A Decidable Mu-Calculus: Preliminary Report

V.R. Pratt
M.I.T. *and* Stanford

This research was supported by NSF grant no. MCS-7804338.

Abstract

We describe a mu-calculus which amounts to modal logic plus a minimization operator, and show that its satisfiability problem is decidable in exponential time. This result subsumes corresponding results for propositional dynamic logic with test and converse, thus supplying a better setting for those results. It also encompasses similar results for a logic of flowgraphs. This work provides an intimate link between PDL as defined by the Segerberg axioms and the mu-calculi of de Bakker and Park.

Motivation

The notion of minimization is found both in recursive function theory and program semantics, supplying them with an attractive algebraic alternative to iteration and recursion respectively. In the former it is used to specify the least root of a function, that is, the least argument for which the function vanishes. In the latter it is used to specify the least fixpoint of a function defined on a lattice, formalizing the notion of a recursively defined function.

Here we develop a logic of iterative programs in which minimization, of the least-root rather than least-fixpoint type, plays a central role. The language of the logic will be more expressive than that of propositional dynamic logic (PDL), and more succinct by up to an exponential even for those constructs expressible in PDL, yet will have a satisfiability problem that is no more complex than that of PDL. Moreover it will have fewer constructs than PDL: specifically, all five program constructs of PDL (test, union, sequence, star, and converse) are eliminated, with their place being taken by a single construct, minimization. Except for this change, the syntax and semantics of the mu-calculus we describe here is identical to that of propositional dynamic logic as defined by the Segerberg axioms [17], whose mathematical content is studied in [15].

The use of roots rather than fixpoints is so that converse can be treated along with the other constructs. There does not seem to be a way of capturing converse using least fixpoints. We accomplish this absorption of converse only at the price of complicating a step of the algorithm; using fixpoints and keeping converse as an explicit operation is more easily treated.

Our treatment will be algebraic, following [15] more closely than [7] since the notion of continuity central to the latter is absent from the former and the present paper. The advantage of an algebraic treatment is that the results apply equally whether programs are considered to be binary relations on states (Kripke models [13]), languages (execution sequence models [8]), sets of state sequences (trajectory models [6]), sets of sequences of state transitions (Abrahamson semantics for parallel programs [1]), Boolean functions (predicate transformers [3]), or anything else satisfying the Segerberg axioms. This point is made in more technical detail in [14].

These applications subsume the objectives of such program logics as Pnueli's temporal logic (TL) [12], whose primary application is to reasoning about ongoing processes, in contrast to the non-process-oriented Kripke structure model. Propositional dynamic logic as defined by the Segerberg axioms is independent of whether Kripke semantics is intended, and is at least as useful for reasoning about parallel processes as temporal logic, in fact more so since it has the capability of referring to several programs at once, which simplifies reasoning about programs that are structured algebraically. Yet despite its greater expressive power, the mu-calculus of this paper, in reducing five program constructs to a single construct, supplies a language that has no more constructs than TL. In place of the TL primitives X (single-step) and F (essentially X*), our mu-calculus version of PDL has the primitives $\Diamond$ and μ.

Our original motivation for developing the present logic came from a problem to do with flowcharts: what is the complexity of satisfiability for the flowchart version of PDL? This problem was felt by some PDL workers to be inherently of double exponential complexity. We have recently shown [16] that its complexity is a single exponential; the complexity result of this paper is a further strengthening of this result in that our mu-calculus is at least as expressive and succinct as flowchart logic yet has a one-exponential decision method.

An *a posteriori* reason for our interest in this mu-calculus is the link it makes between the work of de Bakker and de Roever [2] and Park [10,11] on mu-calculi, and the work on dynamic logic. The link could not be made so intimately without the results of [15] characterizing the exact strength of the Segerberg axioms in terms of minimality. No attempt at determining the complexity of mu-calculi has been made hitherto, so this paper can be viewed as the transfer of techniques developed within one framework to another more general framework. One could view the role of PDL in this transfer as a mental stepping stone to a result that might have been difficult to even conjecture at *ab initio*.

Another interesting aspect of our mu-calculus is that the finite-model or filtration theorem, that if a term has a model it has a finite model, generalizes from PDL to the mu-calculus. Yet despite this being a more general result, the setting for its proof turns out to be more natural than for PDL, leading to a substantial simplification of the proof, including the elimination of Fischer-Ladner closure.

Syntax

The language consists of Boolean terms, that is, terms intended to denote elements of a Boolean algebra. A term is either a *Boolean variable* (one of P,Q,R,...), a *Boolean combination* of terms (either a *negation* ~p or a *disjunction* $p \vee q$), the *application* Ap of a function symbol A to a term p (corresponding to the PDL diamond construct <A>p where A is atomic), or the *minimization* $\mu\mathbf{Q}.\tau(\mathbf{Q})$ where $\mathbf{Q} = (Q_1,...,Q_n)$ and $\tau(\mathbf{Q})$, the *minuend* (so to speak), is a Boolean term possibly containing free occurrences of the Boolean variables $Q_1,...,Q_n$. We impose on a minuend the two conditions, defined below, that $\tau(1) \rightarrow 0$ (expressing that $\tau(\mathbf{Q})$ simplifies syntactically to 0), and that like-signed occurrences of the Q_i's of a given minimization not combine conjunctively in its minuend. (The notions of free and bound occurrences are as for quantified variables in predicate calculus; thus $\mu\mathbf{Q}.\tau(\mathbf{Q})$ contains no free occurrences of any Q_i, those Q_i's occurring free in $\tau(\mathbf{Q})$ all being bound in $\mu\mathbf{Q}.\tau(\mathbf{Q})$ by the $\mu\mathbf{Q}$.)

We write $\tau(\mathbf{q}) \rightarrow 0$, where $\mathbf{q}$ is an n-tuple of terms, to indicate that $\tau(q)$ simplifies to 0 under repeated application of the following simplification rules.

$$
\begin{aligned}
&{\sim}1,\ p\wedge 0,\ 0\wedge p,\ 0\vee 0,\ A0 \ \rightarrow\ 0\\
&p\vee 0,\ 0\vee p,\ p\wedge 1,\ 1\wedge p \ \rightarrow\ p\\
&{\sim}0,\ p\vee 1,\ 1\vee p,\ 1\wedge 1 \ \rightarrow\ 1\\
&\mu R.\tau'(R) \ \rightarrow\ 0 \quad \text{when } \tau'(0) \ \rightarrow\ 0
\end{aligned}
$$

We require $\tau(1) \rightarrow 0$ to ensure that the set of roots of $\tau(Q)$ is nonempty. This condition is straightforward to test, at least in comparison to testing that $\tau(1)$ is semantically equivalent to 0.

Before defining conjunctive combination it will be convenient to adopt a normal form in which negations are "pushed down" to the variables. This is done by rewriting ${\sim}{\sim}p$ as p, ${\sim}(p\vee q)$ as ${\sim}p\wedge{\sim}q$, ${\sim}Ap$ as $[A]{\sim}p$, and ${\sim}\mu R.\tau(R)$ as $\nu R.{\sim}\tau(R)$, these last three constructs being called respectively *conjunction*, *box*, and *maximization*. A term obtained by rewriting all possible subterms according to these rules will be said to be in *monotonic normal form*. We assume this normal form henceforth. It may be shown that all six operators, disjunction, conjunction, diamond, box, minimization, and maximization, are monotonic.

Two terms are said to *combine conjunctively* when they occur within distinct arguments of a conjunction. A term is considered to combine conjunctively with itself when it appears in the argument of either a box or a maximization. Examples of conjunctive combinations of likesigned variables are $Q_1 \wedge Q_2$, $\sim Q_1 \wedge \sim Q_2$, $[A]\sim Q$, and $\nu Q.Q$, but not $Q_1 \wedge \sim Q_2$, $\sim Q_1 \vee \sim Q_2$, $A \sim Q$, or $\mu Q.Q$. It follows that box and maximization terms occuring within a minuend $\tau(Q)$ must be independent of Q. This conjunctive combining condition is also easily checked, and as will be seen ensures that the set of roots of $\tau(Q)$ forms a lattice (is closed under conjunction and disjunction).

These two restrictions on minimizations are the appropriate analogues for a least-root calculus of the restrictions of syntactic monotonicity and syntactic continuity in Park's least-fixpoint calculus [10,11]. Minimization and the nonmonotonic nature of Boolean algebras do not mix well without such syntactic assistance.

We let B_0 denote the set of Boolean variables, F the set of function symbols, and $B(B_0,F)$, or just B, the set of all Boolean terms over these variables and function symbols.

We abbreviate $p \wedge \sim q$ to p-q and p-p to 0.

Semantics

A *model* is a triple $(\mathbf{B},\mathbf{F},\mathbf{h})$ where $\mathbf{B}$ is a Boolean algebra, $\mathbf{F}$ is a subset of the *strict* $(A0 = 0)$ *finitely additive* $(A(p \vee q) = Ap \vee Aq)$ functions on B, and $\mathbf{h}: B \cup F \rightarrow \mathbf{B} \cup \mathbf{F}$ is a semantic function (hence of homomorphic character) mapping terms to elements of $\mathbf{B}$ and function symbols to elements of F. $\mathbf{h}$ interprets Boolean variables arbitrarily, logical connectives in the standard way, and Ap as the result of applying h(A) to h(p). The value of $\mathbf{h}(\mu Q.\tau(Q))$ is the first coordinate, i.e. q_1, of the least $\mathbf{q}$ in $\mathbf{B}^n$ for which $h_{Q \leftarrow q}(\tau(Q)) = 0$, where $h_{Q \leftarrow q}: B \cup F \rightarrow \mathbf{B} \cup \mathbf{F}$ satisfies $h_{Q \leftarrow q}(Q) = \mathbf{q}$ and otherwise agrees with $\mathbf{h}$. (The first coordinate of $\mathbf{q}$ plays a role analogous to that of the start state of a finite state automaton, as will be seen later.)

We say that the term p *has a model* or is *satisfiable* when for some model $(\mathbf{B},\mathbf{F},\mathbf{h})$, $\mathbf{h}(p) \neq 0$. Such a model is called a *model of* p.

Mu-Algebras

One often needs to determine the whole of a model inductively starting from the values of the variables. Minimization presents difficulties here. It is indeed true that there is at most one model having a given $\mathbf{B}$ and given values for $\mathbf{h}$ at B_0 and F. However such a model need not exist, since the least root of $\tau(Q)$ need not exist (though since $\tau(\mathbf{1})$ vanishes syntactically there must exist roots). A *mu-algebra* is a pair $(\mathbf{B},\mathbf{F})$ such that any mapping of B_0 into $\mathbf{B}$ and F into $\mathbf{F}$ extends to an $\mathbf{h}$ such that $(\mathbf{B},\mathbf{F},\mathbf{h})$ is a model.

Theorem 1. Any finite $(\mathbf{B},\mathbf{F})$ is a mu-algebra.

Proof. It suffices to show that least roots always exist when $\mathbf{B}$ is finite. Theorem 6, proved in the Appendix, states that the set of roots of $\tau(Q)$ forms a lattice. A finite lattice always has a least element, the desired least root. ∎

Examples

For examples of constructs not involving μ we refer the reader to the PDL literature, e.g. [4]. Such terms as $\mu Q.Q \vee {\sim} Q$ are not well formed since they violate the condition $\tau(\mathbf{1}) \rightarrow 0$. Such terms as $\mu Q.Q \wedge AQ$ are not well formed since they contain conjunctive combinations. The following identities hold.

$$\mu Q.0 = 0$$
$$\mu Q.Q = 0$$
$$\mu Q.{\sim}Q = 1$$
$$\mu R.(\tau(R,p) \vee \tau(R,q)) = \mu R.\tau(R,p) \vee \mu R.\tau(R,q)$$

$$\begin{gathered}
\text{provided no } R_i \text{ occurs in p or q} \\
\mu Q.(p \vee \mu R((Q \vee AR)\text{-}R))\text{-}Q = \mu Q.(p \vee AQ)\text{-}Q \\
\mu Q.p \wedge (\mu R.{\sim}Q \wedge A{\sim}R) = Ap
\end{gathered}$$

The last two formulas are the mu-calculus equivalents of the (shorter in this case) PDL identities $\langle A^{**}\rangle p = \langle A^{*}\rangle p$ and $\langle A^{--}\rangle p = \langle A\rangle p$.

Correspondence with PDL

There is a direct translation from PDL into the mu-calculus. We denote by p' the mu-calculus translation of the PDL formula p.

$$\begin{aligned}
P', Q', \ldots &= P, Q, \ldots \\
({\sim}p)', (p \vee q)', \ldots &= {\sim}p',\ p' \vee q',\ \ldots \\
(\langle A\rangle p)' &= Ap' \\
(\langle a \cup b\rangle p)' &= (\langle a\rangle p)' \vee (\langle b\rangle p)' \\
(\langle a;b\rangle p)' &= (\langle a\rangle\langle b\rangle p)' \\
(\langle p?\rangle q)' &= p' \wedge q' \\
(\langle a^{*}\rangle p)' &= \mu Q.((p' \vee (\langle a\rangle Q)')\text{-}Q) \\
&\quad \text{where Q does not occur free in a or p} \\
(\langle a^{-}\rangle p)' &= \mu Q.(p' \wedge (\langle a\rangle {\sim}Q)') \\
&\quad \text{where Q does not occur free in a or p}
\end{aligned}$$

Note the doubling of p in the translation of $\langle a \cup b\rangle p$. This compounds to yield exponential growth: $\langle a \cup b\rangle^{n} p$

would translate to 2^n disjunctions. An alternative translation of $\langle a \cup b \rangle p$ is $\mu QR.((\langle a \rangle R)'\text{-}Q) \vee ((\langle b \rangle R)'\text{-}Q) \vee (p'\text{-}R)$. This translation has the merit of not doubling p and so replacing exponential growth by constant factor growth. It also supplies an example of a multivariate minimization. The translation is less obscure in the light of the discussion on flowcharts below.

In [15] in effect we showed that the Segerberg axioms exactly define the values of the PDL formulas on the left to be equal to that of the mu-calculus terms on the right, except for $\langle a^{-} \rangle p$ which at the time we could not characterize. However the Parikh axioms $p \rightarrow [a]\langle a^{-} \rangle p$ and $p \rightarrow [a^{-}]\langle a \rangle p$ [9] turn out to define the value of $\langle a^{-} \rangle p$ to be exactly that of the value shown on the right. (To see this, observe that the first axiom says that $\langle a^{-} \rangle p$ is a root of $p \wedge \langle a \rangle {\sim} Q$, while the second says that $\langle a^{-} \rangle p$ is less than any other root Q since $\langle a^{-} \rangle p \leq \langle a^{-} \rangle [a] Q \leq Q$.) It follows that these translations preserve the Segerberg semantics of the PDL formulas shown on the left.

A corollary of this translation is:

Theorem 2. Every mu-algebra expands to a dynamic algebra.

(The expansion is needed to supply F with the regular operations ∪, ;, *, ?, and ˉ. The content of the theorem is that these operations are all definable in a mu-algebra. They are not in general definable for arbitrary infinite (B,F).)

We raise the converse of Theorem 2 as an open problem.

This semantic correspondence between Segerberg PDL and the mu-calculus is particularly interesting in the light of the question as to the right definition of the class of dynamic algebras. The question is whether to use Kozen's [7] or the author's [12] definition of the class of dynamic algebras. Kozen's class imposes a continuity requirement. In [12] we argued at length for the merits of avoiding continuity, on the ground that much of the fundamental theory of dynamic algebras could be developed without it. The connection between the Segerberg semantics and mu-calculi, independent of continuity, adds further support to our position.

Flowgraph Logic

Many regular sets can be represented exponentially more compactly with finite state automata than regular expressions [3a]. This casts doubts on the succinctness of PDL, and leads one to explore notations in which large regular expressions in PDL formulas are replaced by small finite state automata, using some notation convenient for representing graphs. A logic based on such a notation was presented in [16]. We sketch here how our mu-calculus can serve this function, with the same level of succinctness as provided by graphs.

Let $\langle a\rangle p$ be a PDL formula in which a is a large regular expression with a small equivalent nondeterministic finite state automaton having s states (state 1 being the start state), e edges, and n final states. The approach is to translate $\langle a\rangle p$ into the mu-calculus term $\mu Q_1 Q_2 ... Q_s . E_1 \vee ... \vee E_e \vee F_1 \vee ... \vee F_n$ where the variable Q_i corresponds to the ith state, the disjunct E_i to the ith edge, and the disjunct F_i to the ith final state, of the automaton. For each edge labelled B from state i to state j, there will be the disjunct $BQ_j\text{-}Q_i$, while for each final state Q_f there will be the disjunct $p\text{-}Q_f$.

The disjuncts corresponding to final states can be collected into the form $p\text{-}(Q_{f_1} \wedge ... \wedge Q_{f_n})$. This avoids duplicating p in the translation.

Consider for example an automaton with state set {1,2,3}, a ring of three edges $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ labelled respectively A,B,C, a fourth edge $3 \rightarrow 2$ labelled D, with start state 1 and final states 2 and 3. One close-to-minimal regular expression for this is

$E = (A(BD)^*BC)^*A((BD)^* \cup (BD)^*B)$.

The translation of $\langle E \rangle P$ using this automaton is

$\mu Q_1Q_2Q_3.(AQ_2\text{-}Q_1) \vee (BQ_3\text{-}Q_2) \vee (CQ_1\text{-}Q_3) \vee (DQ_2\text{-}Q_3) \vee (P\text{-}(Q_2 \wedge Q_3))$.

The second translation we gave earlier for $\langle a \cup b \rangle p$ was obtained in this way by starting with the obvious finite-state automaton accepting $a \cup b$.

The above technique may be understood by thinking of each Q_i as denoting the language accepted by the automaton if the start state is taken to be state i. The process of converting the finite state automaton to a mu-calculus term is very similar to that of converting the automaton to a Type 3 grammar. Read each edge-related disjunct $AQ_j\text{-}Q_i$ as the production $Q_i \rightarrow AQ_j$ and each final-state-related disjunct $p\text{-}Q_i$ as the production $Q_i \rightarrow p$ (where p is to be thought of as some sort of delimiter). Vanishing of the disjunction is equivalent to vanishing of all the disjuncts, which in turn is equivalent to reading the arrows of the grammar as language inclusions.

It is evident from the discussion in [5] that Hitchcock and Park also had flowgraphs (or flowcharts) in mind as an application for their mu-calculus.

Theorem 3. If p has a model it has a finite model.

Proof. Let (**B**,**F**,**h**) be a model of p. Take **B'** $\subseteq$ **B** to be the Boolean algebra generated by the image under **h** of the subterms of p and **F'** $\subseteq$ **F** to be the set of strict finitely additive functions on **B'**. Take **h'**(P) = **h**(P) for each Boolean variable in the language, and take **h'**(A) to be the function that maps each p in **B'** to $\bigwedge\{q \in \mathbf{B}' | q \geq \mathbf{h}(A)p)\}$. (The "$\geq$" here is that of **B**, and can be used because **B'** $\subseteq$ **B**. This also explains why **h**(A) can be applied to p.)

(**B'**,**F'**,**h'**) is finite because p has only finitely many subterms and a finitely generated Boolean algebra is finite. In fact n elements generate a Boolean algebra of cardinality at most 2^{2^n}, giving us an upper bound on the size of the model in terms of the number of subformulas, and hence length, of p. Note that this bound corresponds to a bound of 2^n on the number of states of a Kripke structure, cf. [4]. **B'** corresponds to predicates on states, of which there may be exponentially many in the number of states.

Now extend **h'** to the rest of *B* and *F* using the standard interpretations for negation, disjunction, application, and minimization. Minimization is always well defined since

B' is finite.

(B',F',h') is a model, by construction. To verify that it is a model of p it suffices to show that h' agrees with h on the subformulas of p, since $h(p) \neq 0$. We proceed inductively.

By construction h' agrees with h on Boolean variables. Since B' is a subalgebra of B, h' agrees with h on all Boolean combinations of terms on which h' agrees with h. $h'(Ap) = h'(A)h'(p) = h'(A)h(p)$
$= \bigwedge\{q \in B' | q \geq h(A)h(p) = h(Ap)\} = h(Ap)$ since $h(Ap) \in B'$. $h(\mu Q.\tau(Q))$ is a root of $\tau(Q)$ in (B',F'.h') since $h(\mu Q.\tau(Q)) \in B'$ and by induction h' agrees with h on τ, whence $h'(\mu Q.\tau(Q)) \leq h(\mu Q.\tau(Q))$. For the other direction we have that for any q in B', $h'_{Q \leftarrow q}(\tau(Q)) \geq h_{Q \leftarrow q}(\tau(Q))$ (argued below), whence any root in B' of $\tau(Q)$ is a root in B of $\tau(Q)$ and so $h(\mu Q.\tau(Q)) \leq h'(\mu Q.\tau(Q))$. Hence h' agrees with h on the subformulas of p.

To verify that $h'_{Q \leftarrow q}(\tau(Q)) \geq h_{Q \leftarrow q}(\tau(Q))$, note that the first (inductively speaking) parts of $h'_{Q \leftarrow q}(\tau(Q))$ that can disagree with $h_{Q \leftarrow q}(\tau(Q))$ are applications. Negative applications (boxes) cannot contain μ-bound variables, so only positive applications can disagree. The definition of h'(A) then forces these positive applications to be larger in $h'_{Q \leftarrow q}(\tau(Q))$ than in $h_{Q \leftarrow q}(\tau(Q))$, whence the former must be larger than the latter. ∎

Testing Satisfiability

The problem is to test whether a mu-calculus term is satisfiable (is nonzero in some model). The method we present here is derived from the method of [14] for PDL.

Our algorithm constructs a single model (**B**,**F**,**h**) and tests whether the input has value 0 in this model. **B** is a quotient of the free Boolean algebra freely generated by the Boolean variables and applications of the input, **F** is a subset of the strict finitely additive functions on **B**, and **h** maps variables and applications to their image under the quotient. The major part of the algorithm is the calculation of the quotient, whose purpose is to ensure that $\mathbf{h}(Ap) = \mathbf{h}(A)\mathbf{h}(p)$.

The steps of the algorithm are as follows.

1. Construct the atoms of the free Boolean algebra **B** freely generated by the Boolean variables and applications occurring in the input.

2. Construct **h** fixing Boolean variables and applications and mapping each function symbol A to the greatest function on **B** satisfying $\mathbf{h}(A)\mathbf{h}(p) \leq \mathbf{h}(Ap)$ for every

application Ap occurring in the input. (It suffices to represent a strict finitely additive function in terms of its action on atoms of **B**.) Take **F** to be $\mathbf{h}(F)$ where F is the set of function symbols appearing in the input. To within the quotient construction of step 3, **B**, **F**, and **h** are now all determined.

3. For each application Ap in the input for which $\mathbf{h}(A)\mathbf{h}(p) < \mathbf{h}(Ap)$ (a *defect*), subtract $\mathbf{h}(Ap) - \mathbf{h}(A)\mathbf{h}(p)$ from all elements of **B** and correspondingly pointwise from elements of **F** and from values of **h**. (This is the quotient operation.) Repeat this operation until no further defects remain.

4. The input is 0 in all models just if it is zero in the resulting model.

For step 3 it will be necessary to be able to calculate **h** on Boolean combinations and minimizations. The operations of **B** are available for the former. A simple method for computing $\mu Q.\tau(Q)$ is to form the intersection of all roots of $\tau(Q)$. However there may be up to $|\mathbf{B}|$ such roots, so even if the roots could be found easily this method would require doubly exponential time. We need to be able to find the least root without touching all the other roots.

When all negative μ-variables have no application between them and their binding μ, $\mu Q.\tau(Q)$ can be computed as $\tau^*(0) = \cup_{i<\omega}\tau^i(0)$. This computation must converge since **B** is finite. Moreover the longest chain in **B** is of length 2^n, so ω may be reduced to 2^n.

When negative μ-variables appear below applications and/or minimizations our treatment is considerably more complex; we defer its account to a longer version of this paper.

The existence of this algorithm gives:

Theorem 4. The mu-calculus is decidable in deterministic exponential time.

To within a polynomial, no stronger result is possible due to the reduction of PDL to the mu-calculus, and the deterministic exponential lower bound for PDL satisfiability proved in [4]. (We count symbols rather than bits in our estimates, converting the $d^{n/\log n}$ lower bound of [4] to a d^n lower bound.)

Open Problems

1. Is the equational theory of this mu-calculus finitely axiomatizable? That is, is there a finite subset of the equational theory from which the rest of the theory follows

by equational logic?

2. Is every dynamic algebra with test and converse a mu-algebra? (We have already shown the converse. Moreover we have answered this positively in the finite case, since every finite Boolean algebra and nonempty set of strict finitely additive functions is both a dynamic algebra and a mu-algebra.)

3. Can converse be represented if least fixpoints are used in place of least roots?

Appendix: A Fundamental Property of Roots

We show in this appendix that the set of roots of a minuend $\tau(Q)$ forms a lattice, that is, is closed under conjunction and disjunction. Closure under conjunction is used to establish existence of least roots in finite mu-algebras (Theorem 1). Closure under disjunction is used in the proof (omitted from this paper) of a lemma, cited as "lemma *" below, that in effect says that any operation on programs implicitly definable in the mu-calculus has for its range finitely additive functions. (For example, since * and $^-$ are implicitly definable, $a^*(p \vee q) = a^*p \vee a^*q$ and $a^-(p \vee q) = a^-p \vee a^-q$.)

We approach this theorem by developing properties of the monotonic mu-calculus, defined to have disjunction, diamond, minimization, and their respective duals, but not negation.

Let $\sigma(\mathbf{p},\mathbf{q})$ be a function mapping a pair of n-tuples of terms to a term, all terms being in the monotonic mu-calculus. We say that σ is *barren* when its subterms are barren, and

(i) if it is of the form $\sigma_1(\mathbf{p},\mathbf{q})\wedge\sigma_2(\mathbf{p},\mathbf{q})$ then not both $\sigma_1(\mathbf{p},\mathbf{q})$ and $\sigma_2(\mathbf{p},\mathbf{q})$ may depend on $\mathbf{p}$, nor may both depend on $\mathbf{q}$;

(ii) if it is of one of the forms $\Box\sigma_1(p,q)$ or $\nu Q.\sigma_1(p,q)$ then it must be independent of both p and q.

(The motivation for this property is that $\tau(Q)$ has no conjunctive combinations just when it is of the form $\sigma(Q,\sim Q)$ for some barren σ.)

Lemma 5. If σ is barren then $\sigma(\mathbf{p}\vee\mathbf{q},\mathbf{r}\wedge\mathrm{s}) \leq \sigma(\mathbf{p},\mathbf{r})\vee\sigma(\mathbf{q},\mathrm{s})$ and $\sigma(\mathbf{p}\wedge\mathbf{q},\mathbf{r}\vee\mathrm{s}) \leq \sigma(\mathbf{p},\mathbf{r})\vee\sigma(\mathbf{q},\mathrm{s})$.

Proof. We proceed by induction on the structure of σ. By symmetry of the arguments of σ it suffices to show the first inequality. The cases are:

1. $\sigma(u,v) = u_i$.
Then $\sigma(p\vee q, r\wedge s) = (p\vee q)_i = p_i\vee q_i = \sigma(p,r)\vee\sigma(q,s)$.

2. $\sigma(u,v) = v_i$.
Then $\sigma(p\vee q, r\wedge s) = (r\wedge s)_i = r_i\wedge s_i \leq \sigma(p,r)\vee\sigma(q,s)$.

3. $\sigma(u,v) = w$ independent of u and v. Immediate.

4. $\sigma(u,v) = \sigma_1(u,v) \vee \sigma_2(u,v)$

$$\begin{aligned}
\text{Then } \sigma(p\vee q, r\wedge s) &= \sigma_1(p\vee q, r\wedge s) \vee \sigma_2(p\vee q, r\wedge s)\\
&\leq \sigma_1(p,r)\vee\sigma_1(q,s) \vee \sigma 2(p,r)\vee\sigma 2(q,s)\\
&= \sigma(p,r)\vee\sigma(q,s).
\end{aligned}$$

5. $\sigma(u,v) = \sigma_1(u,v) \wedge \sigma_2(u,v)$

$$\begin{aligned}
\text{Then } \sigma(p\vee q, r\wedge s) &= \sigma_1(p\vee q, r\wedge s) \wedge \sigma_2(p\vee q, r\wedge s)\\
&= \sigma_1(p\vee q, r\wedge s) \wedge \sigma_2(0, r\wedge s) \text{ w.l.o.g.}\\
&\leq (\sigma_1(p,r)\vee\sigma_1(q,s)) \wedge \sigma_2(0, r\wedge s) \quad \text{(ind. hyp.)}\\
&= (\sigma_1(p,r)\wedge\sigma_2(0,r\wedge s)) \vee (\sigma_1(q,s)\wedge\sigma_2(0,r\wedge s))\\
&\leq (\sigma_1(p,r)\wedge\sigma_2(p,r)) \vee (\sigma_1(q,s)\wedge\sigma_2(q,s))\\
&\qquad\qquad \text{(monotonicity of } \sigma)\\
&= \sigma(p,r)\vee\sigma(q,s).
\end{aligned}$$

6. $\sigma(u,v) = \Diamond\sigma_1(u,v)$.

$$\begin{aligned}
\text{Then } \sigma(p\vee q, r\wedge s) &= \Diamond\sigma_1(p\vee q, r\wedge s)\\
&\leq \Diamond(\sigma_1(p,\overset{r}{\cancel{q}})\vee\sigma_1(q,s))\\
&= \Diamond\sigma_1(p,\underset{r}{\cancel{q}}) \vee \Diamond\sigma_1(q,s)
\end{aligned}$$

$$= \sigma(p,\overset{r}{\cancel{q}}) \vee \sigma(\underset{q}{\cancel{r}},s).$$

7. $\sigma(u,v) = \Box\sigma_1(u,v)$.
Then $\sigma(p\vee q,r\wedge s) = \Box\sigma_1(0,0)$
$= \Box\sigma_1(p,\cancel{r}) \vee \Box\sigma_1(\overset{r}{\cancel{q}},s)$ [illegible]
$= \sigma(p,\underset{r}{\cancel{q}}) \vee \sigma(\underset{q}{\cancel{r}},s)$.

8. $\sigma(u,v) = \mu Q.\sigma_1(u,v)$.
Then $\sigma(p\vee q,r\wedge s) = \mu Q.\sigma_1(p\vee q,r\wedge s)$
$\leq \mu Q.(\sigma_1(p,r)\vee\sigma_2(q,s))$ (ind.hyp.)
$\leq \mu Q.\sigma_1(p,r) \vee \mu Q.\sigma_2(q,s)$ (lemma *)
$= \sigma(p,\underset{r}{\cancel{q}}) \vee \sigma(\underset{q}{\cancel{r}},s)$.

9. $\sigma(u,v) = \nu Q.\sigma_1(u,v)$. Same argument as 7. ∎

Theorem 6. The roots of a minimization argument $\tau(Q)$ form a lattice.

Proof. $\tau(Q)$ may be written as $\sigma(Q,\sim Q)$ where σ is barren. Let p and q be two roots of $\tau(Q)$. Hence $\sigma(p,\sim p) = \sigma(q,\sim q) = 0$. But $\sigma(p\vee q,\sim p\wedge\sim q) \leq \sigma(p,\sim p)\vee\sigma(q,\sim q) = 0$ and $\sigma(p\wedge q,\sim p\vee\sim q) \leq \sigma(p,\sim p)\vee\sigma(q,\sim q) = 0$, whence $\tau(p\vee q) = \tau(p\wedge q) = 0$, i.e. $p\vee q$ and $p\wedge q$ are roots of τ. ∎

Bibliography

[1] Abrahamson, K., Modal Logic of Concurrent Nondeterministic Programs, Preprints of the International Symposium on Semantics of Concurrent Computation, Evian-les-Bains, July 1979.

[2] de Bakker, J.W., and W.P. de Roever, A calculus for recursive program schemes, In **Automata, Languages and Programming** (ed. Nivat), 167-196. North Holland, 1973.

[3] Dijkstra, E.W., **A Discipline of Programming.** Prentice-Hall. 1976

[3a] Ehrenfeucht, A., and P. Zeiger, Complexity Measures for Regular Expressions, JCSS, **12.**, 2, 134-146, April, 1976.

[4] Fischer, M.J. and R.E. Ladner., Propositional Dynamic Logic of Regular Programs, JCSS, **18**, 2, 194-211, April 1979.

[5] Hitchcock, P. and D. Park., Induction Rules and Termination Proofs, In **Automata, Languages and Programming** (ed. Nivat, M.), 225-252,. North-Holland, 1973.

[6] Hoare, C.A.R. and P.E. Lauer, Consistent and Complementary Formal Theories of the Semantics of Programming Languages, Acta Informatica 3, 135-153, 1974.

[7] Kozen, D., On Models of Dynamic Logic, Research Report, January 1980. To appear in J. Symbolic Logic.

[8] Ogden, W.F., W.E. Riddle, and W.C. Rounds, Complexity of Expressions Allowing Concurrency, Proc. 5th ACM Symposium on Principles of Programming Languages, 185-194, Tucson, Arizona, Jan. 1978.

[9] Parikh, R., A Completeness Result for a Propositional Dynamic Logic, Lecture Notes in Computer Science No. 64, 403-415, Springer-Verlag, 1978.

[10] Park, D., Fixpoint Induction and Proofs of Program Properties, In **Machine Intelligence 5.** Edinburgh University Press. 1969.

[11] Park, D., Finiteness is mu-ineffable, Theoretical Computer Science, 176-181, Nov., 1976.

[12] Pnueli, A., The Temporal Logic of Programs, 18th IEEE Symposium on Foundations of Computer Science, 46-57. Oct. 1977.

[13] Pratt, V.R., Semantical Considerations on Floyd-Hoare Logic, Proc. 17th Ann. IEEE Symp. on Foundations of Comp. Sci., 109-121. Oct. 1976.

[14] Pratt, V.R., Models of Program Logics, Proc. 20th IEEE Conference on Foundations of Computer Science, San Juan, PR, Oct. 1979.

[15] Pratt, V.R., Dynamic Algebras and the Nature of Induction, Proc. 12th ACM Symp. on Theory of Computing, 22-28, Los Angeles, CA, May, 1980.

[16] Pratt, V.R., Complexity of Flowgraph Logic, manuscript, Dec., 1980.

[17] Segerberg, K., A Completeness Theorem in the Modal Logic of Programs, Preliminary report. Notices of the AMS, 24, 6, A-552. Oct. 1977.