MIT/LCS/TM-191

PROPOSITIONAL DYNAMIC LOGICS OF PROGRAMS: A SURVEY

Rohit Parikh

January 1981

PROPOSITIONAL DYNAMIC LOGICS OF PROGRAMS: A SURVEY

Rohit Parikh

Math Dept, Boston Univ. and Laboratory for Computer Science, MIT

§1. Introduction

The use of logic in program verification is an old idea, as such ideas go. Early work by Engeler, Floyd, Hoare and Salwicki [E,F,Ho,Sa2] has already developed into a rich field with many workers. However the *propositional* versions of these logics are relatively new; work in this field goes back only to Fischer and Ladner's 1977 paper where they showed that the propositional version of Pratt's dynamic logic is decidable. Since then the field has developed rapidly, a very large number of preliminary questions have already been settled, and interesting side areas are developing. In what follows, we shall try to give an overview, stating some of the main results and referring the reader to the original sources for the more difficult or elaborate proofs.

Propositional logics of programs bear the same relationship to ordinary (first order) logics of programs that the propositional calculus has to first order logic. In particular, the various propositional logics of programs are decidable. Moreover first order program logics tend to be highly undecidable (usually Π^1_1) so they cannot be axiomatised recursively; an axiomatisable logic must be recursively enumerable. By contrast, the propositional logics do have nice axiomatisations [Ga,KP,Pa,Se,V,Hal,HKP] that cast some light on the logical nature of the various program constructs and the way they interact with the usual Boolean operations.

In Propositional Dynamic Logic (PDL), programs are treated as modalities that operate on properties to produce other properties. Since syntactically, a property is expressed as a logical formula, a program expression becomes a modal operator on the syntactic category of formulae. Models for such logics are, naturally enough, generalisations of Kripke's

1. Research supported in part by NSF grant MCS 79 10261.

models for modal logic. Thus in dynamic logic the basic model of computation is the state space. A program may be thought of as a means of travelling about in this state space, and a property is something that is true or false of an individual state. Thus a property will be realised mathematically as a *set of states*, those states where the property holds, whereas a program α is a *set of pairs of states* (s,t) where s is the *initial state* of some computation of α and t is the *final state.*

Once we have the notion of model, we can easily define the notions of satisfiability and validity. A formula is *satisfiable* if it holds at some state in some model. It is *valid* if it holds at every state in every model. Many of the logics that we shall consider will have the finite model property. That is, if a formula A has a model at all, then it has a finite model whose maximum size can be estimated by looking at the formula. This immediately gives us a way of deciding satisfiability for a given formula of these logics. Since a formula A is valid (true under all interpretations) iff $\neg A$ is not satisfiable, we also get decidability of the validity problem.

In PDL both the basic programs and the program constructs are allowed to be nondeterministic. The basic programs may be nondeterministic in the sense that an initial state s does not need to determine the final state. Moreover, two of the program constructs, $\cup$ and $*$ are also nondeterministic and convert deterministic relations into nondeterministic ones. We can get special versions of PDL by imposing determinacy conditions. If we require the basic programs to be deterministic, i.e. to be *partial functions* on the state space, then we get the logic DPDL. If the program constructs are *also* required to be deterministic, then we get strict DPDL, i.e. SDPDL. In SDPDL, the nondeterministic constructs $\cup$ and $*$ of PDL and DPDL are dropped and we replace them with the deterministic constructs "if...then...else..." and "while...do...". It is clear that SDPDL is a subsystem of DPDL, but DPDL is *not* a subsystem of PDL. Rather the other way around. Just as the theory of dense linear orders is an *extension* of the theory of all linear orders, so the theory of deterministic programs is an *extension* of the theory of all programs. In fact DPDL proved to be more intricate than PDL and the basic questions have been solved only recently by Ben-Ari, Halpern and Pnueli [BP], [Hal].

In the Kripke models for PDL a program is represented as a binary relation on the space of states. This representation is adequate for stating the *correctness* conditions of a program. Thus for instance the formula $A \rightarrow [\alpha]B$, where A and B are formulae and α is a program, says that if the precondition A holds before the program α begins, then the postcondition B will hold if and when α terminates. When α is represented as the binary relation R_α, then $A \rightarrow [\alpha]B$ holds in a model M iff for all pairs of states (s,t) such that $(s,t) \in R_\alpha$, if s satisfies A then t satisfies B. However, we may also want to state conditions that have to hold *as* the program is running and the notation and semantics for dynamic logic do not directly allow us to do this. The binary relation R_α contains information only about the *initial* and *final* state and is a little like a commuter who drives from his house to his place of work without noticing what he sees on the way. Thus if we do want to take the *intermediate* states of the program into account, then a program had better be represented not as a set of pairs, but rather as a set of sequences of states, the sequences being the trajectories of the program as it runs. This sort of extension of the language and models to include intermediate states of a program leads us from dynamic logic into process logic, an area which seems to have clear potential for applications in parallel programming.

In dealing with process logic, it is most useful to see it as an elaboration of dynamic logic. In the [HKP] version of process logic, this fact is explicitly made use of, the Segerberg axioms for PDL [Seg] are retained and supplemented by axioms that talk about intermediate states, and we get a very smooth completeness theorem. Alas, the decision procedure is not (Kalmar) elementary, but perhaps nicer bounds can be found for some of the simpler formulae that are more like those that actually arise in applications.

Another nice offshoot of PDL is dynamic algebra. There seem to be two principal versions of dynamic algebra which differ slightly, and which are due respectively to Kozen and Pratt. Pratt [Pr4] uses his algebras to give an algebraic proof of the completeness theorem for PDL. Kozen [K1] gives a representation theorem which relates concrete Kripke models with his abstract dynamic algebras, thereby generalising the Stone representation theorem for Boolean algebras. The theorem shows in essence that there is a duality between "non-standard" Kripke models and certain topological spaces that extends the more familiar duality between Boolean algebras and totally disconnected topological spaces.

Any survey necessarily reflects the point of view of its author. We have given more space to results that we knew better, or which seemed more interesting to us. However, the last section, brief notes, contains some pointers to items not adequately covered here. We apologise for any distortions or omissions of other people's work.

§2. The Language of PDL

The *basic* notion in dynamic logic is that of a state. Other notions of dynamic logic are derived from it. Thus we think of a program as *taking* us from a state of our computer to another state of the same computer. Similarly, a *property* is something which a state *has* or *fails to have.* What we study is the interaction between properties and programs in this state context. Properties are expressed by formulae of our logic, and they will be closed under the boolean operations $\neg$, $\vee$, $\wedge$. Programs in propositional logics will be constructed from some basic programs by means of regular flowcharts, syntactically realised by means of the Kleene operations $\cup$, ; , and $*$. The informal meanings of these constructs are as follows: $\alpha;\beta$ is the program "do α, then do β". $\alpha\cup\beta$ is the program, "non-deterministically do α or β". And finally, α^* is the program, "do α, zero or more times". Then in addition, if A is a formula, then A? will be a program which equals "if A then skip, else abort". Where "skip" is the identity program, and "abort" is the null program. In terms of this notion of ?, we can define "if A then do α else do β" as $(A?;\alpha)\cup(\neg A?;\beta)$. Similarly, "while A do α" can be expressed as $(A?;\alpha)^*;(\neg A?)$. The precise mathematical definitions of the semantics of ; , $\cup$ and $*$ are given in definition 2.4. Unless otherwise stated, we shall confine ourselves to *regular* constructions of programs.

The *special* feature of dynamic logics is the *box operator* which equals, roughly, Dijkstra's weakest liberal precondition. The box operator operates on formulae A and produces new formulae $[\alpha]A$, where the program α is also variable. Since we allow non-deterministic programs, there is a choice. We take "the state s satisfies the formula $[\alpha]A$" to mean: *every* terminating computation of α, which begins at the state s, results in condition A holding. The dual construct $\langle\alpha\rangle A$, which can also be defined as $\neg[\alpha]\neg A$, means: *some* computation of α which begins at the state s, terminates with condition A holding.

Now we give *precise* definitions of the basic notions of PDL. A reader familiar with PDL may skip or skim over these but we recommend some attention to the note following definition 2.2 and to the comments at the end of definition 2.4.

Definition 2.1: The symbols of PDL (also DPDL) include atomic program symbols $a_1,...,a_n$, atomic propositional symbols $P_1,...,P_m$, and $\cup$, ; , * , ? , $\wedge$, $\vee$, $\neg$, [,] , < , > , (,).

$$\Sigma = \{a_1,...,a_n\}.$$

Definition 2.2: We define *program expressions* α and *formulae* A by simultaneous recursion. (Program expressions will be called just programs)

(i) Each a_i is a program. Each P_j is a formula.
(ii) If A, B are formulae, so are $(A \vee B)$, $(A \wedge B)$, $(\neg A)$.
(iii) If α, β are programs, so are $(\alpha;\beta)$, $(\alpha \cup \beta)$,(α^*)
(iv) If A is a formula and α is a program, then $[\alpha]A$ and $\langle\alpha\rangle A$ are formulae.
(v) If A is a formula then (A?) is a program.

Note: The programs created by condition (v) above are called *tests.* We shall use the convention that $\rightarrow$ stands for the truth-functional connective, and $\Rightarrow$ for the English "implies". The symbol - -> will be used for "is mapped into". Letters α, β etc. will stand for programs and A, B, etc. for formulae.

Definition 2.3: A model M for PDL consists of

(1) a nonempty universe W
(2) for each atomic propositional symbol P_j a subset $\rho(P_j)$ of W, and
(3) for each a_i in Σ a subset R_i of $W \times W$.

Now we define the semantics of PDL. The semantics assigns to each program expression α, a *binary relation* R_α on W, and to each formula A, and state $s \in W$, a *truth value.* Intuitively, $(s,t) \in R_\alpha$ should be read, "the program α can take us from the state s to the state t". If α is deterministic, then t is *uniquely* determined by s. $M,s \models A$ is read, "the formula A *holds* at the state s of the model M".

Definition 2.4: Given a model M, a state s, a formula A and a program α, we define R_α and $M,s \models A$, by simultaneous recursion, as follows: .

(i) $M,s \models P_j$ iff $s \in \rho(P_j)$.
(ii) $M,s \models A \vee B$ iff $M,s \models A$ or $M,s \models B$.
(similarly $\wedge$ and $\neg$)
(iii) $M,s \models [\alpha]A$ iff for all t, $(s,t) \in R_\alpha \Rightarrow M,t \models A$
$M,s \models \langle\alpha\rangle A$ iff for some t, $(s,t) \in R_\alpha$ and $M,t \models A$.

(iv) If $\alpha = a_i$ then $R_\alpha = R_i$.
(v) $R_{\alpha;\beta} = (R_\alpha)\circ(R_\beta)$
$= \{(s,t) \mid (\exists u)((s,u) \in R_\alpha \wedge (u,t) \in R_\beta)\}$.
(vi) $R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$,
(vii) R_{α^*} = reflexive transitive closure of R_α.
(viii) If $\alpha = (A?)$ then $R_\alpha = \{(s,s) \mid M,s \models A\}$.

Finally, we let, $M \models_g A$ iff for all $s \in W$, $M,s \models A$. (g stands for "global".) If Γ is a set of formulae, then $M,s \models \Gamma$ if for all A in Γ, $M,s \models A$. Similarly $M \models_g \Gamma$. Now we get two notions of consequence.

$\Gamma \models A$ iff for all M,s, $M,s \models \Gamma$ implies $M,s \models A$.

$\Gamma \models_g A$ iff for all M, $M \models_g \Gamma$ implies $M \models_g A$.

The formula A is said to be *valid* iff it holds at all states in all models, iff $\not{c} \models A$ iff $\not{c} \models_g A$.

Local consequence and global consequence coincide if Γ is empty. But if Γ is nonempty, then local consequence is stricter. E.g. we do not have $P \models [\alpha]P$ since running the program α may destroy the fact that P holds. However with global consequence, we do have $P \models_g [\alpha]P$. In §6 we are concerned with global consequence which is the natural one for partial correctness. However, theorem 7.1 applies to both kinds of consequence.

§3. Decision Methods and Complexity

Propositional dynamic logic begins historically with Fischer and Ladner's paper [FL] where they define PDL, show that it is decidable, and establish upper and lower bounds. The decision method consists of the refinement of a technique known among modal logicians as filtration. We shall give a brief account of this technique in our present context where programs are present.

Let A be a formula of PDL. Suppose now that A is satisfiable, that is to say, A has a model. We would like to ask if A has a *finite* model, and if so, how big that model might be. Clearly, if we know how big the model might be, then we can find this model by brute force, i.e., by simply searching through all models of a certain size or less.

The existence of a finite model for a satisfiable formula is obtained in the following way. Given a formula A let FL(A), the Fischer-Ladner closure of A, be the set of all formulae that are *relevant* to the semantics of A. If we were studying propositional logic, the corresponding set would be the set of all the *subformulae* of A. In the present context it is the smallest set S such that

(i) $A \in S$
(ii) If $B \wedge C \in S$ then $B, C \in S$. Similarly for $\vee$, $\neg$.
(iii) If $[\beta;\gamma]B \in S$, then $[\beta][\gamma]B, [\gamma]B \in S$.
(iv) If $[\beta \cup \gamma]B \in S$, then $[\beta]B, [\gamma]B \in S$.
(v) If $[\beta^*]B \in S$, then $B, [\beta][\beta^*]B \in S$.

It can be shown that the size of S is no greater than the length of A, which we will always take to be the number n of symbols in A. Now given a model M of A define an equivalence relation $\equiv$ on W by letting $s \equiv t$ iff for all $B \in S$, $M,s \models B \leftrightarrow M,t \models B$.

Clearly the relation $\equiv$ defined this way has at most 2^n equivalence classes. Let $W/\equiv$ be the set of equivalence classes of W under $\equiv$. Then Fischer and Ladner show how to convert the model M of A into a new model M/S whose state space is $W/\equiv$. We immediately get

Lemma 3.1: If a formula A of PDL is satisfiable, then it has a model of size at most 2^n where n is the length of A.

It is not hard to show that *given* a model N of size k, we can find out in time polynomial in k and the length n of A, if a given formula A is satisfied in N at all. Thus we get the immediate corollary.

Theorem 3.2 (Fischer-Ladner): The satisfiability (validity) problem for PDL is decidable in NDEXPT(n) (non-deterministic exponential time) where n is the length of the formula.

Proof: *Nondeterministic*, because we have to *guess* a model of size $\leq 2^n$ and then check if it works. However, Pratt [Pr2] was able to eliminate the guessing game, and we get,

Theorem 3.3 (Pratt): PDL is decidable in DEXPT(n) where n is the length of the formula.

In the same paper, Fischer and Ladner also gave a *lower* bound for the computational complexity of PDL.

Theorem 3.4: (Fisher-Ladner): There is a constant $c > 1$ such that the satisfiability (validity) problem for PDL is not a member of DTIME (c^n) where n is the length of the formula, in number of symbols.

Proof: See [FL] theorem 7.2. (The lower bound given there *looks* different since the length of a formula is measured, not in the number of symbols, as in 3.2 above, but in the number of bits.)

In view of Pratt's improvement (theorem 3.3) of the upper bound, theorem 3.4 yields a complete characterisation of the time complexity of PDL.

§4. Axiomatisation

In their paper [FL] where Fischer and Ladner defined PDL and proved that it was decidable, they left open the question of axiomatisation. Of course, if a logic is decidable, then it has *some* recursive axiomatisation. However the question whether there is a *nice* axiomatisation with *axiom schemes* was not clear and results of Redko [R] cast some doubt on the existence of such an axiomatisation. See also [Sa1].

A set of axioms for PDL was suggested by Segerberg [Seg] and proved complete, independently, by Gabbay [G] and Parikh [Pa1]. We give below the resulting axiomatic system. We are treating $[\alpha]$ as basic. $\langle\alpha\rangle$ is an abbreviation for $\neg[\alpha]\neg$. In (1)-(10) below, α is an arbitrary program and A is an arbitrary formula.

(1) All (or enough) tautologies from the propositional calculus.

(2) $[\alpha](A \to B) \to ([\alpha]A \to [\alpha]B)$

(3) $[\alpha \cup \beta]A \leftrightarrow [\alpha]A \wedge [\beta]A$

(4) $[\alpha;\beta]A \leftrightarrow [\alpha][\beta]A$

(5) $[\alpha^*]A \to A \wedge [\alpha]A$

(6) $[\alpha^*]A \to [\alpha^*][\alpha^*]A$

(7) (Induction) $A \wedge [\alpha^*](A \to [\alpha]A) \to [\alpha^*]A$

Rules of inference:

$$\text{(mp)} \quad \frac{A \quad A \to B}{B} \qquad \text{(gen)} \quad \frac{A}{[\alpha]A}$$

Theorem 4.1: (Gabbay-Parikh-Segerberg) A formula A of PDL is valid iff it is provable in the system above.

Proof: For a particularly easy to follow proof, see [KP]. Pratt [Pr2] gives a Gentzen style system for PDL.

It can be shown [Pa2] that the converse operation $\alpha \dashrightarrow \alpha^-$ can be accomodated if the axioms

(8a) $A \to [\alpha]\langle\alpha^-\rangle A$ and (8b) $A \to [\alpha^-]\langle\alpha\rangle A$

are included. The semantics of α^- are given by:

$$R_{\alpha^-} = (R_\alpha)^{-1} = \{(t,s) \mid (s,t) \in R_\alpha\}.$$

Tests are not provided for in the axioms given above, but they can be taken care of with the additional axioms

(9) $[B?]A \leftrightarrow (B \to A)$.

§5. Results on PCAs

The partial correctness assertion (PCA) $A\{\alpha\}B$, expressed in PDL as $A\rightarrow[\alpha]B$ means: "if the precondition A holds when α begins, then the post-condition B holds if and when α terminates." In other words, $M\vDash A\{\alpha\}B$ iff $M\vDash_g A\rightarrow[\alpha]B$. (We do not need to say $M\vDash_g A\{\alpha\}B$, since it is *part* of the semantics of $A\{\alpha\}B$ that it holds throughout the model.) In the section on PCA's below, tests are restricted to be program-free formulae A. Moreover, the formulae that occur in the statements of theorems are also program free. I.e. they are formulae of the propositional calculus (PC from now on). If Γ is a set of formulae, and A is a single formula, then $\Gamma\vDash_{pc}A$ means that A is a truth-functional consequence of Γ. If Γ is empty, then $\vDash_{pc}A$ means that A is a tautology of PC.

Clearly questions about (propositional) PCAs are questions in PDL, but since they are questions of a special type, one may hope for extra information or comparative ease of decision procedures. We shall be interested in the questions: (1) When do a finite number of PCAs imply another? and (2) What is the computational complexity of deciding such an implication? We give information about such questions for the case of (i) a single program, (ii) several loop-free programs, and (iii) arbitrary regular programs with program-free tests.

Definition 5.1: $A_1\{\alpha_1\}B_1,\ldots,A_k\{\alpha_k\}B_k \vDash A\{\alpha\}B$ iff for all models M, if $M\vDash A_i\{\alpha_i\}B_i$ for $\forall i \leqslant k$, then $M\vDash A\{\alpha\}B$.

Theorem 5.2: The set of expressions $(A_1\{\alpha_1\}B_1,\ldots,A_k\{\alpha_k\}B_k,A\{\alpha\}B)$ such that $A_1\{\alpha_1\}B_1,\ldots,A_k\{\alpha_k\}B_k \nvDash A\{\alpha\}B$ is a context sensitive language (CSL).

Proof: See [Pa4].

It can be shown that if C is any truth functional combination of $A_1\{\alpha_1\}B_1,\ldots,A_n\{\alpha_n\}B_n$ then the question "is C satisfiable?" is still in CSL. Moreover, if α is star-free, then the problem is in (NP)$\cap$(linear space).

Theorem 5.3: In theorem 5.2, if the programs α_i are *-free (loop-free), then the problem is NP-complete.

It is known ([St] p. 90, remark 4.14) that the inequivalence of regular expressions is log-lin complete in CSL. Let $\alpha \equiv \beta$ mean that α and β are equivalent regular expressions, i.e. the languages L_α and L_β are equal. Then $\alpha \equiv \beta$ iff $P\{\alpha\}Q \models P\{\beta\}Q$ and $P\{\beta\}Q \models P\{\alpha\}Q$, hence the PCA problem is also log-lin complete in CSL. Thus the bound of theorem 5.2 is best possible.

We now consider the case where the α_i's all equal α. This simple case has some nice characterisations. If A_i, B_i, A, B are formulae of PC, we write $(A_1,B_1),\ldots,(A_k,B_k) \models (A,B)$ for $(\forall\alpha)(A_1\{\alpha\}B_1,\ldots,A_k\{\alpha\}B_k \models A\{\alpha\}B))$.

We state the following lemmas without proof. One of the lemmas is well known. The other also probably occurs in the literature.

Lemma A: Let $\top$, $\bot$ respectively, stand for the truth values true and false. If ϕ is any formula with propositional variables $P_1,\ldots,P_n$ then ϕ is equivalent to
$(\phi(P_1,\ldots,P_{n-1},\top)\wedge P_n)\vee((\phi(P_1,\ldots,P_{n-1},\bot)\wedge\neg P_n)$

Definition 5.4: The propositional function $\phi(P_1,\ldots,P_k)$ is *monotone* iff ϕ can never go from $\top$ to $\bot$ when some P_i goes from $\bot$ to $\top$, the other P_j being fixed.

Lemma B: ϕ is monotone iff ϕ is identically $\top$ or identically $\bot$ or can be expressed using $\wedge$ and $\vee$ only.

Proof: The proof is easy using lemma A above and induction on n.

Definition 5.5: The Floyd-Hoare rules for a single program are:

$$\frac{A_1\{\alpha\}B_1, \qquad A_2\{\alpha\}B_2}{(A_1 ? A_2)\{\alpha\}(B_1 ? B_2)} \qquad \text{where ? is } \vee \text{ or } \wedge.$$

$$\frac{A\models_{pc}B, \quad B\{\alpha\}C, \quad C\models_{pc}D}{A\{\alpha\}D}$$

The axioms are $\perp\{\alpha\}B$ and $A\{\alpha\}\top$.

Notational Remark: For theorem 5.6, let the formulae $A_1,..,A_k,B_1,..,B_k,A,B$ of the PC be constructed from the propositional letters $P_1,...P_m$. A'_i etc. are obtained from A_i etc. when each P_i is replaced everywhere by a new Q_i.

Theorem 5.6:

(a) The Floyd-Hoare rules above are complete, for proving assertions of the form $(A_1,B_1),...,(A_k,B_k) \models (A,B)$.

(b) (i) $(A_1,B_1),...,(A_k,B_k) \models (A,B)$ iff

(ii) there is a monotone ϕ such that $A \models_{pc} \phi(A_1,...,A_k)$ and $\phi(B_1,...,B_k) \models_{pc} B$ iff

(iii) $\models_{pc} ((A_1 \rightarrow B_1') \wedge ... \wedge (A_k \rightarrow B_k')) \rightarrow (A \rightarrow B')$.

(c) The set of all true assertions of the form $(A_1,B_1),...,(A_k,B_k) \not\models (A,B)$ is NP-complete.

Proof: See [Pa4]

§6. On Models for PDL

PDL lacks the compactness property. It is possible for an infinite set Γ to (semantically) imply a formula A even though no finite subset of Γ does. An example is $\Gamma = \{P, [a]P, [a;a]P,..., [a^n]P,...\}$. Then $\Gamma\models[a^*]P$. But no finite subset of Γ implies $[a^*]P$. (A slightly more complicated argument shows that $\models_g$ is also incompact.) Fischer and Ladner have used a filtration technique in their decision procedure which gives us some information about finite models, but none about infinite ones and incompactness makes the latter harder. The reason is that as is shown in [FL], every consistent formula, and hence every consistent finite set of formulae, has a finite model. A consistent infinite set of formulae (i.e. one from which a contradiction cannot be proved) will have the property that every finite subset has a model. However, the set itself may or may not have a model.

A second complication is as follows: Suppose M_1, M_2 are two models with the same set W of states and such that for all $s \in W$, for all formulae A of PDL, $M_1,s \models A$ iff $M_2,s \models A$. Are M_1, M_2 isomorphic? The answer turns out to be "no."

For example, let $W = \{0,1,2,\ldots,\omega\}$. Let $M_1,i \models P_j$ iff $M_2,i \models P_j$ iff $j < i$. $M_1,\omega \models P_j$ and $M_2,\omega \models P_j$ for all j. Now let R_a in M_1 be the set $\{(0,j) \mid j<\omega\}$ and in M_2 the set $\{(0,j) \mid j\leq\omega\}$. Then M_1, M_2 are not isomorphic but have the same formulae holding at the same states of W.

In Definition 6.1 below we define two notions of equivalence between models, (in addition to isomorphism) and provide canonical elements in the equivalence classes for each kind of equivalence. Roughly, in a canonical model there is no duplication of states and each R_a is "as large as possible" without destroying the semantics. A canonical closed model is a canonical model with the additional property that it has all the states that it "might" have. I.e. if a "possible state" is arbitrarily closely approximated by states already in the model, then that possible state is also there. Thus a canonical closed model corresponds to a closed set of real numbers which contains all its limit points. Dexter Kozen, [K1] has also pointed out that such notions are essentially topological in nature. If we consider the topology (see also [Pa5]) on W induced by the family of all sets $U_A = \{s \in W \mid M,s \models A\}$, then if M is canonical then this topology is Hausdorff and if it is canonical closed then M is a closed subspace of M_u below. However, M_u is not compact since, as we noted before, the logic is not compact.

Definition 6.1: Given models M_1, M_2 of PDL,

(i) $M_1 \leq M_2$ iff $\forall s \in W_1$, $\forall B$, $\exists t \in W_2$, $M_1,s \models B$ iff $M_2,t \models B$.

(ii) $M_1 \equiv M_2$ iff $M_1 \leq M_2$ and $M_2 \leq M_1$

(iii) $M_1 \leq_s M_1$ iff $\forall s \in W_1, \exists t \in W_2$, $\forall B$, $M_1,s \models B$ iff $M_2,t \models B$. (The subscript s stands for "strong". The difference between (i) and (iii) is that in (iii) the state t depends only on the state s.)

(iv) $M_1 \equiv_s M_2$ iff $M_1 \leqslant_s M_2$ and $M_2 \leqslant_s M_1$.

Theorem 6.2: ($M_1 \approx M_2$, M_1 is isomorphic to M_2)
$\Rightarrow (M_1 \equiv_s M_2) \Rightarrow (M_1 \equiv M_2)$.

Proof: Trivial.

None of the reverse implications hold. Thus the equivalence classes for $\equiv_s$, $\equiv$, are not singletons (modulo isomorphism). However, we can find canonical elements of them. I.e. in each equivalence class under $\equiv$ or $\equiv_s$, there is a "nicest" element.

Definition 6.3: M is *canonical* iff

(1a) $\forall p, q \in W,\ p \neq q \Rightarrow (\exists B)(M,p \models B\ \wedge M,q \models \neg tB)$.

and (1b) $\forall p, q \in W,\ (p,q) \in R_a$ iff $\forall B,\ M,p \models [a]B \Rightarrow M,q \models B$.

M is *canonical closed* iff

(2a) it is canonical and

(2b) Given M',q, if for all B $\exists p$ such that $M',q \models B \rightarrow M,p \models B$, then $\exists p_0$ such that $\forall B,\ M',q \models B \rightarrow M,p_0 \models B$.

Note that the "only if" half of condition 1b above holds in all models of PDL simply because of the semantics of $[\alpha]$. However, in canonical models, the R_a are "packed full" so that the other direction holds as well. The process of filling out the R_a will always produce a filled out model without changing the semantics. Now if the states that satisfy the same formulae are identified, then we get a canonical model, in which each state still satisfies the same formulae. This proves part 1 of theorem 6.4, next. A canonical closed model has an extra property. If M is canonical closed, and there is a "possible state" s, and M has arbitrarily close *approximations* to s, then M contains a *copy* of s. hence the word "closed", which can be given a topological meaning.

Theorem 6.4:

(1) $\forall M\ \exists! M'$, M' is canonical and $M \equiv_s M'$.

(2) $\forall M\ \exists! M'$, M' is canonical closed and $M \equiv M'$.

(3) $\exists!$ universal M_u canonical closed such that for all M,
$\exists! \theta : W \dashrightarrow W_u$
s.t. $\forall B,\ M,p \models B$ iff $M_u,\theta(p) \models B$. (Here $\exists!$ means: there is a unique...).

Proof: See the appendix.

In the theorem below, S ranges over finite Fischer-Ladner closed sets of formulas of PDL. M/S is the factor model, as defined in §3.

Theorem 6.5:
(i) $M_1 \equiv M_2$
iff (ii) $\forall S$, $M_1/S \equiv M_2/S$
iff (iii) $\forall S$, $M_1/S \equiv_S M_2/S$
iff (iv) $\forall S$, $M_1/S \approx M_2/S$.

Proof: See [Pa4].

§7. DPDL and PDL:

DPDL has the language and semantics of PDL but the class of models is restricted by the requirement that all atomic programs are deterministic. I.e. each R_a is a partial function. The filtration technique, used in [FL] for PDL, fails for DPDL which is therefore less easy to understand. The reason is that if one starts with a model M, which is, in fact a model for DPDL, and then *factorises* out a congruence relation of a suitable kind, as is done in [FL], [Pa2] for PDL, one need *not* get a finite model of DPDL. Rather one can end up identifying distinct initial states in W and the result is that an atomic program which was deterministic in M may become non-deterministic in the factor model. Thus DPDL is not *easier* to study than PDL, but more complex. We substantiate this argument in this section by pointing out a translation from PDL to DPDL which yields a lower bound for the complexity of the latter.

Notation: For this section only, if A is a formula of PDL then A′ is obtained when every program atom a_i is replaced by $(a_i;b^*)$ where b is new.

Note that the map A --> A′ is computable in linear time and leads to a linear growth in length.

Theorem 71: $\Gamma \models A$ in PDL iff $\Gamma' \models A'$ in DPDL.

Proof: The proof given in [Pa4] contains a mistake. For a corrected proof, see the appendix.

Theorem 7.1 above is not dependent on the class of programs being regular. It applies equally well to context free (recursive) programs or even larger classes of programs.

Since theorem 7.1 reduces PDL to DPDL, the lower bound for PDL ([FL] theorem 7.2) automatically becomes a lower bound for DPDL.

Theorem 7.2: There is a constant $c > 1$ such that the satisfiability (validity) problem for DPDL is not a member of DTIME (c^n) where n is the length of the formula, in number of symbols.

Proof: This is an immediate corollary of theorem 7.1 above and [FL] theorem 5.2, which is exactly like theorem 7.2 except that it is about PDL.

Theorem 7.3 (Ben-Ari-Halpern-Pnueli). The validity problem for DPDL is decidable in time deterministic exponential in the length of the formula. Moreover, for a formula A of length n, if A is satisfiable, then it has a model of size at most $4^n.n!$.

Finally, while every formula in the notation of PDL that is valid in PDL, is also valid in DPDL, deterministic programs α satisfy the *additional* formulae

(10) $\langle\alpha\rangle A \rightarrow [\alpha]A$

where A is arbitrary. It has been shown by J. Halpern [Hal] that adding axioms (10) where α is a *primitive* program yields a complete axiomatisation for DPDL. His argument to show this is quite literally thorny. See [Hal] to appreciate this pun. Another, independent proof, also appears in [V].

Theorem 7.4 (Halpern-Valiev): Any complete axiomatisation for PDL becomes a complete axiomatisation for DPDL if the axiom scheme

$\langle a \rangle A \to [a]A$

is added where a is an arbitrary atomic program and A is an arbitrary formula.

We could also define a strict version of DPDL where not only the atomic programs but also other programs are deterministic, i.e. where $\cup$ and * are eliminated in favour of "if..then..else.." and "while..do..". We do not have any specific information about strict DPDL, except what we know about it as a subsystem of DPDL.

§8. Process Logics

Process logics in the context of dynamic logic occur first in [Pr3] where Pratt defined several process connectives to augment the box and diamond of dynamic logic. An example of such a connective is *throughout*, written $\sqcup\!\sqcup$. Thus if α is a program, A is a formula, and M is a model, then $\sqcup\!\sqcup\alpha A$ holds at a state s if the property expressed by A holds throughout every execution of α that begins at s. Now clearly this property of α cannot be ascertained merely by looking at the binary relation R_α which knows only about the initial and final state of any execution of α. To be able to talk about connectives like $\sqcup\!\sqcup$ at all, α must be represented, not as a binary relation, but as a set of *trajectories*, a trajectory being a complete list of states during any one execution of a program.

In [Pa3] we defined a language SOAPL for process logic which included Pratt's process logic. SOAPL was proved decidable by reducing it to SnS, whose decidability had been proved by Rabin [Ra]. Harel [H1] showed that SOAPL was more expressive than Pratt's version of process logic defined in [Pr3]. However, apart from the fact that the decision procedure did not look elementary, the system in SOAPL was syntactically quite complex. A simpler system, using results of Hans Kamp to achieve expressive simplicity, was defined by Nishimura [Ni]. The principal result of [Ni] is that the Nishimura system includes SOAPL. Nishimura's system is further refined in [HKP] and completeness and decidability results are proved. the system PL of [HKP] is not elementarily decidable because the first order theory of linear order is reducible to it. We give below the syntax and semantics of PL and give the set of complete axioms. For details see [HKP].

The syntax of PL is obtained from that of PDL by adding three new operators *f* (for "first") *chop*, and *suf* (for "suffix"). We extend definition 2.2 by adding the condition:

(vi) If A and B are formulae, then so are fA, AchopB, and AsufB.

The difference in semantics is greater. In a model of PL, truth values of formulae are attached not to states, as in PDL, but to finite or infinite sequences of states, called *paths*, from now on. Thus a model of PL consists of a universe of states W, together with an assignment of a set R_α of paths to each atomic program α, and a set of paths $\rho(P)$ to each atomic formula P. In what follows, M is a model and p,q are paths. The symbols s,t refer to states as usual. If p, q are paths $(s_0,\dots,s_k)$ and $(t_0,\dots,..)$ respectively, (p must be finite, but q may be finite or infinite), then pq is defined iff $s_k = t_0$ and in that case pq = $(s_0,\dots,s_k,t_1,\dots,..)$. The length l(p) of p is k, the number of states on p minus 1. Note that $l(pq) = l(p) + l(q)$. We shall use the words "prefix" and "suffix" respectively, to denote initial segments and final segments of paths. We assume that the given atomic predicates are state predicates. I.e. that if p,and q are two paths with the same initial state, then for an atomic P, $p \in \rho(P)$ iff $q \in \rho(P)$. This last condition will be called the *locality* condition, and models satisfying this condition will be called *local* models.

Definition 8.1: We define the semantics of Process Logic.

(i) R_a is given and we define

$$R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$$
$$R_{\alpha;\beta} = \{pq \mid p \in R_\alpha \wedge q \in R_\beta\}$$
$$R_{\alpha*} = \{p_1 \cdots p_m \mid m \geq 0 \text{ and for all } i \leq m,\ p_i \in R_\alpha\}.$$

(ii) $M,p \models P$ iff $p \in \rho(P)$

(iii) $M,p \models A \vee B$ iff $M,p \models A$ or $M,p \models B$

(iv) $M,p \models \neg A$ iff $M,p \not\models A$

(v) $M,p \models [\alpha]A$ iff for all $q \in R_\alpha$, if pq is defined, then $M,pq \models A$

(vi) $M,p \models fA$ iff $M,(s_0) \models A$.

(Note that (s_0), denoted first(p), is a prefix of p of length 0)

(vii) $M,p \models A \text{suf} B$ iff $\exists q$ such that

(a) q is a *proper* suffix of p and $M,q \models B$ and

(b) if r is a proper suffix of p and q is a proper suffix of r, then $M,r \models A$.

(viii) $M,p \models A \text{chop} B$ iff $\exists q$, r such that $p = qr$ and $M,q \models A$ and $M,r \models B$.

Using the given notions of PL, we can easily define some very useful auxiliary notions. Recall that $\top$ stands for "true", $\bot$ for "false".

Definition 8.2:

(i) $nA = \bot \text{suf} A$

(ii) $\bar{n}A = \neg n \neg A$

(iii) $L_0 = \neg n \top$

($M,p \models L_0$ iff p has length 0.)

(iv) $\text{some}A = \top \text{suf} A$

(v) $\text{all}A = \neg \text{some} \neg A$

(vi) $\text{fin} = L_0 \vee \text{some}L_0$

We see now how the notions of temporal logic [GPSS] can be interpreted in PL. The connectives some, all, next, and suf correspond to the connectives F, G, X, until of TL (temporal logic). Thus PL is expressively as rich as TL. It also includes PDL, and indeed, we can think of (local) PL as a sort of least upper bound of PDL and TL. We now state some decidability and undecidability results for PL.

Theorem 8.3: The validity problem for local Process logic with the connectives f, suf, and chop is decidable but not elementary.

Proof: The proof of part of this result has already appeared in [HKP], where the decidability of PL without chop is proved by reducing it to SnS in the same way as SOAPL. The extension to include chop is straightforward. The non-elementarity of PL is shown by using it to describe the computations of Turing machines with a bounded (but very large) amount of tape.

Theorem 8.4: The validity problem for Process logic *without* the locality condition is undecidable.

Proof: This theorem is proved by showing that the problem of the satisfiability of equations between regular expressions with variables for languages, is undecidable, and then showing how to reduce this problem to the satisfiability problem for (non-local) Process logic. For details, see [CHMP].

§9. Dynamic Algebra. (*Note*: This section is due to Dexter Kozen)

Dynamic algebra is an algebraic abstraction of the standard Kripke model semantics of PDL. It plays exactly the same role in PDL as Boolean algebra in classical propositional logic: it provides a cleaner level of abstraction, independent of the vagaries of syntax. Theoretical computer science provides many examples of dynamic algebras besides Kripke models; see [Pr5].

Dynamic algebras were first introduced in [K1] and subsequently studied by Pratt [Pr4,5,7], Reiterman and Trnkova [RT1], Nemeti [N], and the author [K1-5]. A survey of these results follows.

Definition of dynamic algebras

PDL has two sorts, propositions and programs. Accordingly, a dynamic algebra is a two-sorted algebraic structure $(K,B,\langle\rangle)$, where: B is a Boolean algebra; K is a *Kleene algebra* or algebra of regular events [C] with operators $;$, $\cup$, $*$, 0, λ (identity), and sometimes $^-$ (reverse); and $\langle\rangle$ is a "scalar multiplication" $K \times B \rightarrow B$.

There are several possible definitions of Kleene algebras (see [C]). As the axioms we will use appear elsewhere in this volume [K2], we will restrict ourselves to some examples: the family of all binary relations on a set S, where $\cup$ is set union, $;$ is relational composition, 0 is the null set, λ is the identity relation, and $*$ is reflexive transitive closure; the family of regular sets over $\{0,1\}^*$, where $;$ is concatenation and λ is the set containing only the null string of $\{0,1\}^*$; any Boolean algebra, where λ is 1, $;$ is $\wedge$, and $\alpha^*= 1$ for all α; and the structure MIN consisting of the extended natural numbers $\mathbb{N}\cup\{\infty\}$, where $\cup$ gives the minimum of two numbers, $;$ is addition, the Kleene algebra constant λ is the number 0, the Kleene algebra constant 0 is ∞, and $\alpha^* = 0$ for all α. The last structure appears in the study of shortest path problems [AHU]. Other examples appear in [Pr5].

The axioms for scalar multiplication $\langle\rangle$ are just the Segerberg axioms for PDL, with the exception that Segerberg *induction axiom*

(ind) $(X \wedge [\alpha^*](X \to [\alpha]X)) \to [\alpha^*]X$

is replaced by the stronger **-continuity condition*

(*-cont) $\langle\alpha^*\rangle X = \bigvee_n \langle\alpha^n\rangle X$

which says that $\langle\alpha^*\rangle X$ is the supremum of the countable collection $\langle\alpha^n\rangle X$ with respect to the natural ordering $\leq$ on the Boolean algebra **B**.

The definition above is the original definition of dynamic algebras that appeared in [K1], however Pratt [Pr4,5,7] later adopted a more general definition of dynamic algebras in which K was not required to satisfy any axioms at all and $\langle\rangle$ was not required to satisfy the *-continuity condition, but only the induction axiom. In other words, Pratt's definition says that a dynamic algebra is simply a two-sorted algebra satisfying the Segerberg axioms. In the interest of conservation of terminology, we shall for the present adopt Pratt's terminology and use "dynamic algebras" to refer to this wider class of models and call those models satisfying the original definition above **-continuous.*

In a nonstandard Kripke model, the *-continuity condition does not imply that $\langle\alpha^*\rangle X = \bigcup_n \langle\alpha^n\rangle X$ in general, since the set $\bigcup_n \langle\alpha^n\rangle X$ may not be an element of B; it says rather that $\langle\alpha^*\rangle X$ is the smallest element of **B** containing all the sets $\langle\alpha^n\rangle X$. All dynamic algebras arising in practice, including and especially the standard Kripke models, are *-continuous. A proof that ***-cont** implies **ind** can be found in [K1] and a non-*-continuous dynamic algebra is given in [Pr5].

The scalar multiplication $\langle\rangle$ is so called because three of the PDL axioms, namely

$$\langle\alpha \cup \beta\rangle X = \langle\alpha\rangle X \vee \langle\beta\rangle X$$
$$\langle\alpha\rangle(X \vee Y) = \langle\alpha\rangle X \vee \langle\alpha\rangle Y$$
$$\langle\alpha\beta\rangle X = \langle\alpha\rangle(\langle\beta\rangle X)$$

are reminicent of the axioms for scalar multiplication in vector spaces or modules, with $\cup$ and $\vee$ playing the role of addition and ; playing the role of multiplication of scalars.

The property of separability

A dynamic algebra $D = (K, B, \langle\rangle)$ is called *separable* if for any $\alpha, \beta \in K$, $\alpha \neq \beta$, there exists an $X \in B$ such that $\langle\alpha\rangle X \neq \langle\beta\rangle X$; in other words, distinct elements of K are distinguished by their action as scalars. A Kleene algebra K is *inherently separable* if there exists a separable dynamic algebra over K . The property of separability turns out to be very important in the theory of dynamic algebras.

Not all dynamic algebras are separable and not all Kleene algebras are inherently separable (an example of a non-inherently separable Kleene algebra is the MIN example given above; see [K1]). However, the Kleene algebra of any standard Kripke model is inherently separable, since the Boolean algebra can be augmented to include all subsets of states.

If we define the relation $\approx$ of *inseparability* by

$$\alpha \approx \beta \text{ iff for all } X,\ \langle\alpha\rangle X = \langle\beta\rangle X$$

then the relation $\approx$ is a dynamic algebra congruence, thus there is a quotient algebra $D/\approx\ = (K/\approx, B, \langle\rangle)$, where

$$K/\approx\ = \{ \alpha/\approx \ | \ \alpha \in K \}$$

and $\alpha/\approx$ is the $\approx$-congruence class of α. The quotient algebra is separable, and $(K, B, \langle\rangle)$ is separable iff $\approx$ is the identity on K .

The equational theory of dynamic algebras

If the defined propositional operator $\equiv$ is considered an equality, then without loss of generality PDL can be considered an equational system. Instead of the PDL assertion X one writes the equation $X \equiv 1$, and instead of the Hilbert-style rules of inference proposed by Segerberg one uses the single rule of substitution of equals for equals. It then follows that the completeness of the Segerberg system for PDL is equivalent to the coincidence of the equational theory of dynamic algebras and the equational theory of standard Kripke models. This is explained in more detail in [Pr4]. In that paper, Pratt argued that the following models of computation give examples of dynamic algebras, and all have the same equational theory, namely that of dynamic algebras: standard Kripke models, nonstandard Kripke models, finite Kripke models, flowchart models, regular languages, trajectory models, predicate transformer algebras. The class of *-continuous dynamic algebras should be included in this list as well, since it contains all Kripke models and is contained in the class of dynamic algebras. These results say that several extant models of computation, although intuitively quite distinct, are logically indistinguishable as far as equations are concerned.

In [Pr7], Pratt proved a purely algebraic result about dynamic algebras that generalizes the Fischer/Ladner finite model property [FL] and admits the completeness of the Segerberg axioms of PDL as a corollary:

Theorem [Pr7]. Every free separable dynamic algebra is a subalgebra of a direct product of finite dynamic algebras.

In fact, Pratt's theorem is somewhat stronger, but this version suffices for all practical purposes.

Proof. Let $D = (K,B,\langle\rangle)$ be the free separable dynamic algebra on the generating sets B, K. Then D is isomorphic to the term algebra over B and K modulo the Segerberg axioms and the relation $\approx$ defined in the previous section.

If X is any term over B and K, construct a finite dynamic algebra D_X as follows: let FL(X) denote the finite set of elements of B represented by terms in the Fischer/Ladner closure of X (see [FL]) and let B_X be the finite Boolean subalgebra of B generated by FL(X). Let K_X be the set of all functions $\alpha: B_X \dashrightarrow B_X$ satisfying the two properties (1) $\alpha(Y \vee Z) = \alpha(Y) \vee \alpha(Z)$ and (2) $\alpha(0) = 0$. D is made into a dynamic algebra by defining $\langle\alpha\rangle Y = \alpha(Y)$. Also, D is separable.

Now define the map $f_X: D \dashrightarrow D_X$ as follows: for generators $P \in B$, if P appears in the term X then $f_X(P) = P$, otherwise $f_X(P) = 0$. For generators $\alpha \in K$,

$$f_X(\alpha)(Y) = \{ Z \in B_X \mid \langle\alpha\rangle Y \leq Z \}.$$

Since **D** is freely generated by B and K, f_X extends uniquely to a homomorphism f_X:**D**-->$\mathbf{D}_X$. It can now be proved by induction on the structure of terms that for any $Y \in FL(X)$, $f_X(Y) = Y$; in particular, $f_X(X) = X$.

Now let $\Pi\mathbf{D}_X$ be the direct product of all these $\mathbf{D}_X$ for all terms X. **D** is embedded in $\Pi\mathbf{D}_X$ via the map $f = \Pi f_X$ which takes Y to the sequence $\Pi f_X(Y)$ and α to the sequence $\Pi f_X(\alpha)$. The map f is one-one on **B** since if $f(X) = f(Y)$ then $f(X\Delta Y) = 0$ (where $X\Delta Y = X\text{-}Y \cup Y\text{-}X$) and hence $f_{X\Delta Y}(X\Delta Y) = 0$, therefore $X\Delta Y = 0$ and $X = Y$. It is also one-one on K since if $f(\alpha) = f(\beta)$ then $f(\langle\alpha\rangle X) = f(\langle\beta\rangle X)$ for all X, so $\langle\alpha\rangle X = \langle\beta\rangle X$ for all X by the above argument, therefore $\alpha = \beta$ by separability. ■

Corollary. The Segerberg axioms for PDL are complete.

Proof. It suffices to show that the equational theory of the class of dynamic algebras coincides with the equational theory of the standard Kripke models. The inclusion in one direction is trivial. Now suppose the equation $X = Y$ holds in all standard Kripke models. Then it certainly holds in all finite Kripke models. Since every finite dynamic algebra is represented by a finite Kripke model, it holds in all finite dynamic algebras as well. Since equations are preserved by direct products and subalgebras, it follows from the above theorem that $X = Y$ in any free separable dynamic algebra, and therefore $X = Y$ in all separable dynamic algebras. Now for any dynamic algebra *C*, $C/\approx$ is separable, hence $X = Y$ in $C/\approx$. Therefore $X = Y$ in *C*. ■

Another corollary to Pratt's theorem is that every free dynamic algebra with at least one Boolean generator is separable and represented by standard Kripke model. Nemeti [N] has removed the restriction that there be at least one Boolean generator.

The representation of dynamic algebras

The Stone representation theorem for Boolean algebras says that every Boolean algebra is isomorphic to a Boolean algebra of sets. An analog of this result holds for separable dynamic algebras and nonstandard Kripke models:

Theorem [K1]. Every separable dynamic algebra is isomorphic to a (possibly nonstandard) Kripke model.

Proof. The construction is an extension of that of the Stone representation theorem for Boolean algebras. Let (K,B,$\langle\rangle$) be a separable dynamic algebra and let u, v denote ultrafilters of B. The set S of states will be the the set of ultrafilters of B. For each $X \in B$, the set X' will be the set of ultrafilters containing X, and B' will be the set of all such X'. Under the set-theoretic Boolean algebra operations, B' is a Boolean algebra isomorphic to B (this is just the Stone representation theorem). Now for each $\alpha \in K$, define the binary relation α' on S by: $(u,v) \in \alpha'$ iff $\langle\alpha\rangle X \in u$ whenever $X \in v$, or equivalently, iff $X \in v$ whenever $[\alpha]X \in u$. It turns out that the set K' of all such α' is a Kleene algebra under the usual binary relation-theoretic interpretations of the operators ; , $\cup$, and $^-$, and K' is isomorphic to K . Moreover, under the standard Kripke model interpretation of $\langle\rangle$, (K',B',$\langle\rangle$) is a dynamic algebra isomorphic to (K,B,$\langle\rangle$). In general, however, the * operation in K' will not be reflexive transitive closure in the binary-relation theoretic sense, hence (K',B',$\langle\rangle$) may be nonstandard. ∎

Dynamic algebras which are not Kripke models

There are examples of dynamic algebras, even separable *-continuous dynamic algebras, which are not isomorphic to any standard Kripke model [K3,RT1,K4]. The following counterexample is from [RT1]:

Let B be the power set of ω. Define the function $\langle a\rangle$:B --> B by

$$\langle a\rangle Y = \{ x \mid \exists y \in Y \ \ |x-y| \leq 1 \} \text{ if } Y \text{ is finite, } \omega \text{ otherwise.}$$

Let K be the Kleene algebra generated by $\langle a\rangle$ under union for $\cup$ and functional composition for ; , with $\langle\alpha^*\rangle X = \omega$ for all $X \neq 0$, $\langle\alpha^*\rangle 0 = 0$. It is not hard to show that (K,B,$\langle\rangle$) is a separable dynamic algebra. Moreover, it is *-continuous, since for any α and $X \neq 0$, the sequence $\langle\alpha^n\rangle X$ is a chain of sets whose union is ω, and $\langle\alpha^*\rangle X = \omega$. However, (K,B,$\langle\rangle$) is not represented by any standard Kripke model, because it does not satisfy the following condition, which is clearly satisfied by any standard model: if Y_n is a set of Boolean elements for which $\bigvee_n Y_n$ exists, then for any α, $\bigvee_n \langle\alpha\rangle Y_n$ exists and is equal to $\langle\alpha\rangle \bigvee_n Y_n$. This does not hold in the $(K,B,\langle\rangle)$ constructed above, which can be seen by taking $Y_n = \{4n\}$ and $\alpha = a$.

This counterexample does not work in the presence of the reverse operator $^-$:

Proposition [RT2]. In all dynamic algebras with $^-$, if Y_n is any set of Boolean elements whose supremum $\bigvee_n Y_n$ exists, then for any α, $\bigvee_n \langle\alpha\rangle Y_n$ exists and is equal to $\langle\alpha\rangle \bigvee_n Y_n$.

Proof. Clearly $\langle\alpha\rangle \bigvee_n Y_n$ is an upper bound for $\langle\alpha\rangle Y_n$. If Z is any other upper bound, then $\langle\alpha\rangle Y_n \leq Z$ for all n, so $[\alpha^-]\langle\alpha\rangle Y_n \leq [\alpha^-]Z$ for all n, since $[\alpha^-]$ is monotone, and by one of the $^-$ axioms,

$$Y_n \leq [\alpha^-]\langle\alpha\rangle Y_n \leq [\alpha^-]Z$$

for all n, therefore $\bigvee_n Y_n \leq [\alpha^-]Z$. But then by the other axiom for $^-$,

$$\langle\alpha\rangle \bigvee_n Y_n \leq \langle\alpha\rangle[\alpha^-]Z \leq Z\ ,$$

therefore $\langle\alpha\rangle \bigvee_n Y_n$ is the least upper bound. ∎

Nevertheless, even in the presence of $^-$, not every separable *-continuous dynamic algebra is isomorphic to a standard Kripke model; counterexamples can be found in [K3,K4]. On the other hand, there are certain conditions under which dynamic algebras are represented by standard Kripke models. For example, every finite dynamic algebra is so representable. A harder example is given by

Theorem [K4]. Every *-continuous separable dynamic algebra over an atomic Boolean algebra is represented by a standard Kripke model.

The duality of dynamic algebras and Kripke models

A *Boolean space* is a topological space that is compact and Hausdorff and has a base of clopen sets. There is a well-known duality between Boolean algebras and Boolean spaces: after performing the Stone representation theorem on a Boolean algebra B to obtain a Boolean algebra B' of subsets of a set S, if the sets X' are allowed to generate a topology on S, then the resulting space is a Boolean space. Likewise, the clopen sets of any Boolean space form a Boolean algebra. This duality is very useful in that it admits the perspectives and techniques of two branches of mathematics.

Like set-theoretic Boolean algebras, Kripke models (S,K,B) with state set S have a natural topology on S, namely that generated by the elements of B. Attempting to adapt the Stone duality to dynamic algebra leads us to define *dynamic spaces* as those topological Kripke models (S,K,B) for which (1) (S,B) is a Boolean space, and (2) all elements of K are closed in the product topology. We arrive at a duality between separable dynamic algebras and dynamic spaces completely analogous to the duality between Boolean algebras and Boolean spaces.

If D is a separable dynamic algebra, let $\mathcal{S}(D)$ denote the nonstandard Kripke model obtained in the representation theorem. If A is a Kripke model, let $\mathcal{C}(A)$ denote its dynamic algebra.

Theorem [K2]. (1) If D is a separable dynamic algebra then $\mathcal{S}(D)$ is a dynamic space. (2) If A is a dynamic space then $\mathcal{C}(A)$ is a separable dynamic algebra.

Theorem [K2]. (1) If D is a separable dynamic algebra, then D is isomorphic to $\mathcal{C}(\mathcal{S}(D))$. (2) If A is a dynamic space, then A is homeomorphic to $\mathcal{S}(\mathcal{C}(A))$.

This duality gives a useful topological perspective to problems in dynamic algebra. For example, the *-continuity condition for a dynamic algebra says that in the corresponding nonstandard Kripke model obtained from the representation theorem, the set $\langle\alpha^*\rangle X$ is the topological closure of the set $\cup_n\langle\alpha^n\rangle X$, so that the set $\langle\alpha^*\rangle X - \cup_n\langle\alpha^n\rangle X$ of "nonstandard points" for α and X is nowhere dense. (In standard models, since * is reflexive transitive closure, $\langle\alpha^*\rangle X = \cup_n\langle\alpha^n\rangle X$.)

Further details are given elsewhere in this volume [K2].

*Induction vs. *-continuity*

If dynamic algebras, *-continuous dynamic algebras, and standard Kripke models are indistinguishable by equations, what does the assumption of *-continuity give us? This question is answered in the two papers [K4,5]. Recall the relation $\approx$ of inseparability defined above. If A is a dynamic algebra, let $A/\approx$ denote the separable quotient algebra. It is not hard to show that if A is *-continuous then so is $A/\approx$.

Theorem [K4]. Any countable separable *-continuous dynamic algebra is isomorphic to $A/\approx$ for some standard Kripke model A.

Proof. Let $(K,B,\langle\rangle)$ be a separable *-continuous dynamic algebra. If the construction of the representation theorem of [K1] is carried out, the result is a (possibly nonstandard) Kripke model with the same dynamic algebra $(K,B,\langle\rangle)$. Let S denote the set of states.

In spite of the fact that $\langle\alpha^*\rangle X$ need not be $\cup_n\langle\alpha^n\rangle X$, the *-continuity condition guarantees that $\langle\alpha^*\rangle X$ is the least element of B containing $\cup_n\langle\alpha^n\rangle X$. In the natural topology on S (generated by the elements of B), this says that sets of the form $\langle\alpha^*\rangle X - \cup_n\langle\alpha^n\rangle X$ are nowhere dense. Therefore, if K and B are both countable, then the union of all such sets, call it M, is meager. The Baire Category Theorem then implies that every nonnull $X \in B$ intersects $S - M$; using this fact, it can be shown that all points of M can be dropped from the Kripke model without changing the dynamic algebra.

The resulting Kripke model may still be nonstandard, for although now $\langle\alpha^*\rangle X = \cup_n\langle\alpha^n\rangle X$, it is still not necessary that α^* be the reflexive transitive closure of α. However, the elements of K , taken as primitive, generate a standard Kripke model A, using reflexive transitive closure instead of *. Since $\langle\alpha^*\rangle X = \cup_n\langle\alpha^n\rangle X$, this process introduces no new Boolean elements. Using this and the fact that $(K,B,\langle\rangle)$ is separable, it follows that $(K,B,\langle\rangle) \simeq A/\approx$, thus A is the desired standard model.

The above result does not hold for non-*-continuous algebras. This is the key to the power of *-continuity. Let us extend the equational language to allow quantification over the X and α to get the first-order language $L_{\omega,\omega}$, and extend further to allow countable conjunctions and disjunctions to get the infinitary first-order language $L_{\omega_1,\omega}$.

Lemma [K5]. A and $A/\approx$ are equivalent with respect to all $L_{\omega_1,\omega}$ sentences.

Proof. Straightforward induction on formula structure. ∎

Theorem [K5]. (i) There is a first-order sentence true in all standard Kripke models but violated in some separable dynamic algebra. (ii) The class of *-continuous dynamic algebras and the class of standard Kripke models have the same $L_{\omega_1,\omega}$ theory.

Proof. (i) An *atom* of a Boolean algebra is a minimal nonzero element. An element X of a Boolean algebra is *atomless* if there does not exist an atom $Y \leq X$. An element X is said to be atomic if no nonzero $Y \leq X$ is atomless, or in other words, if every nonzero $Y \leq X$ has an atom $Z \leq Y$. The properties of being atomless or atomic are first-order expressible. We construct an dynamic algebra $(K,B,\langle\rangle)$ whose Boolean algebra **B** is a subalgebra of the direct product of an atomic Boolean algebra and an atomless Boolean algebra. K has a program δ such that both the atomic part and the atomless part of B are preserved under application of $\langle\delta\rangle$, but neither part is preserved under $\langle\delta^*\rangle$. The structure $(K,B,\langle\rangle)$ therefore violates the first-order property "for any α, if $\langle\alpha\rangle X$ is atomless whenever X is, then $\langle\alpha^*\rangle X$ is atomless whenever X is." On the other hand, any standard Kripke model has this property, since $\langle\alpha^*\rangle X = \bigcup_n \langle\alpha^n\rangle X$, and if all elements of a family of sets are atomless, then so is their union.

(ii) Let ϕ be any sentence of $L_{\omega_1,\omega}$. We wish to show that ϕ is satisfied by some standard Kripke model iff ϕ is satisfied by some *-continuous dynamic algebra. One direction is trivial. Now suppose ϕ is satisfied by some *-continuous dynamic algebra. By the downward Lowenheim-Skolem theorem for infinitary logic [Ke], ϕ is satisfied by a countable *-continuous dynamic algebra B. By the Lemma, ϕ is also satisfied by the countable *-continuous dynamic algebra $B/\approx$, and $B/\approx$ is separable, thus by the Theorem of [K4], $B/\approx \, \underset{\sim}{} \, A/\approx$ for some standard Kripke model A. Again by the Lemma, $A \models \phi$.

It can also be shown that the *-continuity condition for Kleene algebras gives a natural (although infinitary) complete axiomatization for the algebra of regular events. It has been shown that there is no finite equational axiomatization [R] and the finitary axiomatizations that have been given are somewhat unnatural and difficult to prove complete [Sa1].

§10. Brief Notes

In a paper of this size, there are bound to be omissions. We have not discussed several topics that we do not know well enough or which have already been discussed quite adequately in other places. We briefly mention some of these topics and refer the reader to the appropriate sources in the literature.

Propositional algorithmic logics: These are propositional versions of algorithmic logic, just as PDL is the propositional version of dynamic logic. For details see [Mi].

Probabilistic logics: If a program is non-deterministic, it is very tempting to assign probabilities to the various executions of the program. Then the problem arises, of computing the probabilities for a complex program in terms of the probabilities of its components. Once this is done in a satisfactory way, questions of completeness and decidability will arise. There is some spadework in [K6] and [Re], but the subject is still in a developing state.

Parallel programs: It is not hard to see that the semantics of PDL does not contain enough information to tell us how the program will act when it is interacting in parallel with other programs. That is, two programs α and β may satisfy $R_\alpha = R_\beta$, and yet, α and β may act differently in the presence of another program γ with whom they are acting in parallel. One approach is to augment PDL by adding a shuffle operator. This is done in [Ab].

Appendix

Proof of theorem *6.4*: We already have shown (1). To see (3) take all semantically consistent, complete sets of formulae. I.e. for any s in any model M, let $\tilde{s}$ be the set of formulae that hold in M at s. Let

W_u be the set of all these $\tilde{s}$ coming from these millions of models. And now define $R_{a,u}$, the relation R_a in M_u, to be the set

$$\{(\tilde{s},\tilde{t}) \mid (\forall A)([a]A \in \tilde{s} \Rightarrow A \in \tilde{t})\}.$$

It is quite straightforward to show that $M_u,\tilde{s}\models A$ iff $A \in \tilde{s}$. This M_u is the required universal model. For any M the map $\theta(s) = \tilde{s}$ is the required map.

Finally, to see (2), the M′ there is just the subset of W_u consisting of those $\tilde{s}$ such that every element of $\tilde{s}$ is true at some state of M.

Proof of theorem 7.1: Suppose $\Gamma\models A$ in PDL, then by substituting the programs $(a_i;b^*)$ for the atomic programs a_i, we have $\Gamma'\models A'$ in PDL. However, every model of DPDL is also a model of PDL and hence whatever happens in all PDL models of Γ′ must also happen in all DPDL models of Γ′. I.e. $\Gamma'\models A'$ in DPDL.

Conversely, suppose that $\Gamma\not\models A$ in PDL. Then there is a (countable) model M of PDL in which Γ holds and A does not, say at some state s. By standard techniques one can show that M can be a tree model. Such tree models are obtained by unwinding loops. This converts a possibly finite model with loops into an infinite tree model which is, in some sense, simpler. W is, of course, countable. Now for each state s of W consider the states t such that $(s,t) \in R_a$. Enumerate them in some order t_1,.... Let $(s,t_1) \in R_{a'}$, and for all n, if t_n, t_{n+1} both exist, then the pair $(t_n,t_{n+1}) \in R_b$. (If there are no such states t_i, then $R_{a'}$ will have *no* pair of states whose first element is s.) It is easily seen that each $R_{a'}$ and R_b is deterministic and $R_a = R_{a';b^*}$. Thus the program $R_{a';b^*}$ takes us to precisely the places where R_a does, but is made up of *deterministic* atoms $R_{a'}$, R_b. Replacing the various R_a by the $R_{a'}$ and R_b, defines a new model M′ such that M′ is a model of DPDL and such that if the state s satisfies a formula A in M then s satisfies A′ in M′. Thus s satisfies Γ′ in M′ and does not satisfy A′ in M′. Hence $\Gamma' \not\models A'$ in DPDL.

Acknowledgements

We have benefited from discussions with Georg Gati, Joe Halpern, David Harel, Dennis Kfoury, Richard Ladner, and Jerzy Tiuryn. Special thanks are due to Dexter Kozen, who wrote for us the section on Dynamic Algebras, to Erwin Engeler who organised the Zurich seminar in logics of programs in July 1979; to our colleague Albert Meyer, who introduced us to dynamic logic; and to Vaughan Pratt who invented the subject of dynamic logic. Many other friends have listened patiently to our first clumsy versions of proofs. We thank them all.

Bibiliography.

[Ab] Abrahmson, Karl, Modal Logic of Concurrent Nondeterministic Pograms, *Proc. Conf. on Parallel Programs*, Evian, France, 1979, 21-33.

[AHU] Aho A.V., J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[BKMRS] Banachowski, L., A. Kreczmar, G. Mirkowska, H. Rasiowa, A. Salwicki., An Introduction to Algorithmic Logic; Metamathematical Investigations in the Theory of Programs, In *Math. Found. of Comp. Sci.* (eds. Mazurkiewicz and Pawlak), Banach Center Publications, Warsaw. 1977.

[Be] Berman, F., A completeness technique for D-axiomatizable semantics, *Proc. 11th ACM Symp. on Theory of Comp.* (May 1979), 160-166.

[BP1] Ben-Ari, M., and Pnueli, A., Finite Models for Deterministic Propositional Dynamic Logic, Typescript, Dec. 1979.

[BP2] Berman, F. and M. Paterson., Test-Free Propositional Dynamic Logic is Strictly Weaker than PDL, T.R. 77-10-02, Dept. of Computer Science, Univ. of Washington, Seattle. Nov. 1977.

[BS] Bell, J.S. and A.B. Slomson, *Models and Ultraproducts.* North Holland, Amsterdam, 1971.

[C] Conway, J.H. *Regular Algebra and Finite Machines.* Chapman-Hall, London, 1971.

[CHMP] Chandra, A., Halpern, J., Meyer, A., and Parikh, R., Equations between Regular Terms and an Application to Process Logic, to appear in the *Proceedings of STOC 1981.*

[CO] Constable, R.L. and M.J. O'Donnell. *A Programming Logic.* Winthrop, Cambridge, Mass., 1978.

[E] Engeler, E., Algorithmic properties of structures, *Math. Systems Theory. 1*, 183-195. 1967.

[FL] Fischer, M.J. and R.E. Ladner., Propositional Modal Logic of Programs, *Proc. 9th Ann. ACM Symp. on Theory of Computing*, 286-294, Boulder, Col., May 1977.

[F] Floyd, R.W., Assigning Meanings to Programs, In *Mathematical Aspects of Computer Science* (ed. J.T. Schwartz), 19-32, 1967.

[G] Gabbay, D., Axiomatizations of Logics of Programs, Typescript, Nov. 1977.

[GPSS] Gabbay G., Pnueli, A. Shelah, S. and Stavi, J., The Temporal Analysis of Fairness, *7th Annual ACM Symposium on POPL*, Jan 1980, 163-173.

[GM] Greif, I. and A. R. Meyer, Specifying Programming Language Semantics, *Proc. 6th Ann. ACM Symp. on Principles of Programming Languages*, 180-189, San Antonio, Texas, Jan. 1979.

[Hos] Halmos, P.R. *Lectures on Boolean Algebras.* Springer Verlag, New York, 1974.

[H1] Harel, D., Two Results on Process Logic, *Information Processing Letters*, 8 (1979) 195-198.

[H2] Harel, D. First Order Dynamic Logic, *Lecture Notes in Computer Science*, no. 68, Springer Verlag 1979.

[Hal] Halpern, J., DPDL: Models, Complexity, Completeness, Typescript, March 1980.

[HKP] Harel, D., Kozen, D., Parikh, R., Process Logic: Expressiveness, Decidability, Completeness, *Research Report*, April 1980.

[Ho] Hoare, C.A.R., An Axiomatic Basis for Computer Programming, CACM 12, 576-580, 1969.

[Ka] Kamp, H., Tense Logic and the Theory of Linear Order, Ph.D. thesis, UCLA 1968.

[Ke] Keisler, H.J. *Model Theory for Infinitary Logic.* North Holland, Amsterdam, 1971.

[K1] Kozen, D., A representation theorem for models of *-free PDL, *Proc. 7th Int. Colloq. on Automata, Languages, and Programming*, Lecture Notes in Computer Science 85, ed. Goos and Hartmanis, Springer-Verlag, Berlin, 1980, 351-362. Also Report RC7864, IBM Research, Yorktown Heights, New York, Sept. 1979.

[K2] Kozen, D., On the duality of dynamic algebras and Kripke models, Report RC7893, IBM Research, Yorktown Heights, New York, Oct. 1979.

[K3] Kozen, D., On the representation of dynamic algebras, Report RC7898, IBM Research, Yorktown Heights, New York, Oct. 1979.

[K4] Kozen, D., On the representation of dynamic algebras II, Report RC8290, IBM Research, Yorktown Heights, New York, May 1980.

[K5] Kozen, D., On induction vs. *-continuity, Report RC8468, IBM Research, Yorktown Heights, New York, Sept. 1980.

[K6] Kozen, D., Semantics of Probabilistic Programs, *20th IEEE-FOCS Symposium*, San Juan, 1979, 101-114.

[KP] Kozen, D. and Parikh, R., An elementary Completeness Proof for PDL, to appear in *TCS.*

[Kr] Kripke, S., Semantical considerations on Modal Logic, *Acta Philosophica Fennica*, 83-94, 1963.

[LP] Litvinchouk, S.D. and V.R. Pratt., A Proof-checker for Dynamic Logic, *Proc. 5th Int. Joint Conf. on AI*, 552-558, Boston, Aug. 1977.

[Ma] Manna, Z., *Mathematical Theory of Computation.* McGraw-Hill, 1974.

[Me] Meyer, A., WS1S is not elementary decidable, *Proceedings of the Boston Logic Colloquium*, edited by R. Parikh, Springer Lecture Notes in Mathematics 1974.

[Mi] Mirkowska, G., Complete Axiomatisation of Algorithmic Properties of Program Schemes with Bounded Nondeterministic Interpretations, *12th Annual ACM-STOC Symposium* (1980), 14-21.

[N] Nemeti, I., Every free algebra in the variety generated by the representable dynamic algebras is separable and representable, manuscript, Hungarian Academy of Sciences, Budapest, 1980.

[Ni] Nishimura, H. Descriptively Complete Process Logic, typescript, Kyoto University, Dec. 1979.

[OP] Owicki, S., A Consistent and Complete Deductive System for the Verification of Parallel Programs, *Proc. 8th Ann. ACM Symp. on Theory of Computing*, 73-86. Hershey PA. May 1976.

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

[Pa1] Parikh, R., A Completeness Result for PDL, Manuscript dated 11/29/77.

[Pa2] Parikh, R., A Completeness Result for PDL, *Symposium on Mathematical Foundations of Computer Science*, Zakopane, Poland, Sept. 1978.

[Pa3] Parikh, R., Second Order Process Logic, *19th IEEE Symposium on Foundations of Computer Science.* Oct. 1978.

[Pa4] Parikh, R., Propositional Logics of Programs, *Proc. 7th Ann. ACM Symp. on Principles of Programming Languages*, Las Vegas, Jan. 1980.

[Pa5] Parikh, R., Some Applications of Topology to Program Semantics, to appear in *Math. Systems theory.*

[Pr1] Pratt, V.R., Semantical Considerations on Floyd-Hoare Logic, *Proc. 17th Ann. IEEE Symp. on Foundations of Comp. Sci.*, 109-121. Oct. 1976.

[Pr2] Pratt, V.R., A Practical Decision Method for Propositional Dynamic Logic, *Proc. 10th Annual ACM Symposium on Theory of Computing*, 326-337, San Diego, Calif., May 1978.

[Pr3] Pratt, V.R., Process Logic, *Proc. 6th Ann. ACM Symp. on Principles of Programming Languages*, Jan. 1979.

[Pr4] Pratt, V.R., Models of Program Logics, *20th IEEE-FOCS Symposium*, San Juan, 1979, 115-122.

[Pr5] Pratt, V.R., Dynamic Algebras: Examples, Constructions, Applications, MIT/LCS/TM-138, July 1979.

[Pr6] Pratt, V.R., Dynamic Logic, *Proc. 6th International Congress for Logic, Philosophy, and Methodology of Science*, Hanover, W. Germany, Aug. 1979.

[Pr7] Pratt, V.R., Dynamic algebras and the nature of induction, Proc. 12th ACM Symp. on Theory of Computing (May 1980), 22-28.

[R] Redko, V.N., On Defining Relations for the Algebra of Regular Events (Russian) *Ukrain. Math. Z.* 16 (1964), 120-126.

[Re] Reif, J., Logics for Probabilistic Computation, *12th Annual ACM-STOC Symposium* (1980), 8-13.

[Ra] Rabin, M., Decidability of Second Order Theories and Automata on Infinite Trees, *Transactions* of *the Amer. Math. Soc.*, 141 (1969) 1-35.

[RT1] Reiterman, J. and V. Trnkova, Dynamic algebras which are not Kripke structures, *Proc. 9th Symp. on Math. Found. of Computer Science* (Aug. 1980), 528-538.

[RT2] Reiterman, J. and V. Trnkova, private communication.

[Sa1] Salomaa, A., Two Complete Axiom Systems for the Algebra of Regular Events, *Jour. ACM* 13 (1966), 158-169.