LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - The University of Edinburgh

Fixpoints of Büchi automata

by

Mads Dam

LFCS Report Series ECS-LFCS-92-224

LFCS July 1992

Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Fixpoints of Büchi automata

Mads Dam*
Dept. of Computer Science, University of Edinburgh, U.K.

Abstract

We give a new and direct proof of the equivalence between the linear time μ-calculus νTL and Büchi automata. In contrast to previously known constructions ours is both elementary and compositional: Constructions on automata are provided which compute their least and greatest fixed points. Applications include the translation of fragments of the full branching μ-calculus into the modal μ-calculus, and the problem of providing reasonable sound and complete axiomatisations for νTL.

1 Introduction

The relation between automata as devices for recognising behaviours, and fixpoints, or equations, as means of characterising them is an important recurring theme in the theory of computation. The ω-regular languages provides an example of particular interest in concurrency theory. They are characterised on the one hand by formulas in the linear time μ-calculus νTL, linear time logic augmented by least and greatest fixed points of formally monotone contexts. They are also exactly the languages recognised by Büchi automata, finite automata applied to words of infinite length. Both νTL and Büchi automata have had considerable attention as formalisms for specifying and verifying concurrent programs (c.f. [1, 2, 8, 11, 13, 21]).

We suggest examining the connection between νTL and Büchi automata further. Büchi automata at present lacks a structural theory which is usable in practice, for instance for machine implementation or to support equational reasoning. The equivalence with S1S, the monadic second-order theory of successor, is nonelementary [12] and thus offers little concrete assistance. The linear time μ-calculus is potentially much more valuable for this purpose. Fixpoints, on the other hand, can be very troublesome in practical use: Experience, for instance with the Edinburgh Concurrency Workbench [4], has shown that already at the

*Supported by SERC grant GR/F 32219

second level of alternation formulas can become highly unintelligible. To remedy this, automata can prove useful tools for visualising properties written in νTL.

The value of a compositional, or syntax-directed approach in such an enterprise is well documented. It gives a direct account in terms of automata of each νTL connective. For all νTL-connectives except the fixpoint quantifiers there are corresponding standard constructions on automata (c.f. [14, 19]). In this paper we provide procedures for the fixpoint quantifiers. That is, given an automaton recognising the language expressed by the νTL-formula ϕ where ϕ is formally monotone in the variable X, we produce automata recognising the least and greatest fixpoints, $\mu X.\phi$ and $\nu X.\phi$ respectively, of the operator $\lambda X.\phi$. The result is a compositional procedure for deriving from each νTL-formula an equivalent Büchi automaton.

This procedure may be of value not only in the linear time case. The branching time μ-calculus [6] extends νTL by the universal and existential path quantifiers A and E. An important fragment of this logic, here called μCTL*, allows only the formation of linear fixed points. That is, a fixed point formula $\sigma X.\phi$ is allowed only when no occurrence of X in ϕ is within the scope of a path quantifier in ϕ. This logic is a natural generalisation of CTL* and is powerful enough to describe most temporal properties of interest in practical applications, moreover in a fairly transparent way. Our procedure can be used as a compositional translation of μCTL* into ECTL*, an extended version of CTL* where the linear time fragment is replaced by Büchi automata [18]. By composing with the relatively straightforward translation of ECTL* into the modal μ-calculus of Dam [5] a compositional translation of μCTL* into the latter logic is obtained, so that for instance the model checker of Stirling and Walker [16], implemented in the Edinburgh Concurrency Workbench, become available.

Our approach should be compared with the more established automata-theoretic techniques, e.g. [17, 20]. Their approach is global rather than compositional: The automaton for a formula ϕ is built as the intersection of an automaton that checks local model conditions with the complement of an automaton that checks for non-well-foundedness of a certain regeneration relation.

Of course only one fixpoint construction, for instance for greatest fixpoints, is needed due to the equivalence $\mu X.\phi \equiv \neg\nu X.\neg\phi[\neg X/X]$. It turns out, however, that the construction for least fixed points generalises the construction for greatest fixed points in a natural way, and by using it we can rewrite νTL-formulas to Büchi automata without ever having to complement automata explicitly. Indeed, as Emerson and Jutla [7] for tree automata, our result gives the Complementation Theorem as a trivial corollary.

The paper is organised as follows: In section 2 we introduce νTL, and in section 3 we introduce Büchi automata and show how they can be represented in νTL. The fixpoint construction first builds an intermediate automaton with nonstandard, syntactically determined acceptance conditions. This construction is described in section 4. The construction for greatest fixed points is given in

section 5, and for least fixed points in section 6. In section 7 we discuss a number of possible optimisations and give two simple examples. Finally, in section 8, we apply our construction to the problem of axiomatising νTL. A natural candidate axiomatisation is obtained by adapting Kozen's axiomatisation of the modal μ-calculus [9]. Using our construction we can view Büchi automata as normal forms for νTL, and we show that a sound and complete axiomatisation is obtained by adding an axiom schema equating each formula with its normal form. Such a feature can be highly useful in the context of machine-based implementations. It is nonetheless of interest to investigate to what extent this schema is really needed. By proving cases of our construction correct without reference to the normal form schema, the latter can be narrowed down considerably and indeed for the aconjunctive fragment [9] it can be eliminated altogether. Moreover for νTL the condition of aconjunctivity does not restrict expressive power. Our approach is related to Siefke's completeness result for S1S [15] and Kozen's recent completeness result for the algebra of regular events [10]. Key ingredients in their proofs are, as here, algebraic correlates of constructions on automata.

2 The linear time μ-calculus

Formulas ϕ, ψ, γ of the linear-time μ-calculus νTL are built from propositional variables X, Y, Z, boolean connectives $\neg$ and $\wedge$, the nexttime operator O, and the least fixpoint operator $\mu X.\phi$, subject to the formal monotonicity condition that all free occurrences of X lie in the scope of an even number of negations. Other connectives are derived in the usual way, and in particular greatest fixpoints are derived by $\nu X.\phi \triangleq \neg\mu X.\neg\phi[\neg X/X]$. Intuitively, least fixed points are used for eventuality properties, and greatest fixed points for invariants.

Fix a finite set Σ of propositional variables. A *model* $\mathcal{M}$ assigns to each variable $X \in \Sigma$ a subset $\mathcal{M}(X) \subseteq \omega$. Models are extended to arbitrary formulas with free variables in Σ in the following way:

$$\begin{aligned}
\mathcal{M}(\neg\phi) &= \overline{\mathcal{M}(\phi)} \\
\mathcal{M}(\phi \wedge \psi) &= \mathcal{M}(\phi) \cap \mathcal{M}(\psi) \\
\mathcal{M}(O\phi) &= \{i+1 \mid i \in \mathcal{M}(\phi)\} \\
\mathcal{M}(\mu X.\phi) &= \cap\{A \subseteq \omega \mid \mathcal{M}[X \mapsto A](\phi) \subseteq A\}
\end{aligned}$$

Here $\mathcal{M}[X \mapsto A]$ is the obvious update of $\mathcal{M}$. There is a bijective correspondence between models and ω-words α over the alphabet 2^Σ. The model $\mathcal{M}$ determines the ω-word $\alpha_\mathcal{M} : i \mapsto \{X \mid i \in \mathcal{M}(X)\}$, and the *language defined by* ϕ is

$$L(\phi) = \{\alpha_\mathcal{M} \mid 0 \in \mathcal{M}(\phi)\}. \tag{1}$$

Operations on ω-words α include the *n'th suffix*, α^n, and, where $n \leq m$, the *n, m-segment*, $\alpha(n,m) = \alpha(n)\cdots\alpha(m)$.

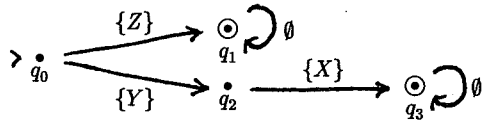

Figure 1: Büchi automaton $\mathcal{A}_1$ for $Z \vee (Y \wedge OX)$

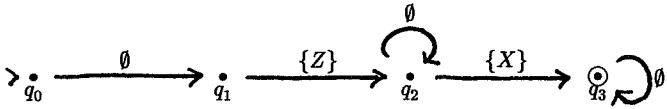

Figure 2: Büchi automaton $\mathcal{A}_2$ for $O((O(\mu Y.X \vee OY)) \wedge Z)$

3 Büchi automata

Automata provide an alternative way of defining ω-languages. We use a slightly modified account of Büchi automata, closely related to Alpern and Schneider's use of transition predicates [1]. Fix a finite set Σ of propositional variables. An *atom* over Σ is a pair $a = (a^+, a^-)$ where a^+ and a^- are subsets of Σ. Intuitively, a transition labelled a is enabled when all members of a^+ are true and all members of a^- false. The set of all atoms over Σ is denoted by $At(\Sigma)$.

A Büchi-automaton (over Σ) is an NFA $\mathcal{A} = (Q, q_0, \{\xrightarrow{a}\}_{a \in At(\Sigma)}, F)$ where Q is the finite set of states, $q_0 \in Q$ is the initial state, $\xrightarrow{a} \subseteq Q \times Q$ is the transition relation for each $a \in At(\Sigma)$, and $F \subseteq Q$ is the set of accepting states. We sometimes write $\mathcal{A}(q_0)$ instead of just $\mathcal{A}$ to emphasize the initial state. An (infinite) *run* of $\mathcal{A}$ on the word α is an ω-word Π over Q s.t. $\Pi(0) = q_0$ and for all $i \geq 0$ there is an atom $a \in At(\Sigma)$ s.t. $\Pi(i) \xrightarrow{a} \Pi(i+1)$, $a^+ \subseteq \alpha(i)$ and $a^- \cap \alpha(i) = \emptyset$. Finite runs are defined similarly. An infinite run is *successful* if some accepting state in F occurs infinitely often in it, and $\mathcal{A}$ *accepts* α if a successful run of $\mathcal{A}$ on α exists. The language *recognised by* $\mathcal{A}$ is $L(\mathcal{A}) = \{\alpha \mid \mathcal{A} \text{ accepts } \alpha\}$.

Example 3.1 In all examples here and below formulas are positive in their free propositional variables. The negative component of atoms can consequently be omitted.

(1) The automaton $\mathcal{A}_1$ of fig. 1 recognises the language defined by the νTL formula $Z \vee (Y \wedge OX)$.

(2) The automaton $\mathcal{A}_2$ of fig. 2 recognises $O((O(\mu Y.X \vee OY)) \wedge Z)$, equivalent to the PTL formula $O((OFX) \wedge Z)$. □

(3) The automaton $\mathcal{A}_3$ of fig. 3 recognises $\nu X.Y \wedge OOX$ which is the set of all models for which Y holds at all even moments. This property is not definable in PTL [23]. □

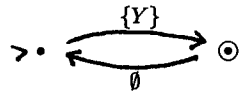

Figure 3: Büchi automaton $\mathcal{A}_3$ for $\nu X.Y \wedge OOX$

The Büchi automaton $\mathcal{A}$ is represented as a νTL formula $fm(\mathcal{A})$ in the following way (c.f. [13]). Let $F_{\mathcal{A}} = \{q_1, \ldots, q_n\}$ and for each $1 \leq i \leq n$, let $\mathcal{A}_i$ be $\mathcal{A}$ with F replaced by the singleton $\{q_i\}$. Then $L(\mathcal{A}) = \bigcup_{1 \leq i}^{n} L(\mathcal{A}_i)$ so we can let $fm(\mathcal{A}) \triangleq \bigvee_{1 \leq i}^{n} fm(\mathcal{A}_i)$. To represent the $\mathcal{A}_i$, states are represented as fixpoint formulas, the unique accepting state as a ν-formula and all other states as μ-formulas. For each state q we let X_q be a distinguished propositional variable. Atoms are dealt with by defining

$$a.\phi \triangleq O\phi \wedge \bigwedge a^{+} \wedge \bigwedge\{\neg X \mid X \in a^{-}\} \tag{2}$$

Then the representation, $fm(q)\rho$, of q relative to the environment $\rho \subseteq Q$ keeping track of earlier encountered states is defined in the following way:

$$fm(q)\rho = \begin{cases} X_q & if\ q \in \rho \\ \mu X_q.\bigvee\{a.fm(q')\rho \cup \{q\} \mid q \xrightarrow{a} q'\} & if\ q \notin \rho\ and\ q \neq q_i \\ \nu X_q.\bigvee\{a.fm(q')\{q\} \mid q \xrightarrow{a} q'\} & otherwise \end{cases} \tag{3}$$

When the ambiguity is resolved from context we abbreviate $fm(q_0)\emptyset$, the representation of $\mathcal{A}_i$, as $fm(q_0)$. It hardly needs saying that no variables need be introduced in (3) that are never actually used. Note that we can assume that every state q has a successor, i.e. that there are a and q' such that $q \xrightarrow{a} q'$ so that only nonempty disjunctions in (3) are needed. This assumption applies throughout the rest of the paper.

Example 3.2 The automaton $\mathcal{A}_3$ of fig. 3 is represented as $Y \wedge O\nu X.O(Y \wedge OX)$. Without resetting environments at the accepting state as in (3) the result would be $\mu X.Y \wedge OOX$, equivalent to $false$. □

The representation is closely related to the translation of ECTL* into the modal μ-calculus of Dam [5] and can be proved correct in the same way.

Theorem 3.3 *For each Büchi automaton $\mathcal{A}$, $L(\mathcal{A}) = L(fm(\mathcal{A}))$.* □

4 Intermediate automata

To derive equivalent Büchi automata from νTL-formulas we give for each connective of νTL a corresponding construction on automata. Each formula can be put in *positive form*, generated by

$$\phi ::= X \mid \neg X \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid O\phi \mid \nu X.\phi \mid \mu X.\phi$$

so we only need consider negation applied to propositional variables. It is easy to produce automata $aut(X)$ and $aut(\neg X)$ respectively recognising $L(X)$ and $L(\neg X)$, and to produce an automaton $O\mathcal{A}$ recognising $L(O\phi)$ when $\mathcal{A}$ recognises $L(\phi)$. Corresponding to the $\vee$ is the sum operation $\mathcal{A}_1 + \mathcal{A}_2$ which adjoins a new initial state to the disjoint sum of the statesets of $\mathcal{A}_1$ and $\mathcal{A}_2$. Corresponding to the $\wedge$ is a product automaton $\mathcal{A}_1 \times \mathcal{A}_2$ which accepts when first an accepting state of $\mathcal{A}_1$ and then of $\mathcal{A}_2$ is encountered (c.f. [19]).

Completing this procedure it thus remains to produce automata $\nu X.\mathcal{A}$ and $\mu X.\mathcal{A}$ for $\nu X.\phi$ and $\mu X.\phi$ respectively when $\mathcal{A} = (Q, q_0, \{\stackrel{a}{\rightarrow}\}_{a \in At(\Sigma)}, F)$ recognises $L(\phi)$. We assume here that ϕ is formally monotone in X. In this case our procedure takes care to ensure that whenever $q \stackrel{a}{\rightarrow} q'$ in $\mathcal{A}$ then $X \notin a^-$. We can moreover assume that ϕ does not contain any unguarded occurrences of variables, using for instance the technique of Banieqbal and Barringer [2]. The construction will then ensure that whenever $q_0 \stackrel{a}{\rightarrow} q$ then $X \notin a^+$.

The central tool in building the fixpoint automata is a subset construction which gives an intermediate automaton $\mathcal{A}'$ with nonstandard, structurally determined acceptance conditions. The states of $\mathcal{A}'$ are subsets of Q, and the initial state is the singleton $\{q_0\}$. For the transition relation let Σ be the set of variables free in ϕ excluding X, and let a range over $At(\Sigma)$. There are two cases. Case (1) handles the situation where no reference to the recursion variable X is needed, and case (2) the situation where it is.

(1) Suppose that

(i) $q_1 \stackrel{a_1}{\rightarrow} q'_1, \ldots, q_m \stackrel{a_m}{\rightarrow} q'_m$

(ii) $a = (a_1^+ \cup \cdots \cup a_m^+, a_1^- \cup \cdots \cup a_m^-)$.

Then $\{q_1, \ldots, q_m\} \stackrel{a}{\rightarrow} \{q'_1, \ldots, q'_m\}$.

(2) Suppose that $n \geq 1$ and that

(i) $q_1 \stackrel{a_1}{\rightarrow} q'_1, \ldots, q_m \stackrel{a_m}{\rightarrow} q'_m$,

(ii) $q_{m+1} \stackrel{(a^+_{m+1} \cup \{X\}, a^-_{m+1})}{\longrightarrow} q'_{m+1}, \cdots, q_{m+n} \stackrel{(a^+_{m+n} \cup \{X\}, a^-_{m+n})}{\longrightarrow} q'_{m+n}$,

(iii) $q_0 \stackrel{a_{m+n+1}}{\longrightarrow} q'_{m+n+1}$,

(iv) $a = (a_1^+ \cup \cdots \cup a^+_{m+n+1}, a_1^- \cup \cdots \cup a^-_{m+n+1})$.

Then $\{q_1, \ldots, q_{m+n}\} \stackrel{a}{\rightarrow} \{q'_1, \ldots, q'_{m+n+1}\}$.

Let S range over subsets of Q and assume that $S \stackrel{a}{\rightarrow} S'$. A member q of S can generate members q' of S' in one of two ways: either directly, because of an $\mathcal{A}$-transition, or indirectly because of a reference to the recursion variable. Formally, the generation relation $\multimap \subseteq S \times S'$ is determined in the following way: In case

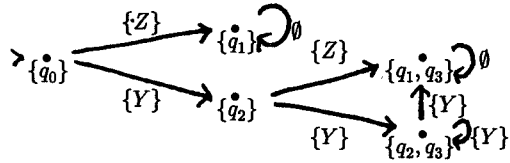

Figure 4: Intermediate automaton $\mathcal{A}'_1$

(1) above we let $q_i \rightarrowtriangle q'_j$ only if $i = j$, and q'_j is then the direct descendant of q_i. In case (2) we let $q_i \rightarrowtriangle q'_j$ only if either $i = j$ in which case q'_j is the direct descendant of q_i, or $m < i \leq m + n$ and $j = m + n + 1$, in which case q'_j is the indirect descendant of q_i. Consider a run Π through $\mathcal{A}'$ and any word π over formulas of the same length as Π with the property that whenever $\pi(i+1)$ is defined then $\pi(i) \rightarrowtriangle \pi(i+1)$ relative to the transition $\Pi(i) \xrightarrow{a} \Pi(i+1)$. We call π a *trail* through Π, written as $\pi \in \Pi$. If $\pi(i+1)$ is the direct descendant of $\pi(i)$ for all i for which $\pi(i+1)$ is defined then π is a *direct trail*.

The following characterisations can be proved using for instance the model checker of Bradfield and Stirling [3] or the model characterisations of Streett and Emerson [17] or Vardi [20].

Theorem 4.1 *The following statements are equivalent:*

(1) $0 \in \mathcal{M}(\nu X.fm(\mathcal{A}))$.

(2) There is an infinite run Π *through* $\mathcal{A}'$ *with the property that whenever* $\pi \in \Pi$ *and* π^i *is a direct trail then* $\pi(j) \in F$ *for infinitely many* j. □

Theorem 4.2 *The following statements are equivalent:*

(1) $0 \in \mathcal{M}(\mu X.fm(\mathcal{A}))$.

(2) There is an infinite run Π *through* $\mathcal{A}'$ *with the property that whenever* $\pi \in \Pi$ *then* π^i *is a direct trail for some* i *and* $\pi(j) \in F$ *for infinitely many* j. □

The trail π is *successful* if π^i is a direct trail for some i and $\pi(j) \in F$ for infinitely many j. A run is *ν-successful* if it has the the property (2) of Theorem 4.1, and it is *μ-successful* if it satisfies (2) of Theorem 4.2.

Example 4.3 (1) The automaton $\mathcal{A}'_1$ of fig. 4 is the intermediate automaton obtained from $\mathcal{A}_1$ of fig. 1 with respect to the recursion variable X. Here states that are not accessible from the initial state have for clarity been removed. Note that all infinite runs through $\mathcal{A}'_1$ are ν-successful, and that only runs that eventually visits the state $\{q_1, q_3\}$ are μ-successful.

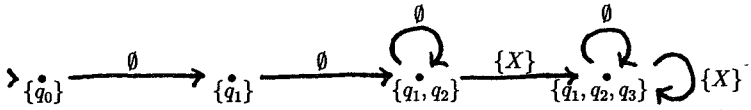

Figure 5: Intermediate automaton $\mathcal{A}'_2$

(2) The automaton $\mathcal{A}'_2$ of fig 5 shows the intermediate automaton resulting from the automaton $\mathcal{A}_2$ of fig. 2, constructed with respect to recursion variable Z. Again non-accessible states have been removed. A run through $\mathcal{A}'_2$ is ν-successful if the transition $\{q_1, q_2, q_3\} \stackrel{\{X\}}{\rightarrow} \{q_1, q_2, q_3\}$ is taken infinitely often. There are no μ-successful runs through $\mathcal{A}'_2$. □

5 Greatest fixed points

For greatest fixed points Theorem 4.1 gives rise to a natural idea of resolution of eventualities. Consider a finite run Π from S_1 to S_2 in $\mathcal{A}'$, let $q \in S_1$ and $\pi \in \Pi$ be the direct trail from q. We can view q as resolved at S_2 if $\pi(j)$ is an accepting state for some j. Let then $pending(\Pi)$ be the subset of S_1 of states that are not resolved at S_2. The idea of the rewriting procedure is embodied by the following easy Lemma:

Lemma 5.1 *An infinite run Π through $\mathcal{A}'$ is ν-successful iff there is a node S and an infinite, strictly increasing sequence $j_0, j_1, \ldots$ such that for all $k \in \omega$,*

(i) $\Pi(j_k) = S$, *and*

(ii) $pending(\Pi(j_k, j_{k+1})) = \emptyset$. □

For each node S the automaton $\mathcal{A}^{\nu}_S$ handles the situation where S is visited infinitely often by an infinite run through $\mathcal{A}'$. The desired automaton, $\nu X.\mathcal{A}$, is then built as the sum of the $\mathcal{A}^{\nu}_S$. Each $\mathcal{A}^{\nu}_S$ is defined in the following way:

(1) States are pairs (T, T') where T is a node, and $T' \subseteq T$. The intention is that T' is the set of members of T currently pending.

(2) The initial state is the pair $(\{q_0\}, \{q_0\})$.

(3) There is a *transition* $(T_1, T'_1) \stackrel{a}{\rightarrow} (T_2, T'_2)$ iff $T_1 \stackrel{a}{\rightarrow} T_2$ in $\mathcal{A}'$, and either

  (i) T'_1 is nonempty, and T'_2 is the set of all $q_2 \in T_2 - F$ such that q_2 is the direct descendant of some $q_1 \in T'_1$, or

  (ii) T'_1 is empty, and then T'_2 is the set of all $q_2 \in T_2 - F$ such that q_2 is the direct descendant of some $q_1 \in T_1$.

(4) The single *accepting state* is the state $(S, \emptyset)$.

The correctness of this account is a direct consequence of Theorem 4.1:

Theorem 5.2 *The automaton $\nu X.\mathcal{A}$ accepts $\alpha_{\mathcal{M}}$ iff $0 \in \mathcal{M}(\nu X.fm(q_0))$.* □

6 Least fixed points

For least fixed points we have additionally to take account of trails that do not eventually coincide with a direct trail and are consequently unsuccessful. Let S be any node occurring infinitely often along some infinite run Π through $\mathcal{A}'$. The crucial observation is that it must be possible to order S in a way which prevents trails that are not eventually direct.

Lemma 6.1 *An infinite run Π through $\mathcal{A}'$ is μ-successful iff there is a node S, a linear order $<$ on S, and an infinite, strictly increasing sequence $j_0, j_1, \ldots$ such that for all $k \in \omega$,*

(1) $\Pi(j_k) = S$,

(2) $pending(\Pi(j_k, j_{k+1})) = \emptyset$, *and*

(3) whenever $\pi \in \Pi$ and $\pi(j_k, j_{k+1})$ is not a direct trail then $\pi(j_{k+1}) < \pi(j_k)$.

PROOF: The if direction is easily checked. For the only-if direction assume that Π is μ-successful. Let S be any node visited infinitely often by Π, and let $j_0, j_1, \ldots$ be any infinite, strictly increasing sequence of j_k such that $\Pi(j_k) = S$. For any $q \in S$ and $k \in \omega$ there is some k' such that $q \notin pending(\Pi(j_k, j_{k'}))$, so as S is finite we can assume both (1) and (2) to be satisfied.

We derive a subsequence and a linear ordering $<$ such that also (3) is satisfied. The ordering $<$ is obtained by defining inductively a numeration $p_0, \ldots, p_m$ of S. For the base case note that there must be some $p_0 \in S$ with the property that for infinitely many k,

$$\text{whenever } \pi \in \Pi \text{ and } \pi(j_k) = p_0 \text{ then } \pi^{j_k} \text{ is a direct trail.} \quad (4)$$

For assume this fails to hold. For each $q \in S$ there is some k_q with the property that whenever $k \geq k_q$ then there is a $\pi \in \Pi$ and $k' > k$ such that $\pi(j_k) = q$ and $\pi(j_k, j_{k'})$ is not a direct trail. Let k_0 be largest among $\{k_q \mid q \in S\}$ Pick any $p_0' \in S$. Then we find a $k_1 > k_0$ such that there is a trail $\pi_0 \in \Pi$ where $\pi_0(j_{k_0}, j_{k_1})$ is not direct, and $\pi_0(j_{k_0}) = p_0'$. And we find a $k_2 > k_1$ suct that there is a trail $\pi_1 \in \Pi$ where $\pi_1(j_{k_1}, j_{k_2})$ is not direct, and $\pi_1(j_{k_1}) = \pi_0(j_{k_1})$. Continuing ad infinitum an unsuccessful trail through Π is then pieced together This completes the base case. Note that at the end of the base case we can assume without loss of generality that (4) holds for all $k \in \omega$.

Suppose then we have obtained $p_0, \ldots, p_i$, and let $T_i = \{p_0, \ldots, p_i\}$. If $S = T_i$ we are done. Otherwise there must be some $p_{i+1} \in S - T_i$ such that for infinitely many k,

$$\text{whenever } \pi \in \Pi, \pi(j_k) = p_{i+1},\ k'' > k \text{ and } \pi(j_k, j_{k'}) \text{ is not direct then } \pi(j_{k'}) \in T_i. \tag{5}$$

For if this fails a contradiction is obtained as in the base case. Similarly we can assume here that (5) holds for all $k \in \omega$.

We then define $<$ in the obvious way, by letting $p_i < p_j$ iff $i < j$. It follows that (3) above is satisfied, and the proof is complete. □

Reflecting Lemma 6.1 the automata $\mathcal{A}^{\mu}_{S}$ are built as the sum of automata $\mathcal{A}^{\mu}_{(S,<)}$ where $<$ is a linear ordering of S. In order to check that $<$ is not violated each automaton $\mathcal{A}^{\mu}_{(S,<)}$ must take into account the states that are accessible both directly and indirectly. For this purpose we define the sets $\mathrm{dir}(T_1) \subseteq S_2$ and $\mathrm{ind}(T_1) \subseteq S_2$ when $T_1 \subseteq S_1$ and $S_1 \xrightarrow{a} S_2$ in $\mathcal{A}'$:

$$\begin{aligned} \mathrm{dir}(T_1) &= \{p_2 \in S_2 \mid \exists p_1 \in T_1 . p_2 \text{ is the direct descendant of } p_1\} \\ \mathrm{ind}(T_1) &= \{p_2 \in S_2 \mid \exists p_1 \in T_1 . p_2 \text{ is the indirect descendant of } p_1\} \end{aligned}$$

The states of $\mathcal{A}^{\nu}_{S}$ are augmented by mappings f which given any member q of S produces a pair (T, T') such that T is the subset of the current node which is directly accessible from the last visit to q in S, and T' the subset which is indirectly accessible. The initial state of $\mathcal{A}^{\mu}_{(S,<)}$ is the state (S, S, f) where f maps each $q \in S$ into the pair $(\{q\}, \emptyset)$. For the transition relation we let $(S_1, S_1', f_1) \xrightarrow{a} (S_2, S_2', f_2)$ iff

(1) $(S_1, S_1') \xrightarrow{a} (S_2, S_2')$ in $\mathcal{A}^{\nu}_{S}$, and

(2) for all $q \in S$, if $f_1(q) = (T_1, T_1')$ then $f_2(q) = (\mathrm{dir}(T_1), \mathrm{dir}(T_1') \cup \mathrm{ind}(T_1) \cup \mathrm{ind}(T_1'))$.

To produce $\mathcal{A}^{\mu}_{(S,<)}$ it remains to fix the accepting state. For this purpose say that a node (S, S', f) is *consistent* with $<$ if whenever $q \in S$, $f(q) = (T, T')$ and $q' \in T'$ then $q' < q$. The accepting states of $\mathcal{A}^{\mu}_{(S,<)}$ are all states of the form $(S, \emptyset, f)$ that are consistent with $<$. The automaton $\mu X.\mathcal{A}$ is then obtained from $\mathcal{A}'$ by replacing each node S of $\mathcal{A}'$ with the sum of $\mathcal{A}'(S)$ and $\mathcal{A}^{\mu}_{S}$, so that runs are allowed to violate the ordering for an arbitrarily long initial segment. We obtain:

Theorem 6.2 *The automaton $\mu X.\mathcal{A}$ accepts $\alpha_{\mathcal{M}}$ iff $0 \in \mathcal{M}(\mu X.fm(q_0))$.* □

7 Efficiency concerns

The construction for greatest fixed points is $2^{\mathcal{O}(n)}$ and for least fixed points $2^{\mathcal{O}(n^2)}$ in the size of $\mathcal{A}$. A number of modifications are easily implemented which are

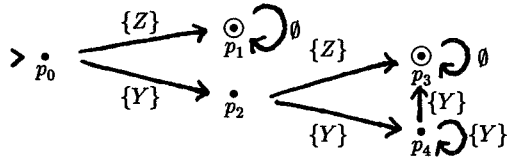

Figure 6: Büchi automaton $\mu X.\mathcal{A}_1$

expected to improve either or both the running time or the size of the resulting automaton very considerably in practical situations. Concerning first the construction for greatest fixed points these modifications are:

(1) A state (T, T') which is not accessible from (S, S) need not be constructed.

(2) If the accepting state $(S, \emptyset)$ is not accessible from the state (T, T') the latter need not be constructed.

(3) The set S can be required to contain an accepting state.

Additionally, for least fixed points the main modification is to restrict the number of orderings on S that need be considered. One idea is to *rank* the members of S such that for each possible ranking only one linear ordering $<$ need be considered. The ordering $<$ must have the property that whenever q_1 is of strictly smaller rank than q_2 then $q_1 < q_2$.

(4) If $q \in S$ has the property that no q' is accessible from q in $\mathcal{A}$ such that q' has an indirect descendant then q has rank 0.

(5) If $q \xrightarrow{(a_1^+ \cup \{X\}, a_1^-)} q'$, q' has rank n, and whenever $q_0 \xrightarrow{a_2} q''$, $a_1^+ \cap a_2^- = \emptyset$ and $a_1^- \cap a_2^+ = \emptyset$ then q'' has rank m or less, then q has rank $\max(n, m) + 1$.

(6) In relation to (3) above, S can be required to contain an accepting state of rank 0.

In the examples below the modifications 1 and 2 are assumed.

Example 7.1 (1) Fig. 6 shows the least fixed point automaton $\mu X.\mathcal{A}_1$ resulting from the intermediate automaton $\mathcal{A}_1'$ of fig. 4. The language recognised by $\mu X.\mathcal{A}_1$ is $\mu X.Z \vee (Y \wedge OX)$ in νTL or YUZ in PTL where U is the strong until-operator that requires Z eventually to hold. The greatest fixpoint automaton $\nu X.\mathcal{A}_1$ is obtained by letting in addition the state p_4 of fig 6 be accepting. The corresponding property in PTL is $YU'Z$ where U' is the weak until-operator that allows Z never to hold.

(2) The intermediate automaton $\mathcal{A}_2'$ of fig. 5 gives the greatest fixed point automaton $\nu Z.\mathcal{A}_2$ of fig. 7. In νTL the language recognised by $\nu Z.\mathcal{A}_2$

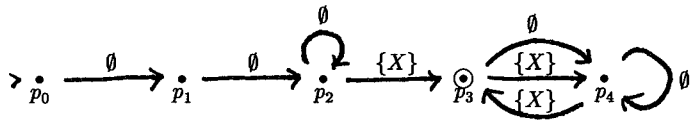

Figure 7: Büchi automaton $\nu Z.\mathcal{A}_2$

is $\nu Z.O((O(\mu Y.X \vee OY)) \wedge Z)$ equivalent to the PTL formula $GOOFX$ (and indeed GFX), expressing the fairness related property that X holds infinitely often. □

8 Axiomatising νTL

We have thus obtained a compositional procedure for deriving from each νTL formula ϕ an equivalent Büchi automaton $aut(\phi)$. Together with standard rules for the boolean connectives, the nexttime operator and an axiom and rule to govern the fixpoint operators (c.f. [9]), this suggests the following axiomatisation of νTL:

Axioms:

(1) All boolean tautologies

(2) $O(\phi \rightarrow \psi) \rightarrow O\phi \rightarrow O\psi$

(3) $O\phi \leftrightarrow \neg O \neg \phi$

(4) $\phi[\mu X.\phi/X] \rightarrow \mu X.\phi$

(5) $\phi \leftrightarrow fm(aut(\phi))$

Rules of deduction:

(6) *Detachment:* From $\phi \rightarrow \psi$ and ϕ infer ψ

(7) *Substitution:* From ϕ infer $\phi[\psi/X]$

(8) *Necessitation:* From ϕ infer $O\phi$

(9) *Fixpoint induction:* From $\phi[\psi/X] \rightarrow \psi$ infer $\mu X.\phi \rightarrow \psi$

Write $\vdash \phi$ iff ϕ is provable using this axiom system. Soundness is proved by induction in the size of proofs as usual.

Theorem 8.1 (Soundness) *If* $\vdash \phi$ *then* $\mathcal{M}(\phi) = \omega$ *for all models* $\mathcal{M}$. □

A formula ϕ is *consistent*, if $\not\vdash \phi \rightarrow false$. The following Lemma, due to Kozen [9], is instrumental for the completeness proof and provides a proof-theoretic correlate of Winskel's model-checking procedure in [22] using the forms $\mu X\{\vec{r}\}\phi$. The simple proof given here is due to Stirling.

Lemma 8.2 *If X is not free in ϕ and $\phi \wedge \mu X.\psi$ is consistent then so is $\phi \wedge \psi[X/\mu X.(\psi \wedge \neg\phi)]$.*

PROOF: If the conclusion fails then $\vdash \psi[X/\mu X.(\psi \wedge \neg\phi)] \rightarrow \neg\phi$. By propositional logic, $\vdash \psi[X/\mu X.(\psi \wedge \neg\phi)] \rightarrow \psi[X/\mu X.(\psi \wedge \neg\phi)] \wedge \neg\phi$. By axiom 4 it follows that $\vdash \psi[X/\mu X.(\psi \wedge \neg\phi)] \rightarrow \mu X.(\psi \wedge \neg\phi)$, so $\vdash \mu X.\psi \rightarrow \mu X.(\psi \wedge \neg\phi)$ by fixpoint induction, and then $\vdash \mu X.\psi \rightarrow \neg\phi$, a contradiction. □

Note that Lemma 8.2 does not rely on axiom schema 5.

Theorem 8.3 (Completeness) *If $\mathcal{M}(\phi) = \omega$ for all models $\mathcal{M}$ then $\vdash \phi$.*

PROOF: (Sketch) The result follows if we show that whenever ϕ is consistent then there is a model $\mathcal{M}$ such that $0 \in \mathcal{M}(\phi)$. By axiom schema 5 we can assume that ϕ has the form $fm(\mathcal{A})$. We then use Lemma 8.2 to show that if $fm(\mathcal{A})$ is consistent then there is an accepting state in $\mathcal{A}$ which is visited infinitely often along some run, and then it follows that $L(\mathcal{A})$ is nonempty. □

Note that only the left-to-right direction of axiom 5 is needed. Moreover axiom 5 can be reduced to the simpler schema

(5') $\mu X.fm(\mathcal{A}) \rightarrow fm(\mu X.\mathcal{A})$

Let $\vdash' \phi$ iff ϕ is provable using only 5' in place of 5, and let $\vdash'' \phi$ iff ϕ is provable without use of neither 5 nor 5'.

Theorem 8.4 *For all ϕ, $\vdash \phi$ iff $\vdash' \phi$.* □

The proofs of Theorem 8.4 and Theorem 8.5 below will be given in the full version of the paper. The proof of Theorem 8.4 involves showing:

(i) $\vdash'' (\neg)X \rightarrow fm(aut((\neg)X))$,

(ii) $\vdash'' Ofm(\mathcal{A}) \rightarrow fm(O\mathcal{A})$,

(iii) $\vdash'' fm(\mathcal{A}_1) \vee fm(\mathcal{A}_2) \rightarrow fm(\mathcal{A}_1 + \mathcal{A}_2)$,

(iv) $\vdash'' fm(\mathcal{A}_1) \wedge fm(\mathcal{A}_2) \rightarrow fm(\mathcal{A}_1 \times \mathcal{A}_2)$, and

(v) $\vdash'' \nu X.fm(\mathcal{A}) \rightarrow fm(\nu X.\mathcal{A})$.

Aconjunctivity [9] is a technical condition which, intuitively, disallows conjunctive branching of the regeneration relation for least fixed point formulas. Theorem 8.3 is not surprising in view of Kozen's completeness result for the aconjunctive fragment of the modal μ-calculus. Unlike the modal μ-calculus, however, for νTL the aconjunctive fragment is as expressive as the full language: All formulas of the form $fm(\mathcal{A})$ are aconjunctive, and ϕ and $fm(aut(\phi))$ are semantically equivalent for all ϕ. Indeed our constructions can be used to prove completeness for the aconjunctive fragment of νTL.

Theorem 8.5 (Completeness for Aconjunctive Fragment) *For all aconjunctive ϕ, if $\mathcal{M}(\phi) = \omega$ for all models $\mathcal{M}$ then $\vdash'' \phi$.* □

The proof amounts to showing for all ϕ:

(a) If $\mu X.\phi$ is aconjunctive then so is $\mu X.fm(aut(\phi))$.

(b) If $\mu X.fm(\mathcal{A})$ is aconjunctive then $\vdash'' \mu X.fm(\mathcal{A}) \rightarrow fm(\mu X.\mathcal{A})$.

It is not currently known whether (b) is provable without the assumption of aconjunctivity.

References

[1] B. Alpern and F. B. Schneider. Verifying temporal properties without temporal logic. *ACM Transactions on Programming Languages*, **11**:147–167, 1989.

[2] B. Banieqbal and H. Barringer. Temporal logic with fixed points. In Colloquium on Temporal Logic in Specification *Lecture Notes in Computer Science*, **398**:62–74, 1987.

[3] J. C. Bradfield and C. P. Stirling. Verifying temporal properties of processes. *Lecture Notes in Computer Science*, **458**:115–125, 1990. To appear in Theoretical Computer Science.

[4] R. Cleaveland, J. Parrow, and B. Steffen. A semantics based verification tool for finite state systems. In *Proc. 9th IFIP Symposium on Protocol Specification, Verification and Testing*, 1989.

[5] M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. In *Proceedings of the 17th Colloquium on Trees in Algebra and Programming* Lecture Notes in Computer Science, 581:145–164, 1992.

[6] E. A. Emerson and E. C. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th International Colloquium on Automata, Languages and Programming* Lecture Notes in Computer Science, **85**:169–181, 1980.

[7] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd Symposium on Foundations of Computer Science*, 1991.

[8] R. Kuiper H. Barringer and A. Pnueli. Now you may compose temporal logic specification. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 51–63, 1984.

[9] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, **27**:333–354, 1983.

[10] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of the 6th Annual Symposium on Logic in Computer Science*, pages 214–225, 1991.

[11] R. Kurshan. Analysis of discrete event coordination. In *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, Lecture Notes in Computer Science, 430:414–453, 1990.

[12] A. R. Meyer. Weak monadic second order theory of successor is not elementary recursive. *Lecture Notes in Mathematics*, 453:132–154, 1975.

[13] D. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, **104**:167–183, 1981.

[14] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pages 319–327, 1988.

[15] D. Siefkes. *Büchi's Monadic Second Order Successor Arithmetic.* Lecture Notes in Mathematics, 120. Springer-Verlag, 1970.

[16] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[17] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, **81**:249–264, 1989.

[18] W. Thomas. Computation tree logic and regular ω-languages. *Lecture Notes in Computer Science*, 354:690–713, 1988.

[19] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.), North-Holland, 1989.

[20] M. Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th Annual ACM Symposium on Principles of Programming Languages*, pages 250–259, 1988.

[21] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symposium on Logic in Computer Science*, pages 332–344, 1986.

[22] G. Winskel. Model checking the modal ν-calculus. *Lecture Notes in Computer Science*, 372, 1989.

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