logic of Programs", *Proc. 8th IEEE Symp. on Foundations of Computer Science,* Providence, 1977, pp. 46-57.

[Pr79] V.R. Pratt, "Models of program logics", *Proc. 20th IEEE Symp. on Foundation of Computer Science,* San Juan, 1979, pp. 115-122.

[Pr80] V.R. Pratt, "A Near-Optimal Method for Reasoning about Action", *J. Comp. and Sys. Sciences* 20(1980), pp. 231-254.

[Pr81] V.R. Pratt, "Using Graphs to understand PDL", *Proc. Workshop on Logics of Programs,* (D. Kozen, ed.), Yorktown-Heights, Lecture Notes in Computer Science, vol. 131, Springer-Verlag, Berlin, 1982, pp. 387-396.

[SC82] A. P. Sistla, E. M. Clarke, "The Complexity of Propositional Linear Temporal Logic", *Proc. 14th ACM Symp. on Theory of Computing,* San Francisco, 1982, pp.159-168.

[Sh79] A.C. Shaw, "Software Specification Languages Based on Regular Expressions", Technical Report, ETH Zurich, June 1979.

[St81] R. Streett, "Propositional Dynamic Logic of Looping and Converse", *Proc. 13th ACM Symp. on Theory of Computing*, Milwaukee, 1981, pp. 375-383.

[Wo81] P. Wolper, "Temporal Logic Can Be More Expressive", *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, Nashville, 1981, pp. 340-348.

[Wo82] P. Wolper, "Synthesis of Communicating Processes from Temporal Logic Specifications", Ph. D. Thesis, Stanford University, 1982.