Theoretical Computer Science 27 (1983) 333-354
North-Holland

RESULTS ON THE PROPOSITIONAL μ-CALCULUS

Dexter KOZEN*

Mathematical Sciences Department, IBM Research Center, Yorktown Heights, NY 10598, U.S.A.

Abstract. In this paper we define and study a propositional μ-calculus $L\mu$, which consists essentially of propositional modal logic with a least fixpoint operator. $L\mu$ is syntactically simpler yet strictly more expressive than Propositional Dynamic Logic (PDL). For a restricted version we give an exponential-time decision procedure, small model property, and complete deductive system, thereby subsuming the corresponding results for PDL.

1. Introduction

The propositional μ-calculus refers collectively to a class of programming logics consisting of propositional model logic with a least fixpoint operator μ. The μ-calculus originated with Scott and De Bakker [22] and was developed by Hitchcock and Park [7], Park [17], De Bakker and De Roever [2], De Roever [20] and others. The system we consider here is very close to a system appearing in [1]. The results of this volume, however, are mostly inspired by the work of Pratt [19], who defines a propositional μ-calculus $P\mu$, shows that $P\mu$ subsumes PDL, and extends the exponential-time decision procedure for PDL to $P\mu$. It is not known, however, whether $P\mu$ contains PDL strictly, and a deductive system is not given.

The usual proof rules for expressions involving least fixpoints do not readily apply to Pratt's $P\mu$ due to its formulation as a least root calculus rather than a least fixpoint calculus. This formulation was chosen in order to capture the reverse operator of PDL. Also, formulas of $P\mu$ are required to satisfy a rather strong condition akin to syntactic continuity. This condition renders illegal several useful formulas: e.g., the formula $\mu X.[b]X$, which is the same as $\neg\Delta b$ in the notation of Streett [21], expresses the property that the program b has no infinite computations. Pratt's syntactic restriction allows the filtration-based decision procedure of [18] to extend to $P\mu$, whereas no filtration-based decision procedure can work in the presence of $\mu X.[a]X$, as shown by Streett [21].

Here we propose weakening the syntactic continuity requirement and returning to a least fixpoint formulation. The resulting system is called $L\mu$. Although full $L\mu$ is decidable, the best bound known is nonelementary [16]. However, under a natural syntactic restriction which is still somewhat weaker than full syntactic continuity, better bounds can be obtained. For the syntactically restricted version, we show:

* These results were obtained during the author's sabbatical at the University of Aarhus, Denmark.

(1) $L\mu$, although syntactically simpler, is strictly more expressive than PDL. The strict containment result follows from a result of Streett [21]. $L\mu$ can express several natural PDL-ineffable formulas that are useful in program verification (see [4] for examples).

(2) $L\mu$ is decidable in deterministic exponential time, and is in fact exponential-time complete. This strengthens the corresponding result for PDL.

(3) There is a natural complete deductive system, involving the fixpoint induction rule of Park [17].

Familiarity with PDL and the concept of least fixpoints is assumed (see [1, 20, 6]).

2. Definition of $L\mu$ and $L\mu+$

$L\mu$ is essentially propositional modal logic with a least fixpoint operator μ. $L\mu+$ is an infinitary language containing $L\mu$, obtained by augmenting $L\mu$ with the ability to construct the α-fold composition of a monotone operator, where α is any ordinal. $L\mu+$ is useful in transfinite inductive arguments.

2.1. Syntax

The primitive nonlogical symbols of $L\mu$ and $L\mu+$ consist of *propositional constants* $P, Q, \ldots$, including the constants 0, 1, *propositional variables* $X, Y, \ldots$, and *program constants* $a, b, \ldots$. *Formulas* $p, q, \ldots$ of $L\mu+$ are defined inductively:

(2.1.1) X, (2.1.5) $\langle a\rangle p$,

(2.1.2) P, (2.1.6) $\alpha X.pX$, α an ordinal,

(2.1.3) $p \vee q$, (2.1.7) $\mu X.pX$,

(2.1.4) $\neg p$,

where in (2.1.6) and (2.1.7) pX is *positive* in the variable X, i.e., every free occurrence of X in pX occurs in the scope of an even number of negations $\neg$. (The notions of scope, bound and free occurrences of variables, closed formulas, etc. are the same as in first-order predicate logic, where μX and αX are treated as quantifiers.)

Intuitively, $\alpha X.pX$ represents the α-fold composition of the operator $\lambda X.pX$ applied to 0.

2.2. Semantics

A *model* is a structure $M = (S, I)$, where S is a set of *states* and I an interpretation of the propositional constants and program constants as, respectively, subsets of S and binary relations on S. We require that $I(0) = \emptyset$ and $I(1) = S$.

A *valuation* is a mapping assigning a subset of S to each variable. Formally, a formula p is interpreted as an operator p^M from valuations to subsets of S. However, since p^M will be independent of the variables not occurring free in p, we will view p^M as a function of its free variables. We will write $p(\bar{X})$ to denote that all free variables of p are among $\bar{X} = X_1, \ldots, X_n$, and $p^M(\bar{A})$ to denote the value of p^M on any valuation that assigns A_i to X_i, $1 \le i \le n$. The operator p^M is defined inductively as follows:

(2.2.1) $X_i^M(\bar{A}) = A_i$, (2.2.4) $(\neg p)^M(\bar{A}) = S - p^M(\bar{A})$,

(2.2.2) $P^M(\bar{A}) = I(P)$, (2.2.5) $(\langle a\rangle p)^M(\bar{A}) = \langle a^M\rangle(p^M(\bar{A}))$,

(2.2.3) $(p \vee q)^M(\bar{A}) = p^M(\bar{A}) \cup q^M(\bar{A})$,

where, in (2.2.5),

$$\langle a^M\rangle(B) = \{s \mid \exists t \in B, (s, t) \in I(a)\}.$$

To define (2.1.6) and (2.1.7), let pX be a formula positive in X, and let $\bar{X}$ denote the other free variables of pX. Thus $pX = p(X, \bar{X})$. We assume by induction hypothesis that the operator p^M has already been defined. Because of the requirement that pX be positive in X, the operator p^M is monotone in the variable X with respect to the subset relation.

(2.2.6a) $0X.pX^M(\bar{A}) = 0^M = \emptyset$,

(2.2.6b) $(\alpha+1)X.pX^M(\bar{A}) = p^M(\alpha X.pX^M(\bar{A}), \bar{A})$,

(2.2.6c) $\delta X.pX^M(\bar{A}) = \bigcup_{\beta<\delta} \beta X.pX^M(\bar{A})$, δ a limit ordinal,

(2.2.7) $\mu X.pX^M(\bar{A}) = \bigcup_\beta \beta X.pX^M(\bar{A})$,

where, in (2.2.7), the union is over all ordinals β. Taking $\mu > \alpha$ for any ordinal α, (2.2.6)(a–c) and (2.2.7) can be combined into the single definition:

(2.2.8) $\alpha X.pX^M(\bar{A}) = \bigcup_{\beta<\alpha} p^M(\beta X.pX^M(\bar{A}), \bar{A})$,

where α is either an ordinal or μ.

For fixed $\bar{A}$, since $p^M(X, \bar{A})$ is monotone in X, it follows that $\alpha X.pX^M(\bar{A}) \subseteq \beta X.pX^M(\bar{A})$ whenever $\alpha \le \beta$. At some level κ, we must have

$$\kappa X.pX^M(\bar{A}) = (\kappa+1)X.pX^M(\bar{A}) = \mu X.pX^M(\bar{A}).$$

The least such κ is called the *closure ordinal* of the operator $\lambda X.p^M(X, \bar{A})$.

It follows from the Knaster–Tarski theorem that $\mu X.pX^M(\bar{A})$ is the $\subseteq$-least fixpoint of the operator $\lambda X.p^M(X, \bar{A})$, and that

(2.2.9) $\mu X.pX^M(\bar{A}) = \bigcap\{B \mid p^M(B, \bar{A}) = B\} = \bigcap\{B \mid p^M(B, \bar{A}) \subseteq B\}$.

If p is closed, then p^M is constant. In this case s is said to *satisfy* p (notation: $M, s \models p$ or $s \models p$) if $s \in p^M$.

3. Notation and basic results

3.1. *Defined operators, positive normal form*

In addition to the primitive operators, we will use the usual defined Boolean operators $\wedge$, $\rightarrow$ and $\leftrightarrow$, as well as the defined operators

$$[a]p = \neg\langle a\rangle\neg p, \qquad \nu X.pX = \neg\mu X.\neg p\neg X.$$

The operator ν is the *greatest fixpoint operator*. It follows from (2.2.9) that $\nu X.pX$ is the greatest fixpoint of the map $\lambda X.pX$, i.e.,

$$(3.1.1)\quad \nu X.pX^M(\bar{A}) = \bigcup\{B \mid B = p^M(B, \bar{A})\} = \bigcup\{B \mid B \subseteq p^M(B, \bar{A})\}$$

and

$$(3.1.2)\quad ([a]p)^M(\bar{A}) = [a^M](p^M(\bar{A}))$$

by (2.2.5), where

$$[a^M](B) = \{s \mid \forall t, (s,t) \in I(a) \rightarrow t \in B\} = S - \langle a^M\rangle(S - B).$$

It is easily proved that every $L\mu$ formula is equivalent to a formula over $\vee$, $\wedge$, μ, ν, $\langle\ \rangle$, $[\ ]$, and $\neg$ in which $\neg$ is applied to primitive P only. Moreover, by renaming bound variables (Proposition 5.7(i) below), we can assume that no variable is quantified twice. Such a formula is said to be in *positive normal form*.

3.2. *Closure*

Let p_0 be a fixed closed formula in positive normal form. The following definitions introduce the *closure* $\mathrm{CL}(p_0)$ of p_0, the analog of the Fischer–Ladner closure of PDL [6]. For convenience, the closure is defined in terms of a mapping e on subformulas of p_0.

Let σ denote either μ or ν. If X is a bound variable of p_0, there is a unique μ- or ν-subformula $\sigma X.pX$ of p_0 in which X is quantified. We denote this subformula by σX. X is called a *μ-variable* if $\sigma X = \mu X.pX$ and a *ν-variable* if $\sigma X = \nu X.pX$.

Definition 3.2.1. Define $p \leqslant q$ if q appears as a subformula of p, and $p < q$ if q appears as a proper subformula of p. For $p_0 \leqslant p$, define $V_p = Y_1, \ldots, Y_n$, $n \geqslant 0$, to be the sequence of all variables Y such that $\sigma Y < p$, taken in the order

$$\sigma Y_1 < \cdots < \sigma Y_n < p.$$

For $\bar{X} = X_1, \ldots, X_k$ a subsequence of V_p, and $\bar{q} = q_1, \ldots, q_k$ a sequence of formulas, define

$$p[\bar{X}/\bar{q}] = p[X_k/q_k][X_{k-1}/q_{k-1}] \cdots [X_1/q_1],$$

where $p[X/q]$ denotes the formula p with all free occurrences of X replaced by q. Note that the order of substitution is from right to left.

Definition 3.2.2. If $V_p = X_1, \ldots, X_n$, let σV_p denote the sequence $\sigma X_1, \ldots, \sigma X_n$.

Define the map e on subformulas of p_0 by

$$e(p) = p[V_p/\sigma V_p].$$

The *closure* of p_0 is the range of e:

$$\mathrm{CL}(p_0) = \{e(p) \mid p_0 \leqslant p\}.$$

Note that $e(p)$ is closed, since if X occurs free in p, then $\sigma X \leqslant p$. It is immediately clear from the definition that $\mathrm{CL}(p_0)$ is a finite set, and is in fact no larger than $|p_0|$, the number of symbols of p_0. The next proposition relates $\mathrm{CL}(p_0)$ to the more usual notion of closure, as found for example in [6].

Proposition 3.2.3. *$\mathrm{CL}(p_0)$ is the smallest set of closed formulas such that*

(i) $p_0 \in \mathrm{CL}(p_0)$,
(ii) *if* $\neg P \in \mathrm{CL}(p_0)$, *then* $P \in \mathrm{CL}(p_0)$,
(iii) *if* $p \vee q \in \mathrm{CL}(p_0)$, *then* $p \in \mathrm{CL}(p_0)$ *and* $q \in \mathrm{CL}(p_0)$,
(iv) *if* $p \wedge q \in \mathrm{CL}(p_0)$, *then* $p \in \mathrm{CL}(p_0)$ *and* $q \in \mathrm{CL}(p_0)$,
(v) *if* $\langle a\rangle p \in \mathrm{CL}(p_0)$, *then* $p \in \mathrm{CL}(p_0)$,
(vi) *if* $[a]p \in \mathrm{CL}(p_0)$, *then* $p \in \mathrm{CL}(p_0)$,
(vii) *if* $\sigma X.pX \in \mathrm{CL}(p_0)$, *then* $p(\sigma X.pX) \in \mathrm{CL}(p_0)$.

Proof. It immediately follows from Definition 3.2.2 that

(viii) $e(p) = p$ if p is closed,
(ix) $e(p \vee q) = e(p) \vee e(q)$,
(x) $e(p \wedge q) = e(p) \wedge e(q)$,
(xi) $e(\langle a\rangle p) = \langle a\rangle e(p)$,
(xii) $e([a]p) = [a]e(p)$,
(xiii) $e(X) = e(\sigma X.p) = \sigma X.(p[V_{\sigma X}/\sigma V_{\sigma X}])$,

where, in (xiii), $\sigma X = \sigma X.p$. Cases (i) and (ii) are immediate from (viii). For case (iii), suppose $p \vee q \in \mathrm{CL}(p_0)$. Then $p \vee q = e(p' \vee q')$ for some subformula $p' \vee q'$ of p_0. By (ix), $p = e(p')$ and $q = e(q')$, therefore p, $q \in \mathrm{CL}(p_0)$. Cases (iv)–(vi) are similar. For case (vii), suppose $\sigma X.pX \in \mathrm{CL}(p_0)$. Formula $\sigma X.pX$ has exactly two pre-images under e, namely X and $\sigma X = \sigma X.p'X$. Then $p_0 \leqslant p'X$, and

$$e(p'X) = p'X[X/\sigma X][V_{\sigma X}/\sigma V_{\sigma X}] = p'(\sigma X.p'X)[V_{\sigma X}/\sigma V_{\sigma X}] = p(\sigma X.pX),$$

therefore $p(\sigma X.pX) \in \mathrm{CL}(p_0)$. $\square$

3.3. *Active variables and aconjunctivity*

Definition 3.3.1. Let p_0 be in positive normal form, $p_0 < p$. A variable Y of p_0 is called *active* in p if $\sigma Y < p$ and $p[\bar{X}/\sigma\bar{X}]$ contains a free occurrence of Y, where $\bar{X}$ is the subsequence of V_p consisting of those variables X for which $\sigma Y < \sigma X < p$.

The subsequence of V_p consisting of the active variables of p is denoted A_p. The subsequence of A_p consisting of the active μ-variables (resp. ν-variables) is denoted $A\mu_p$ (resp. $A\nu_p$).

If X is free in p, then X is active in p, but not vice versa in general; e.g., in

(3.3.2) $\mu X.\nu Y.(X \wedge \mu Z.(\langle a\rangle Y \vee [b]Z))$,

X is not free in $\langle a\rangle Y$ but is active in $\langle a\rangle Y$, since

$$\langle a\rangle Y[Z/\sigma Z][Y/\sigma Y] = \langle a\rangle \nu Y.(X \wedge \mu Z.(\langle a\rangle Y \vee [b]Z))$$

contains a free occurrence of X. However, the relation 'is active in' is somewhat like the transitive closure of the relation 'is free in', in the following sense.

Lemma 3.3.3. *If Y is active in p, and X is active in σY, then X is active in p.*

Proof. Let $\bar X = X_1, \ldots, X_m$, $\bar Y = Y_1, \ldots, Y_n$, $m, n \geq 0$, be all variables such that

$$\sigma X < \sigma X_1 < \cdots < \sigma X_m < \sigma Y < \sigma Y_1 < \cdots < \sigma Y_n < p.$$

Then Y is free in $p[\bar Y/\sigma \bar Y]$ and X is free in $\sigma Y[\bar X/\sigma \bar X]$, therefore X is free in

$$p[\bar Y/\sigma \bar Y][Y/\sigma Y[\bar X/\sigma \bar X]][\bar X/\sigma \bar X] = p[\bar Y/\sigma \bar Y][Y/\sigma Y][\bar X/\sigma \bar X]. \quad \square$$

The problem of determining whether X is active in p can be reformulated as a transitive closure problem, and any standard algorithm for computing the transitive closure of a binary relation will be efficient enough for our purposes.

Definition 3.3.4. Let p_0 be in positive normal form. p_0 is *aconjunctive in the μ-variable X* if, whenever $p_0 < p \wedge q$, X is active in at most one of p, q. p_0 is *aconjunctive* if it is aconjunctive in every μ-variable.

Example (3.3.2) above is not aconjunctive, because $X \wedge \mu Z.(\langle a\rangle Y \vee [b]Z)$ is a subformula of (3.3.2), and the μ-variable X is active in both X and $\mu Z.(\langle a\rangle Y \vee [b]Z)$.

Aconjunctivity is a technical restriction that is used in the proof of Theorem 6.3.1. It is related to, albeit weaker than, syntactic continuity. It is difficult to give the intuition behind the concept of aconjunctivity out of context; we therefore defer further explanation until Section 6.

4. Expressiveness results

$L\mu$ subsumes PDL without the reverse operator, as noted by Pratt [19]. The only least fixpoints PDL can express are of the form $\langle a^*\rangle p$, which in $L\mu$ is expressed $\mu X.p \vee \langle a\rangle X$. Thus $\langle a^*\rangle p$ is the least fixpoint of the monotone operator $\lambda X.p \vee \langle a\rangle X$.

This operator is continuous in X, in the sense that

$$p \vee \langle a\rangle\Big(\bigcup_i A_i\Big) = \bigcup_i (p \vee \langle a\rangle A_i).$$

If any model M, if pX is continuous in X, then

$$\mu X.pX^M = \omega X.pX^M,$$

i.e., the inductive definition of $\mu X.pX$ given in (2.2.7) above need not go beyond ω.

However, there are many non-continuous operators that are potentially useful in program verification. An interesting example is provided by the operator $\lambda X.[a]X$. Its least fixpoint in any model M is

$$\mu X.[a]X^M = \{s \mid \text{there are no infinite } a\text{-paths out of } s\} = \neg\Delta a,$$

where Δ is the loop operator of Streett [21]. $\mu X.[a]X$ is a well-formed formula of $L\mu$, even under the restriction of aconjunctivity, but is illegal in Pratt's system. In the model of Fig. 1, the operator $\lambda X.[a]X$ does not close at ω, since the top state satisfies $(\omega+1)X.[a]X$ but not $\omega X.[a]X$. Thus $\lambda X.[a]X$ is monotone but not continuous.

There are many useful properties that can be expressed with non-continuous operators, including liveness and fairness properties. The prototype liveness property

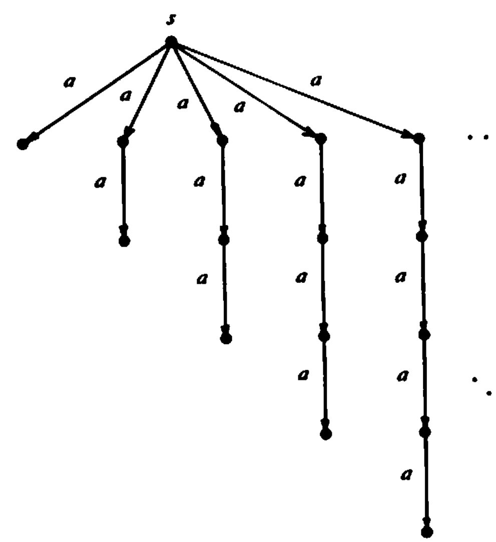

Fig. 1.

'along every a-path, p must eventually come true' is expressed as $\mu X.p \vee [a]X$ in $L\mu$. We refer the reader to [4] for further examples.

The question raised by Pratt about the strict expressiveness of $P\mu$ over PDL is still open, but the following result of Streett shows that $L\mu$, even restricted to aconjunctive formulas, is strictly more expressive than PDL. The proof also reveals why filtration techniques, which are used to obtain complexity and completeness results for PDL, fail for $L\mu$.

Proposition 4.1 ([21]). *$\mu X.[a]X$ is not equivalent to any* PDL *formula.*

Proof. Suppose $\mu X.[a]X \equiv p$ in all models, where p is a formula of PDL. In the model M of Fig. 1, $s \vDash \mu X.[a]X$, therefore $s \vDash p$. The proof of the small model property of PDL [6] allows M to be collapsed to a finite model N by identifying states that are indistinguishable by formulas of FL(p), the *Fischer–Ladner closure* of p. If $[t]$ is the equivalence class of t in the collapsed model, then $N, [t] \vDash q$ iff $M, t \vDash q$ for any $q \in$ FL(p). In particular, $[s] \vDash p$. But $[s]$ cannot satisfy $\mu X.[a]X$, since the collapsing must have created a loop, therefore there is an infinite a-path out of $[s]$. □

The above proof assumes that $\mu X.[a]X \equiv p$ in all models and derives a contradiction. However, it is possible to show that $L\mu$ is strictly more expressive than PDL in the stronger sense that there is a model M and a formula q of $L\mu$ such that no PDL formula p is equivalent to q on M.

Proposition 4.2 ([12]). *In the model given by Fig. 2, the formula $\mu X.[a]\langle a\rangle X$ defines the even states, whereas all* PDL *formulas, even with test and reverse, define only finite and cofinite sets.*

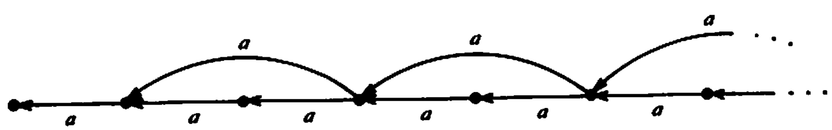

Fig. 2.

Intuitively, PDL cannot simulate an unbounded alternation of $[a]$ and $\langle a\rangle$.

Full $L\mu$ encodes ΔPDL of Streett [21], since $\Delta a = \nu X.\langle a\rangle X$. Under the restriction of aconjunctivity, $L\mu$ can be shown to encode well-structured ΔPDL, which is ΔPDL with the * and $\cup$ operators constrained to appear only in the context of the deterministic program constructors

$$\textbf{if } p \textbf{ then } a \textbf{ else } b = p?; a \cup \neg p?; b \quad \text{and} \quad \textbf{while } p \textbf{ do } a = (p?; a)^*; \neg p?.$$

Primitive programs need not be deterministic (see [8]).

5. A deductive system

The deductive system is equational, as in [15], involving equations $p \equiv q$ and inequalities $p \leq q$. The latter can be considered as an abbreviation for $p \vee q \equiv q$. The logical axioms and rules are those of equational logic, including substitution of equals for equals, provided the syntactic restrictions on μ formulas are not violated. The nonlogical axioms are the following:

(5.1) axioms for Boolean algebra,

(5.2) $\langle a\rangle X \vee \langle a\rangle Y \equiv \langle a\rangle(X \vee Y)$,

(5.3) $\langle a\rangle X \wedge [a]Y \leq \langle a\rangle(X \wedge Y)$,

(5.4) $\langle a\rangle 0 \equiv 0$,

(5.5) $p(\mu X.pX) \leq \mu X.pX$, $\mu X.pX$ free for X in pX,

(5.6) $pY \leq Y \Rightarrow \mu X.pX \leq Y$, Y does not occur in pX.

A formula p is *consistent* if not $\vdash p \equiv 0$. Axioms (5.1)–(5.4) are those of propositional modal logic. Axioms (5.5) and (5.6) say that $\mu X.pX$ is the $\leq$-least object A such that $p(A) \leq A$. Axiom (5.6) is the fixpoint induction rule of Park [17].

The following are some basic theorems of this system. The reader is referred to [1, 20] for the proofs, which are omitted here.

Proposition 5.7. (i) (*Change of bound variable*)

$\mu X.pX \equiv \mu Y.pY$, *X, Y free for Z in pZ,*

(ii) $pX \leq qX \Rightarrow \sigma X.pX \leq \sigma X.qX$,

(iii) (*Monotonicity*)

$q \leq r \Rightarrow p(q) \leq p(r)$, *$X$ positive in pX,*

(iv) $p(\sigma X.pX) \equiv \sigma X.pX$, *$\sigma X.pX$ free for X in pX,*

(v) $\mu X.q \equiv q$, *X not free in q,*

(vi) $p(\mu X.q \wedge pX) \leq q \Rightarrow \mu X.pX \leq q$, *$q$ free for X in pX.*

Proof of (vi)

(a) $p(\mu X.q \wedge pX) \leq q$ (by assumption),

(b) $p(q \wedge \mu X.(q \wedge p(q \wedge X))) \leq q$ (by (a), (5.1) and (iii)),

(c) $p(q \wedge \mu X.(q \wedge p(q \wedge X))) \leq q \wedge p(q \wedge \mu X.(q \wedge p(q \wedge X)))$ (by (b) and (5.1)),

(d) $p(q \wedge \mu X.(q \wedge p(q \wedge X))) \leq \mu X.(q \wedge p(q \wedge X))$ (by (c) and (5.5)),

(e) $p(q \wedge \mu X.(q \wedge p(q \wedge X))) \leq q \wedge \mu X.(q \wedge p(q \wedge X))$ (by (b), (d) and (5.1)),

(f) $\mu X.pX \leq q \wedge \mu X.(q \wedge p(q \wedge X))$ (by (e) and (5.6)),

(g) $\mu X.pX \leq q$ (by (f) and (5.1)). □

Implication (vi) of Proposition 5.7 is crucial in the proof of Theorem 6.3.1. We will use it in its dual form: if $q \wedge \mu X.pX$ is consistent, then $q \wedge p(\mu X.\neg q \wedge pX)$ is consistent. This is the proof-theoretic analog of the following model-theoretic intuition: If there is a state of the model M satisfying $q \wedge \mu X.pX$, then there must be a least α such that there is a state of M satisfying $q \wedge \alpha X.pX$.

6. Complexity and deductive completeness

In this section we prove completeness of the deductive system of Section 5 and give an exponential time decision procedure and small model property for $L\mu$ under the restriction of aconjunctivity. $L\mu$ is decidable without this restriction [16], but is not known to be elementary. These results are proved simultaneously, using a tableau method.

6.1. Construction of the tableau

Let p_0 be in positive normal form. In this section we construct a tableau T for p_0. T is a labeled tree constructed inductively downward by applying the extension rules described below. Certain edges of T will be labeled with primitive programs, others will be unlabeled. Each node s of T will be labeled with a set Γ_s of subformulas of p_0.

Initially, T consists of a single node r_0 labeled $\{p_0\}$. The tree is extended downward by applying the following five extension rules to the leaves, in an order to be specified later.

(6.1.1) $\wedge$*-rule.* If $p \wedge q \in \Gamma_s$, create node t with $\Gamma_t = (\Gamma_s - \{p \wedge q\}) \cup \{p, q\}$ and unlabeled edge $s \to t$.

(6.1.2) $\vee$*-rule.* If $p \vee q \in \Gamma_s$, create two new nodes t, u with $\Gamma_t = (\Gamma_s - \{p \vee q\}) \cup \{p\}$, $\Gamma_u = (\Gamma_s - \{p \vee q\}) \cup \{q\}$ and unlabeled edges $s \to t$, $s \to u$.

(6.1.3) σ*-rule.* If $\sigma X.pX \in \Gamma_s$, create t labeled $\Gamma_t = (\Gamma_s - \{\sigma X.pX\}) \cup \{pX\}$ and unlabeled edge $s \to t$.

(6.1.4) X*-rule.* If $X \in \Gamma_s$ and if $\sigma X = \sigma X.pX$, create t labeled $\Gamma_t = (\Gamma_s - \{X\}) \cup \{pX\}$ and unlabeled edge $s \to t$.

(6.1.5) $\langle\ \rangle$*-rule.* For each $\langle b\rangle p \in \Gamma_s$, create t labeled $\Gamma_t = \{p\} \cup \{q \mid [b]q \in \Gamma_s\}$ and edge $s \to t$ labeled b.

Note that the $\vee$-rule creates two new successors, the $\langle\ \rangle$-rule creates a new successor for each formula of the form $\langle b\rangle p$, and all other rules create one new successor. In the last case, the unique successor of s is denoted $s+$.

The construction process maintains several lists C of integer counters c, which count applications of the X-rule to active variables of formulas in Γ_s. There is one list $C(s, p)$ for each $p \in \Gamma_s$, and the lists are disjoint. If $A_p = X_1, \ldots, X_n$, then $C(s, p) = (c_1, \ldots, c_n)$, where c_i counts applications of the X-rule to X_i. The counter c_i is associated with X_i throughout its lifetime. We denote this correspondence by $X(c_i) = X_i$. In general, there may be several counters at node s associated with the same variable X, since X may be active in several formulas of Γ_s, but these counters will appear on different lists.

The integer value contained in c at node s is denoted $c(s)$. If $X(c)$ is a μ-variable, c is called a μ*-counter*, and $c(s)$ will always fall in the interval $0 \leq c(s) \leq 2^{|p_0|}$. If $X(c)$ is a ν-variable, c is called a ν*-counter*, and $c(s) \in \{0, 1\}$. A ν-counter c is used only as a one-bit flag to determine how recently the σ- or X-rule has been applied to $X(c)$.

If C is a list, let $C\mu$ (resp. $C\nu$) denote the sublist of C consisting of all μ-counters (resp. ν-counters). The construction process also maintains a global list G consisting of all existing μ-counters. $G(s)$ is a shuffle-merge of the lists $C\mu(s, p)$, $p \in \Gamma_s$. Thus the order of the μ-counters in G is consistent with their order on the lists $C\mu$. Whereas the order of the counters c on C is static and determined by the order $<$ on $\sigma X(c)$, the order on the global list G is dynamic and depends on the construction up to that point. $G(s)$ imparts a priority to the μ-counters existing at s, with the leftmost of highest priority.

The lists and counters are maintained as follows. We start with a single list $C(r_0, p_0)$ at the root, and $C(r_0, p_0) = G(r_0) = (\ )$, since p_0 has no active variables. The lists and counters are updated at each application of an extension rule as follows.

(6.1.6) When the σ-rule is applied to $\sigma X.pX$ at node s, recall that Γ_{s+} is obtained from Γ_s by replacing $\sigma X.pX$ with pX. If X is free in pX, then pX has a new active variable that was not active in $\sigma X.pX$, namely X. A new counter c is created with $X(c) = X$ and $c(s+) = 0$, and we append c to the right end of $C(s, \sigma X.pX)$ to get $C(s+, pX)$. If X is a μ-variable, the new counter is also appended to the right end of G, indicating lowest priority. If X is not free in pX, then we take $C(s+, pX) = C(s, \sigma X.pX)$ and $G(s+) = G(s)$, but by Proposition 5.7(v) we can assume w.l.o.g. that this does not happen.

(6.1.7) When the $\vee$-rule is applied to $p \vee q$ at node s with successors t, u as in (6.1.2), recall that the formula p replaces $p \vee q$ in Γ_t and q replaces $p \vee q$ in Γ_u. We obtain $C(t, p)$ (resp. $C(u, q)$) from $C(s, p \vee q)$ be deleting all counters c such that $X(c)$ is not active in p (resp. q). Any deleted μ-counters also disappear from the global lists $G(t)$ and $G(u)$.

(6.1.8) When the $\wedge$-rule is applied to $p \wedge q$ at node s, then we obtain $C(s+, p)$, (resp. $C(s+, q)$) from $C(s, p \wedge q)$ by deleting all counters c such that $X(c)$ is not active in p (resp. q). The global list G remains unchanged. It is here that the condition of aconjunctivity is used: whereas a ν-counter on $C(s, p \wedge q)$ may appear on both $C(s+, p)$ and $C(s+, q)$, $C\mu(s, p \wedge q)$ cleanly splits into disjoint lists $C\mu(s+, p)$ and $C\mu(s+, q)$, since each μ-variable active in $p \wedge q$ is active in exactly one of p, q. If aconjunctivity were not satisfied, the μ-counters on G would have to be duplicated.

(6.1.9) When the X-rule is applied to a variable X at s, and $\sigma X = \sigma X.pX$, take $C(s+, pX) = C(s, X)$, and set $c(s+) = c(s)+1$, where c is the unique counter on $C(s+, pX)$ and $C(s, X)$ such that $X(c) = X$. Note that c appears rightmost on these lists, since $\sigma Y \leq \sigma X$ for all variables Y active in X. If X is a μ-variable, we reset all μ-counters of lower priority than c to 0 (recall that d is of lower priority than c if it appears to the right of c on the global list G). We also reset to 0 any ν-counter appearing on any $C(s+, p)$ to the right of some μ-counter that is incremented or reset to 0.

(6.1.10) When the $\langle\ \rangle$-rule is applied at s, then for any successor t, Γ_t is of the form $\{p, q_1, \ldots, q_n\}$, where $\langle b \rangle p$, $[b]q_i \in \Gamma_s$. Take $C(t, p) = C(s, \langle b \rangle p)$ and $C(t, q_i) = C(s, [b]q_i)$, $1 \leq i \leq n$. $G(t)$ is obtained from $G(s)$ by deleting all counters not appearing on $C(t, p)$ or some $C(t, q_i)$. All ν-counters are reset to 0.

(6.1.11) If $p \in \Gamma_s$ and the $\wedge$-, $\vee$-, σ-, or X-rule is applied at s to some $q \neq p$, and t is a successor of s, then $p \in \Gamma_t$. In this case we take $C(t, p) = C(s, p)$ and leave all counters on $C(t, p)$ intact.

(6.1.12) After updating the lists according to (6.1.6)–(6.1.11), $C(t, p)$ may be temporary ill-defined. For example, if p, $p \wedge q \in \Gamma_s$, and the $\wedge$-rule is applied to $p \wedge q$, then (6.1.8) defines $C(s+, p)$ to be a sublist of $C(s, p \wedge q)$, but (6.1.11) defines $C(s+, p) = C(s, p)$. For another example, if $\langle b \rangle p$, $[b]p \in \Gamma_s$ and the $\langle\ \rangle$-rule is applied, then, at the successor t corresponding to $\langle b \rangle p$, (6.1.10) defines $C(t, p) = C(s, \langle b \rangle p)$ and $C(t, p) = C(s, [b]p)$. Whenever such a conflict occurs, the list of higher priority is kept and the other is discarded, where the priority of a list is determined by the position in G of its highest priority μ-counter. If the lists contain no μ-counters, say $C = (c_1, \ldots, c_n)$ and $C' = (c'_1, \ldots, c'_n)$, then we discard C' and set $c_i :=$ $\max\{c_i, c'_i\}$, $1 \leq i \leq n$.

Whenever two lists C', C are in conflict and C' is the one that is discarded, we write $C' \to C$ and $c' \to c$ for $c' \in C'$, $c \in C$ with $X(c') = X(c)$.

(6.1.13) Whenever a μ-counter changes priority due to the deletion of a higher priority μ-counter, it is reset to 0. Whenever a μ-counter $c \in C$ is incremented or reset to 0, and d is a ν-counter appearing to the right of c on C, then d is also reset to 0.

6.2. *The algorithm*

We now describe an alternating Turing machine algorithm to construct the tableau. The algorithm starts with one process at the root r_0. It then applies the extension rules in a regular fashion, accepting or rejecting on certain conditions described below. When visiting node s of T, a process has representation of Γ_s written on its tape. It also maintains all the lists of counters as described above. At applications of the $\vee$-rule, it makes an existential branch, spawning two subprocesses, each taking one of the successors. At applications of the $\langle\ \rangle$-rule, it branches universally, spawning several processes, one for each successor.

At any node, the $\wedge$-, $\vee$-, σ- and X-rules are applied first. The X-rule may only be applied to a ν-variable $X \in \Gamma_s$ if $c(s) = 0$, where $c \in C(s, X)$ and $X(c) = X$.

Whenever one of the following conditions obtains, the process takes the indicated action.

(6.2.1) There exist P, $\neg P \in \Gamma_s$. Halt and reject.

(6.2.2) Some μ-counter exceeds $2^{|p_0|}$. Halt and reject.

(6.2.3) The only rule that applies is the $\langle\ \rangle$-rule (i.e., Γ_s contains only formulas of the form P, $\neg P$, $\langle a \rangle p$, $[a]p$, or ν-variables X whose counters are nonzero), and neither of the previous conditions holds. Apply the $\langle\ \rangle$-rule.

(6.2.4) No rule applies and none of the previous conditions hold. Halt and accept.

Let $|G|$ denote the maximum length of $G(s)$. Since $G(s)$ is a shuffle of at most $|p_0|$ lists $C\mu(s, p)$ and each $|C\mu(s, p)| \leq |p_0|$, $|G| \leq |p_0|^2$. The above algorithm requires at most $|p_0|^3$ space, enough to encode Γ_s and $|G| \leq |p_0|^2$ counters, each containing a nonnegative integer at most $2^{|p_0|}$. Despite the possibility of infinite computations, this alternating algorithm can be simulated in deterministic exponential time [3].

The next lemma is used here to show that one of the conditions (6.2.1)–(6.2.4) must obtain after a finite time. The lemma is used again in Section 6.3.

Definition 6.2.5. Let $s = s_1, t_1, s_2, t_2, \ldots, s_n, t_n = t$ be nodes along some path in T such that s_{i+1} is an immediate successor of t_i, $1 \leq i < n$. Let $c = c_1, \ldots, c_n$ be counters such that c_i exists in the interval $[s_i, t_i]$ and $c_i \to c_{i+1}$ at t_i (therefore c_i no longer exists at s_{i+1}). Let a_i be the number of times c_i is incremented in the interval $[s_i, t_i]$, and define

$$\alpha(c, s, t) = \sum_{1 \leq i \leq n} a_i.$$

Lemma 6.2.6. *If either* (i) *c is a μ-counter, or* (ii) *c is a ν-counter and the $\langle\ \rangle$-rule is not applied in the interval $[s, t]$, then*

$$\alpha(c, s, t) \leq |p_0|^4 2^{|p_0|}.$$

Proof. (i) Let c, s, t, c_i, s_i, t_i, $1 \le i \le n$, be as in Definition 6.2.5. Note that $X(c) = X(c_i)$, $1 \le i \le n$. Let $p_i \in \Gamma_{s_i}$ such that $c_i \in C(s_i, p_i)$, and let d_i be the leftmost μ-counter on $C(s_i, p_i)$. Using Lemma 3.3.3, it can be shown that d_i exists throughout the interval $[s_i, t_i]$ leftmost on the same list as c_i, and $d_1 \to \cdots \to d_n$. Since the priority of d_i never decreases, and d_{i+1} is of higher priority than d_i, the sequence $d_1 \to \cdots \to d_n$ is no longer than $|G|$.

Let $N = 2^{|p_0|}$, the maximum value of c_i. In the interval $[s_i, t_i]$, c_i's priority can increase at most $|G|$ times. Between priority changes, whenever c_i is reset to 0, a counter of higher priority is incremented. Thus c_i can be incremented or reset to 0 at most $N^{|G|}$ times before either c_i or a higher priority counter exceeds N and condition (6.2.2) obtains, causing the process to halt and reject. Thus c_i can change priority, be reset, or be incremented at most $|G|N^{|G|}$ times. This gives an upper bound on the a_i of Definition 6.2.5, thus

$$\alpha(c, s, t) \le |G|^2 N^{|G|} \le |p_0|^4 2^{|p_0|^3}.$$

(ii) If there exists a μ-variable Y active in $X(c)$, then for each i, there exists a μ-counter d_i appearing leftmost on the same list as c_i throughout the interval $[s_i, t_i]$. As above, the length of the sequence $c_1 \to \cdots \to c_n$ is at most $|G|$. Within the interval $[s_i, t_i]$, c_i can be reset to 0 only if the ( )-rule is applied (6.1.10) or some μ-counter to the left of c_i is incremented or reset to 0 (6.1.13). The former does not occur by assumption. The latter occurs only if the rightmost μ-counter to the left of c_i is incremented or reset to 0. By (i), this can happen at most $|G|N^{|G|}$ times, from which the bound follows.

If there does not exist a μ-variable active in $X(c)$, then c cannot be reset in the interval $[s, t]$, since neither (6.1.10) nor (6.1.13) occurs. Thus c or c_i can be incremented at most once in $[s, t]$, since the X-rule is never applied when a counter is nonzero, and therefore $\alpha(c, s, t) \le 1$. □

Lemma 6.2.7. *One of conditions* (6.2.1)–(6.2.4) *must obtain after a finite time.*

Proof. Suppose there were an infinite path in T with the $\vee$-, $\wedge$-, σ- and X-rules applied along that path without one of (6.2.1)–(6.2.4) ever obtaining. Each rule except the X-rule decreases the size of Γ_s, as measured by the total number of symbols in formulas in Γ_s, therefore there must exist a variable X to which the X-rule is applied infinitely often. Moreover, X can be chosen such that σX is $<$-minimal.

By Lemma 6.2.6, each c with $X(c) = X$ that exists along the path must disappear after a finite time. This says that a new counter for X is created infinitely often through application of the σ-rule. But then there must be a Y with $\sigma Y < \sigma X$ such that the X-rule is applied to Y infinitely often along the path, contradicting the $<$-minimality of X. □

6.3. *Proof of main theorem*

The following theorem asserts the correctness of the algorithm and the completeness of the deductive system of Section 5 simultaneously.

Theorem 6.3.1. *The following are equivalent*:
 (i) *p_0 is consistent,*
 (ii) *the algorithm does not reject,*
 (iii) *p_0 has a finite tree-like model of depth exponential in $|p_0|^3$.*

Proof of (i)→(ii). Suppose p_0 is consistent. First we construct a formula $e'(s, p)$ for each $p \in \Gamma_s$ such that $e'(s, p) \le e(p)$. $e'(s, p)$ is formed by conjoining certain closed formulas $r(s, c)$, $c \in C\mu(s, p)$ (to be defined later) with certain subformulas of $e(p)$, as follows. Let $V_p = \bar{X} = X_1, \ldots, X_m$. For $X_i \in A\mu_p$, $c_i \in C\mu(s, p)$ with $X(c_i) = X_i$ and $\sigma X_i = \mu X_i . qX_i$, let $q_i = \mu X_i . (r(s, c_i) \wedge qX_i)$. For $X_i \in V_p - A\mu_p$, let $q_i = \sigma X_i$. Define

$$e'(s, p) = p[\bar{X}/\bar{q}].$$

By Proposition 5.7(iii), $e'(s, p) \le e(p)$.

Each $r(s, c)$ consists of a conjunction of closed formulas, defined inductively down the tree. If neither the σ- nor the X-rule is applied at s, or if the σ- or X-rule is applied to a ν-variable, let

(6.3.2) $r(t, c) = r(s, c)$

for all successors t of s and counters $c \in G(t)$. If the σ-rule is applied to $\mu X.pX$ at s, yielding a new counter c on $C(s+, pX)$ with $X(c) = X$, define

(6.3.3) $r(s+, c) = \mathit{true}$,

(6.3.4) $r(s+, d) = r(s, d)$, $d \in C(s+, pX)$, $d \ne c$.

If the X-rule is applied to the μ-variable X at s, and $c \in C(s+, X)$ with $X(c) = X$, define

(6.3.5) $r(s+, d) = \mathit{true}$ if d is of lower priority than c,

(6.3.6) $r(s+, d) = r(s, d)$ if d is of higher priority than c,

(6.3.7) $r(s+, c) = r(s, c) \wedge \neg \wedge \Delta'_s$,

where

$$\Delta'_s = \{e'(s+, p) \mid p \in \Gamma_s, p \ne X\}.$$

The formula $r(s+, c)$ in (6.3.7) is well-defined, since (6.3.5) and (6.3.6) determine $r(s+, d)$ for all $d \ne c$, and these determine $e'(s+, p)$ for all $p \in \Gamma_s$, $p \ne X$, and hence determine Δ'_s.

Note that $r(s, c)$ consists of a conjunction of $c(s)$ closed formulas (by convention, $\wedge\emptyset = 1$):

$$r(s, c) = \neg\wedge\Delta'_{s_0} \wedge \cdots \wedge \neg\wedge\Delta'_{s_{c(s)-1}},$$

where s_i, $0 \leq i < c(s)$, is the most recent ancestor of s such that c had value i.

Let

$$\Delta_s = \{e'(s, p) \mid p \in \Gamma_s\}.$$

We now construct a set B of nodes of T containing the root r_0 such that

(6.3.8) if $s \in B$ and the $\vee$-rule was applied at s, then at least one successor of s is in B,

(6.3.9) for any other node $s \in B$, all successors of s are in B,

(6.3.10) for every $s \in B$, Δ_s is consistent.

The set B is constructed inductively down the tree. First set $B := \{r_0\}$; $\Delta_{r_0} = \{p_0\}$ is consistent by assumption.

Suppose $s \in B$ and the $\vee$-rule is applied to $p \vee q$ at s, and t, u are the two successors of s. If $p \in \Gamma_t$ already, and $C(s, p)$ is of higher priority than the sublist of $C(s, p \vee q)$ corresponding to the active variables of p, then the latter list is deleted in (6.1.12), so that $\Delta_t \subseteq \Delta_s$. Then Δ_t is consistent since Δ_s is, so we can extend B by taking $B := B \cup \{t\}$. Similarly, if $q \in \Gamma_u$ and $C(s, q)$ is of higher priority, then we can take $B := B \cup \{u\}$. If neither of the above cases holds, then

$$e'(s, p \vee q) \leq e'(t, p) \vee e'(u, q), \qquad \wedge\Delta_s \leq \wedge\Delta_t \vee \wedge\Delta_u.$$

By Axiom (5.1), one of Δ_t, Δ_u must be consistent, say Δ_t. Set $B := B \cup \{t\}$.

Similarly, at applications of the $\wedge$-, $\langle\ \rangle$- and σ-rules, and applications of the X-rule to ν-variables, B can be extended to include all successors, since if Δ_s is consistent, then Δ_t is consistent for all successors t.

At an application of the σ-rule to a μ-formula at s, a new $r(s+, c)$ appears, but it is *true* at that point, and so does not affect the consistency of Δ_{s+}, by Proposition 5.7(iii). At applications of the σ- and X-rule to ν-formulas, and applications of the $\wedge$-, $\vee$- and $\langle\ \rangle$-rules, no $r(s, c)$ changes.

It remains to show that Δ_{s+} is consistent when the X-rule is applied to a μ-variable X at $s \in B$. It is here that we use Proposition 5.7(vi). Let $\sigma X = \mu X.pX$ and let $c \in C(s, X)$ with $X = X(c)$. If $pX \in \Gamma_t$ already, and $C(s, pX)$ is of higher priority than $C(s, X)$, then $\Delta_t \subseteq \Delta_s$ as above, and we are done. Oherwise, Δ'_s is obtained from $\Delta_s - \{e'(s, X)\}$ by replacing some $r(s, d)$ with $r(s+, d) = true$, namely for those d of lower priority than c at s. Thus

(6.3.11) $\wedge\Delta_s \leq \wedge\Delta'_s \wedge e'(s, X)$.

Therefore, since Δ_s is consistent by hypothesis,

(6.3.12) $\Delta'_s \cup \{e'(s, X)\} = \Delta'_s \cup \{\mu X.(r(s, c) \wedge p'X)\}$

is consistent. Using Proposition 5.7(vi) it follows that

(6.3.13) $\Delta'_s \cup \{p'(\mu X.(r(s, c) \wedge \neg\wedge\Delta'_s \wedge p'X)\}$

is consistent. But (6.3.13) is equal to

$$\Delta'_s \cup \{p'(\mu X.(r(s+, c) \wedge p'X)\} = \Delta'_s \cup \{e'(s+, pX)\} = \Delta_{s+},$$

therefore the latter is consistent.

The above construction gives a subtree B satisfying conditions (6.3.8)–(6.3.10) above. We show now that if π is any process in the computation tree of the above alternating algorithm visiting node s of the tableau, and π is labeled 0 (reject), then $s \notin B$.

A process π may halt and reject outright because of either (6.2.1) or (6.2.2). In (6.2.1), there exist $P, \neg P \in \Delta_s$, therefore Δ_s is inconsistent by (5.1), hence $s \notin B$ by (6.3.10). In (6.2.2) there must exist two ancestors u, v of s such that $\Gamma_u = \Gamma_v$ and the X-rule is applied to the μ-variable $X(c)$ at u and v, thereby incrementing c at u and v, and c is not reset in the interval $[u, v]$. This also implies that the priority of c is unchanged between u and v. If $d \in G(u)$ of higher priority than c, then $d \in G(v)$ with the same priority, and $d(u) = d(v)$, otherwise c would have been reset between u and v. Then $r(u+, d) = r(v+, d)$. The set of counters of lower priority than c at u may differ from that at v, but $r(u+, d) = r(v+, d') = true$ for any such d, d', because these counters were reset to 0. Then

(6.3.14) $\Delta'_v = \Delta'_u$.

Now $\neg\wedge\Delta'_u$ appears in $r(u+, c)$ and hence in $r(v, c)$, and $e'(v, X) = \mu X.(r(v, c) \wedge pX) \in \Delta_v$, therefore

(6.3.15) $\wedge\Delta_v \leq r(v, c) \leq \neg\wedge\Delta'_u = \neg\wedge\Delta'_v$

by (5.1), Proposition 5.7(iii) and (6.3.14). On the other hand,

$$\wedge\Delta_v \leq \wedge\Delta'_v \quad \text{by (6.3.11)}.$$

Thus Δ_v is inconsistent, and $v \notin B$ by (6.3.10). Since v is an ancestor of s, $s \notin B$.

If π is a universal branch, then one of the successors ρ of π must be labeled 0 in the algorithm, and ρ is visiting a successor t of s. By induction, $t \notin B$, therefore $s \notin B$ by (6.3.9). If π is an existential branch, then both successors ρ, τ of π must be labeled 0 in the algorithm, and ρ, τ are visiting successors t, u of s. By induction, $t, u \notin B$, therefore $s \notin B$ by (6.3.8). Proceeding back up to the root, if the initial process π_0 were labeled 0, then $r_0 \notin B$, a contradiction. Therefore the algorithm does not reject.

Proof of (ii) → (iii). If the algorithm does not reject, prune all nodes of the tableau T visited by processes of the algorithm labeled 0 (reject). Prune further so that each $\vee$-node s has exactly one successor $s+$. The tree T' so obtained satisfies (6.3.8) and (6.3.9) above, and contains the root r_0.

We now define a model $M=(S, I)$ from T'. Let S be the set of nodes of T' such that either the $\langle\ \rangle$-rule was applied at s, or no rule was applicable at s (thus s is a leaf). Each edge out of a node in S is labeled with a unique program constant, and all other edges are unlabeled. For $s \in T'$, let $U(s)$ be the set of nodes of T' consisting of s and all ancestors on the path back up to, but not including, the most recent ancestor in S; or back up to and including the root, if no ancestor of s is in S. Note that if $s \in S$ and $s \to u$ in T', then by Lemma 6.2.7 there exists a unique node $t \in S$ such that $u \in U(t)$. For $s, t \in S$, let $(s, t) \in I(a)$ if there is an edge from s to a node in $U(t)$ labeled a. Let $s \in I(P)$ if $P \in \Gamma_s$.

We construct a set of closed formulas Θ_s of $L\mu+$ for each $s \in T'$, as follows. Let $p \in \Gamma_s$, $V_p = \bar{X} = X_1, \ldots, X_n$, $\sigma X_i = \sigma X_i.p_i X_i$. If $X_i \in A\mu_p$ and $c \in C(s, p)$ with $X_i = X(c)$, let

$$\alpha(s, c) = \sup_{t \in T'} \alpha(c, s, t), \qquad q_i = \alpha(s, c) X_i.p_i X_i,$$

where $\alpha(c, s, t)$ is given as in Definition 6.2.5. If $X_i \in V_p - A\mu_p$, let $q_i = \sigma X_i$. Let $\bar{q} = q_1, \ldots, q_n$ and define

$$e''(s, p) = p[\bar{X}/\bar{q}], \qquad \Theta'_s = \{e''(s, p) \mid p \in \Gamma_s\}, \qquad \Theta_s = \cup\{\Theta'_t \mid t \in U(s)\}.$$

Define $p' \sqsubseteq p$ if p' is identical to p except that some α occurring in subformulas $\alpha X.qX$ of p may be replaced by some $\beta < \alpha$. We show that, for all $s \in S$,

(6.3.16) if $p \wedge q \in \Theta_s$, then $p, q \in \Theta_s$,
(6.3.17) if $p \vee q \in \Theta_s$, then either $p \in \Theta_s$ or $q \in \Theta_s$,
(6.3.18) if $\langle a \rangle p \in \Theta_s$, then $\exists\, t \in S$, $(s, t) \in I(a)$ and $p \in \Theta_t$,
(6.3.19) if $[a]p \in \Theta_s$, then $\forall\, t \in S$, $(s, t) \in I(a) \to \exists\, p' \sqsubseteq p$ such that $p' \in \Theta_t$,
(6.3.20) if $\alpha X.pX \in \Theta_s$, α an ordinal or μ, then $p(\beta X.pX) \in \Theta_s$ for some $\beta < \alpha$,
(6.3.21) if $\nu X.pX \in \Theta_s$, then $p(\nu X.pX) \in \Theta_s$.

Proof of (6.3.16). If $p \wedge q \in \Theta_s$, then $\exists\, t \in U(s)$, $\exists\, p', q'$ with $p \wedge q = e''(t, p' \wedge q') \in \Theta'_t$, $p' \wedge q' \in \Gamma_t$, and the $\wedge$-rule applied to $p' \wedge q'$ at t. Then $p', q' \in \Gamma_{t+}$ and $p = e''(t+, p')$, $q = e''(t+, q')$, thus $p, q \in \Theta'_{t+} \subseteq \Theta_s$.

Proof of (6.3.17). The proof of (6.3.17) is similar to (6.3.16).

Proof of (6.3.18). If $\langle a \rangle p \in \Theta_s$, then $\langle a \rangle p \in \Theta'_s$. By the $\langle\ \rangle$-rule, some a-successor u of s has $p \in \Theta'_u$, and by Lemma 6.2.6 there is a unique descendant t of u with $t \in S$ and $u \in U(t)$. Then $(s, t) \in I(a)$ and $p \in \Theta_t$.

Proof of (6.3.19). This proof is similar to that of (6.3.18), except that $p' \sqsubseteq p$ appears in Θ_t instead of p, because if $[a]p'' \in \Gamma_s$ with $[a]p = e''(s, [a]p'')$, then, by (6.1.5), p'' appears in Γ_u for all a-successors u of s, and

$$\alpha(s, c) = \sup_u \alpha(t, c) \quad \text{for any } c \in C\mu(s, p''),$$

where the supremum is taken over all a-successors u of s. Some of these $\alpha(t, c)$ may be strictly less than $\alpha(s, c)$.

Proof of (6.3.20). Either $\alpha = \mu$ or $\alpha \in \omega$. If $\alpha = \mu$ and $\mu X.pX \in \Theta_s$, then $\exists\, t \in U(s)$, $\exists p'X$ with

$$\mu X.pX = e''(t, \mu X.p'X) = \mu X.p'X[\bar{X}/\bar{q}] \in \Theta'_t,$$

where $\bar{X} = V_{\mu X.p'X}$, $\mu X.p'X \in \Gamma_t$, and the σ-rule applied to $\mu X.p'X$ at t. Then $p'X \in \Gamma_{t+}$ and

$$e''(t+, p'X) = p'X[\bar{X}, X/\bar{q}, \alpha(t+, c)X.p'X] = p'(\alpha(t+, c)X.p'X)[\bar{X}/\bar{q}]$$
$$= p(\alpha(t+, c)X.pX) \in \Theta_{t+} \subseteq \Theta_s,$$

where $c \in C(t+, p'X)$ with $X = X(c)$.

If $\alpha \in \omega$ and $\alpha X.pX \in \Theta_s$, then $\exists\, t \in U(s)$, $\exists p'X$ with

$$\alpha X.pX = e''(t, X) = X[\bar{X}, X/\bar{q}, \alpha X.p'X] \in \Theta'_t,$$

where $\bar{X} = V_{\mu X.p'X}$, $X \in \Gamma_t$, the X-rule is applied to X at t, and $\alpha = \alpha(t, c)$ where $c \in C(t, X)$ with $X = X(c)$. Then $p'X \in \Gamma_{t+}$ and

$$e''(t+, p'X) = p'X[\bar{X}, X/\bar{q}, \alpha(t+, c')X.p'X]$$
$$= p'(\alpha(t+, c')X.p'X)[\bar{X}/\bar{q}]$$
$$= p(\alpha(t+, c')X.pX) \in \Theta_{t+} \subseteq \Theta_s,$$

where $c' \in C(t+, p'X)$ with $X = X(c')$. Either $c = c'$ or $c \to c'$, but in either case, $\alpha(t, c) = \alpha(t+, c') + 1$, therefore (6.3.20) is satisfied.

Proof of (6.3.21). If $\nu X.pX \in \Theta_s$, then $\exists\, t \in U(s)$, $\exists p'X$ such that $\nu X.pX = e''(t, \nu X.p'X)$ and the σ-rule is applied to $\nu X.p'X \in \Gamma_t$, or $\nu X.pX = e''(t, X)$ and $X \in \Gamma_t$. In the former case we proceed as in the proof of (6.3.16). In the latter case, if $c \in C(t, X)$ with $X(c) = X$, and $c(t) \neq 0$, then there must have been a most recent time t' at which the value of c changed from 0 to 1. Then any μ-counter d appearing to the left of c on $C(t', X)$ exists at t, and the X-rule is not applied to $X(d)$ in the interval $[t'+, t]$, nor is d reset, otherwise c would have been reset. Then $\alpha(d, t) = \alpha(d, t')$, therefore $\nu X.pX = e''(t', X)$, and the X-rule is applied to X at $\Gamma_{t'}$. As in the proof of (6.3.20),

$$e''(t'+, p'X) = p(\nu X.pX) \in \Theta_{t'+} \subseteq \Theta_s.$$

Now define

$$q^{\Theta} = \{s \in S \mid \exists\, q' \sqsubseteq q,\ q' \in \Theta_s\}.$$

Note that if $q' \sqsubseteq q$, then $q'^{\Theta} \subseteq q^{\Theta}$. If $\bar{q} = q_1, \ldots, q_n$, let $\bar{q}^{\Theta} = q_1^{\Theta}, \ldots, q_n^{\Theta}$. We show by induction on formula structure that, for any $p(\bar{X})$ and $\bar{q}$,

(6.3.22) $p(\bar{q})^{\Theta} \subseteq p^M(\bar{q}^{\Theta})$.

By definition of M,

$$P^{\theta}=\{s \mid P\in\Theta_s\}=P^M,$$

$$\neg P^{\theta}=\{s \mid \neg P\in\Theta_s\}\subseteq\neg P^M \quad \text{by (6.2.1) and (2.2.4)},$$

$$X_i(\bar q)^{\theta}=q_i^{\theta}=X_i^M(\bar q^{\theta}).$$

For the case $p\vee q$,

$$\begin{aligned} p\vee q(\bar q)^{\theta} &\subseteq p(\bar q)^{\theta}\cup q(\bar q)^{\theta} \quad \text{by (6.3.17)}\\ &\subseteq p^M(\bar q^{\theta})\cup q^M(\bar q^{\theta}) \quad \text{by induction hypothesis}\\ &=(p\vee q)^M(\bar q^{\theta}) \quad \text{by (2.2.3)}. \end{aligned}$$

The case $p\wedge q$ is similar, using (6.3.16). For the case $\langle a\rangle p$,

$$\begin{aligned} \langle a\rangle p(\bar q)^{\theta} &\subseteq \langle a^M\rangle(p(\bar q)^{\theta}) \quad \text{by (6.3.18)}\\ &\subseteq \langle a^M\rangle(p^M(\bar q^{\theta}))\\ &\quad \text{by induction hypothesis and the monotonicity of } \langle a^M\rangle\\ &=\langle a\rangle p^M(\bar q^{\theta}) \quad \text{by (2.2.5)}. \end{aligned}$$

For the case $[a]p$,

$$\begin{aligned} [a]p(\bar q)^{\theta} &\subseteq [a^M]\Big(\bigcup_{p(\bar q)'\subseteq p(\bar q)} p(\bar q)'^{\theta}\Big) \quad \text{by (6.3.19)}\\ &\subseteq [a^M](p(\bar q)^{\theta}) \quad \text{by monotonicity of } [a^M]\\ &\subseteq [a^M](p^M(\bar q^{\theta}))\\ &\quad \text{by induction hypothesis and the monotonicity of } [a^M]\\ &=[a]p^M(\bar q^{\theta}) \quad \text{by (3.1.2)}. \end{aligned}$$

For the case $\alpha X.pX$, where either $\alpha=\mu$ or $\alpha\in\omega$,

$$\begin{aligned} \alpha X.pX(\bar q)^{\theta} &\subseteq p(\beta X.pX(\bar q),\bar q)^{\theta} \quad \text{for some } \beta<\alpha, \text{ by (6.3.20)}\\ &\subseteq p^M(\beta X.pX(\bar q)^{\theta},\bar q^{\theta}) \quad \text{by induction hypothesis on } p\\ &\subseteq p^M(\beta X.pX^M(\bar q^{\theta}),\bar q^{\theta})\\ &\quad \text{by induction hypothesis on } \beta \text{ and the monotonicity of } p^M\\ &=(\beta+1)X.pX^M(\bar q^{\theta}) \quad \text{by (2.2.6b)}\\ &\subseteq \alpha X.pX^M(\bar q^{\theta}), \end{aligned}$$

since $\beta+1\leq\alpha$. Finally, for the case $\nu X.pX$,

$$\begin{aligned} \nu X.pX(\bar q)^{\theta} &\subseteq p(\nu X.pX(\bar q),\bar q)^{\theta} \quad \text{by (6.3.21)}\\ &\subseteq p^M(\nu X.pX(\bar q)^{\theta},\bar q^{\theta}) \quad \text{by induction hypothesis.} \end{aligned}$$

By (3.1.1), $\nu X.pX^M(\bar q^{\theta})$ is the greatest fixpoint of the operator $\lambda X.p^M(X,\bar q^{\theta})$, therefore

$$\nu X.pX(\bar q)^{\theta}\subseteq\nu X.pX^M(\bar q^{\theta}).$$

This completes the proof of (6.3.22).

Taking $p=p_0$ in (6.3.22), we get $r_0\in p_0^{\theta}\subseteq p_0^M$, therefore $M,r_0\models p_0$. A finite tree-like model of the appropriate size can be obtained from M by the technique in [8, 13].

Proof of (iii)→(i). This asserts the soundness of the deductive system and is left to the reader. □

Acknowledgment

I would like to thank Rivi Sherman and Joe Halpern for pointing out mistakes in an earlier version [14], and Steve Bloom, Allen Emerson and Joe Halpern for valuable suggestions.

References

[1] J.W. De Bakker, *Mathematical Theory of Program Correctness* (Prentice-Hall, Englewood Cliffs, NJ, 1980).

[2] J.W. De Bakker and W. De Roever, A calculus for recursive program schemes, *Proc. 1st Internat. Coll. on Automata, Languages and Programming* (North-Holland, Amsterdam, 1972) pp. 167–196.

[3] A. Chandra, D. Kozen and L. Stockmeyer, Alternation, *J. Assoc. Comput. Mach.* **28** (1) (1981) 114–133.

[4] E.A. Emerson and E.M. Clarke, Characterizing correctness properties of parallel programs using fixpoints, *Proc. 7th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science **85** (Springer, Berlin, 1980) pp. 169–181.

[5] E.A. Emerson and E.M. Clarke, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: D. Kozen, ed., *Proc. Workshop on Logics of Programs*, Lecture Notes in Computer Science **131** (Springer, Berlin, 1982) pp. 52–71.

[6] M. Fischer and R. Ladner, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (2) (1979) 194–211.

[7] P. Hitchcock and D.M.R. Park, Induction rules and termination proofs, *Proc. 1st Internat. Colloq. on Automata, Languages and Programming* (North-Holland, Amsterdam, 1973) pp. 225–251.

[8] J. Halpern and J. Reif, The propositional dynamic logic of deterministic, well-structured programs (extended abstract), *Proc. 22nd IEEE Symp. on Foundations of Computer Science* (1981) pp. 322–334.

[9] D. Kozen, A representation theorem for models of *-free PDL, *Proc. 7th Internat. Colloq. on Automata, Languages, and Programming*, Lecture Notes in Computer Science **85** (Springer, Berlin, 1980), pp. 352–362.

[10] D. Kozen, On the duality of dynamic algebras and Kripke models, in: E. Engeler, ed., *Proc. Workshop on Logic of Programs*, Lecture Notes in Computer Science **125** (Springer, Berlin, 1979) pp. 1–11.

[11] D. Kozen, On induction vs. *-continuity, in: D. Kozen, ed., *Proc. Workshop on Logics of Programs 1981*, Lecture Notes in Computer Science **131** (Springer, Berlin, 1982) pp. 167–176.

[12] D. Kozen, On the expressiveness of μ, Unpublished manuscript.
[13] D. Kozen, Small models for the propositional μ-calculus, Unpublished manuscript.
[14] D. Kozen, Results on the propositional μ-calculus, *Proc. 9th Internat. Colloq. on Automata, Languages, and Programming* (1982) pp. 348–359.
[15] D. Kozen and R. Parikh, An elementary proof of the completeness of PDL, *Theoret. Comput. Sci.* 14 (1981) 113–118.
[16] D. Kozen and R. Parikh, A decision procedure for the propositional μ-calculus, in: E.M. Clarke and D. Kozen, eds., *Proc. Workshop on Logics of Programs 1983*, Lecture Notes in Computer Science (Springer, Berlin, 1983).
[17] D.M.R. Park, Fixpoint induction and proof of program semantics, in: B. Meltzer and D. Michie, eds., *Mach. Int.* 5 (Edinburgh Univ. Press, 1970) pp. 59–78.
[18] V.R. Pratt, A near optimal method for reasoning about action, *J. Comput. Systems Sci.* 20 (1980) 231–254.
[19] V.R. Pratt, A decidable μ-calculus (Preliminary Rept.), *Proc. 22nd IEEE Symp. on Foundations of Computer Science* (1981) pp. 421–427.
[20] W.P. De Roever, Recursive program schemes: Semantics and proof theory, Ph.D. Thesis, Free University, Amsterdam, 1974.
[21] R. Streett, Propositional dynamic logic of looping and converse, *Proc. 13th ACM Symp. on Theory of Computing* (1981) pp. 375–383.
[22] D. Scott and J.W. De Bakker, A theory of programs, Unpublished manuscript, IBM, Vienna, 1969.