Reprinted from JOURNAL OF COMPUTER AND SYSTEM SCIENCES

Vol. 25, No. 2, October 1982
Printed in Belgium

Process Logic: Expressiveness, Decidability, Completeness

DAVID HAREL* AND DEXTER KOZEN

IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598

AND

ROHIT PARIKH

Department of Mathematics, Boston University, Boston, Massachusetts 02215 and Laboratory for Computer Science, MIT, Cambridge, Massachusetts 02139

Received May 4, 1981; revised February 24, 1982

A process logic (PL) is defined that subsumes Pratt's process logic, Parikh's SOAPL, Nishimura's process logic, and Pnueli's Temporal Logic in expressiveness. The language of PL is an extension of the language of Propositional Dynamic Logic (PDL). A deductive system for PL is given which includes the Segerberg axioms for PDL and it is proved that it is complete. It is also shown that PL is decidable.

1. INTRODUCTION

The introduction of modal logic to program specification and verification has significantly aided our understanding of the process of reasoning about programs. Although the trappings vary, Algorithmic Logic, Dynamic Logic, and Temporal Logic all share common principles rooted in modal logic. These principles are becoming recognized as the appropriate vehicle for expressing many properties of a dynamic nature (see, e.g., Meyer [17]).

Dynamic Logic (DL), introduced by Pratt [8, 25], its propositional counterpart PDL, introduced by Fischer and Ladner [5], and Algorithmic Logic (Salwicki [2, 32]), reflecting their Floyd–Hoare heritage, deal quite successfully with the *before–after* behavior of programs. It is a major limitation, however, that these logics cannot deal with the *progressive* behavior of programs; e.g., DL is unsuited for assertions like, *variable x assumes value* 0 *at some point during the computation.*

Accordingly, various *process logics* have been developed to handle this more difficult task, mostly at the propositional level. Pratt [27] suggested using two process connectives $\perp$ (*during*) and $\lrcorner\!\ulcorner$ (*preserves*), besides the usual DL connectives. He presented lists of axioms and rules for these constructs and proved some partial

* Current address: The Weizmann Institute, Rehovot, Israel.

0022-0000/82/050144-27$02.00/0

completeness results. Subsequently, Parikh [22] defined the language SOAPL and showed that the validity problem for SOAPL was decidable. He also showed that SOAPL is at least as powerful as the language of Pratt, and Harel [9] later showed that it is strictly more powerful. The syntax of SOAPL, as defined in [22], was quite complex, and in particular, did not seem to give rise to a clean set of axioms which might serve as a basis for a completeness result. Indeed, one goal at that point seemed to be proving a completeness theorem for a decidable process language with reasonable expressive power (say, at least as powerful as SOAPL).

Independently, Pnueli [23, 24] was developing the Temporal Logic of Programs (TL), in which assertions about the progressive behavior of a computation can be made. In particular, TL can express freedom from deadlock, liveness, and mutual exclusion [7, 16]. One limitation of TL is that it provides no means for naming programs: the universe of discourse consists of a single, fixed program (in the parlance of [23], *TL* is *endogenous*). Although TL can discuss the synthesis of complex programs from simpler ones to some extent using *at* predicates, it is, in general, difficult to do so, since the logic is not tailored for this purpose. Recently, Nishimura [20] suggested combining PDL and TL by attaching a program α to a TL formula X, the resulting formula asserting that X holds in all computations of α. Nishimura showed that his language (call it, NL) is at least as powerful as SOAPL. He also argued that NL is in some sense as powerful as one could reasonably expect, by the result of Kamp [10] later refined by Gabbay *et al.* [7], that TL is precisely as strong in expressive power as the first-order theory of linear order with a first element. The primitives of Nishimura's system however, are still too intricate to yield a complete deductive system. One problem is, like SOAPL, that it has two kinds of formulas, *state* and *path formulas*. While this dichotomy may appear semantically natural, it leads to problems in constructing an axiom system. In PL, we avoid this dichotomy, leading to a substantial simplification.

Adopting the basic motivation of Nishimura, we define a process logic PL, which like NL can also be viewed as a fusion of TL and PDL. PL is simpler than NL in that all formulas are path formulas. Nevertheless, PL is as expressive as any of the logics PDL, TL, or NL. Moreover, PL is defined in such a way that it is a direct extension of PDL, both intuitively and formally. An appealing consequence of this is that all truths of PDL automatically hold in PL. We give an axiom system for PL, extending the Segerberg axiomatization of PDL, and prove it complete. The completeness proof is an extended version of the completeness proof of [15] for PDL. We also show that PL is dedidable, but we do not know whether it is elementary, in contrast to PDL, which is known to be decidable in exponential time ([28], see also [5]).

Section 2 contains the definition of PL and examples of its expressive power. In particular, PL is at least as powerful as any of the previously mentioned process logics. In Section 3 it is shown that the validity problem for PL is decidable. Section 4 contains the definition of a deductive system for PL and preliminary technical results in preparation for the completeness theorem, which is proved in Section 5. Section 6 indicates some directions for further work.

2. Definition of PL

Basic Concepts

Before defining the syntax and semantics of PL formally, we shall start with a brief intuitive outline. We assume familiarity with the syntax and semantics of PDL (see, e.g., [5]).

PL is interpreted over *path models*. Like a Kripke model of PDL, a path model is built upon a set of *states*, but in addition we may talk about *paths* of states. A path is just a finite or countable sequence of states, repetitions allowed. All formulas in the language of PL are *path formulas*; a formula X is either true or false in path p. This is in contrast to NL, SOAPL, and TL, which have both path and state formulas, and PDL, which has only state formulas.

States will be denoted $s, t,...$, and paths will be denoted $p, q, r,...$. A path is of length k if it consists of $k+1$ states. We identify a state with the 0-length path consisting of that state alone. The first and last states of a path p are denoted $\mathit{first}(p)$ and $\mathit{last}(p)$, respectively ($\mathit{last}(p)$ does not exist for infinite paths). If p, q are two paths such that $\mathit{last}(p) = \mathit{first}(q)$, then pq denotes the fusion of p and q. For example, if $p = s_1 s_2 s_3$ and $q = s_3 s_4 s_5$, then $pq = s_1 s_2 s_3 s_4 s_5$. If $\mathit{last}(p) \neq \mathit{first}(q)$, then pq *is undefined.*

A path q is a *suffix* of p if there exists an r such that $p = rq$. A suffix of p is *proper* if it is not equal to p. If p is not of length 0, then it has a longest proper suffix, denoted $\mathit{next}(p)$. Prefixes and proper prefixes are defined similarly.

Syntax

The language of PL is the language of PDL [5, 15] augmented with two additional operators **f** and **suf**. The operator **f** is applied to a formula X to yield a new formula $\mathbf{f}X$, meant to express the condition that X holds in the unique initial prefix of length 0. The binary operator **suf** corresponds to the U (until) operator of TL. There, if X and Y are state properties, then XUY expresses the path property that there exists a state s along the path satisfying Y such that all states occurring on the path before s satisfy X. The definition of **suf** in PL will be the same, only amended to account for the fact that PL has only path formulas and no state formulas. It is known [7, 10] that all purely temporal connectives are expressible from the U operator of TL, and hence also from the **suf** operator of PL.

Formally, PL has two sorts, *programs* $\alpha, \beta,...$, and *propositions* $X, Y,...$. It has primitive program letters $a, b,...$, primitive proposition letters $P, Q,...$, operators $\cup$, ;, and * which operate on programs, and operators $\vee$, $\neg$, **f**, and **suf** which operate on propositions. Compound propositions and programs are built up from the primitive letters using these operators according to the following rules:

if α, β are programs, then so are $\alpha ; \beta$, $\alpha \cup \beta$, and α^*,

if X, Y are propositions, then so are $X \vee Y$, $\neg X$, $\mathbf{f}X$, and X **suf** Y

if α is a program and X is a proposition, then $\langle \alpha \rangle X$ is a proposition.

We use the abbreviations $\alpha\beta$, $X \wedge Y$, $X \supset Y$, and $X \approx Y$ for $\alpha ; \beta$, $\neg(\neg X \vee \neg Y)$, $\neg X \vee Y$, and $(X \supset Y) \wedge (Y \supset X)$, respectively.

Semantics

A *path model* is a triple $M=(S, \models, R)$, where S is a set of *states*, $\models$ is a *satisfiability relation* for primitive propositions, and R is a relation assigning sets of paths to primitive programs. A path satisfies primitive proposition P iff its first state does. We write $p \models P$ if path p satisfies primitive proposition P, and we write $p \in R_a$ if p is a member of the set of paths assigned by R to primitive program a.

Relations R and $\models$ are extended to compound propositions and programs according to the following rules:

$$R_{\alpha\beta} = R_\alpha R_\beta = \{pq \mid p \in R_\alpha \text{ and } q \in R_\beta\},$$

$$R_{\alpha\cup\beta} = R_\alpha \cup R_\beta,$$

$$R_{\alpha^*} = \bigcup_{i<\omega} R_{\alpha^i},$$

$$p \models X \vee Y \quad \text{if} \quad p \models X \quad \text{or} \quad p \models Y,$$

$$p \models \neg X \quad \text{iff not} \quad p \models X,$$

$$p \models \langle a \rangle X \quad \text{iff} \quad \exists q \in R_\alpha \quad pq \models X,$$

$$p \models \mathbf{f}X \quad \text{iff} \quad \mathit{first}(p) \models X,$$

$$p \models X \,\mathbf{suf}\, Y \quad \text{iff} \quad \exists q \quad \text{such that}$$

(i) q is a proper suffix of p and $q \models Y$, and

(ii) $\forall r$, if r is a proper suffix of p and q is a proper suffix of r, then $r \models X$.

In the definition of **suf**, all suffixes under consideration are proper. This allows us to define an important operator **n** (for *next*). The formula $\mathbf{n}X$ says that a maximal proper suffix exists and satisfies X.

DEFINITION 2.1. $\mathbf{n}X = 0\,\mathbf{suf}X$.

Thus p satisfies $\mathbf{n}X$ if there exists a proper suffix q satisfying X and all intervening suffixes satisfy 0 (false), i.e., there are no intervening suffixes; therefore, q must be $\mathit{next}(p)$. In other words,

$$p \models \mathbf{n}X \quad \text{iff} \quad \mathit{next}(p) \models X.$$

We now define several other useful operators in terms of the primitives **f** and **suf**, and briefly describe their intended meaning. In the PL formulas below, unary operators have precedence over binary operators, but otherwise ambiguity is resolved by parentheses. *True* and *false* are denoted 1 and 0. The notation $\mathbf{n}^k X$ stands for $\mathbf{nn} \cdots \mathbf{n}X$ with k appearances of **n**.

DEFINITION 2.2.

$$\bar{\mathbf{n}}X = \neg\mathbf{n}\neg X$$
$$\mathbf{L}_0 = \bar{\mathbf{n}}0$$
$$\mathbf{L}_k = \mathbf{n}^k\bar{\mathbf{n}}0$$
$$\mathbf{some}X = X \vee (1\,\mathbf{suf}X)$$
$$\mathbf{all}X = \neg\mathbf{some}\neg X$$
$$\mathbf{last}X = \mathbf{some}(X \wedge \mathbf{L}_0)$$
$$\mathbf{fin} = \mathbf{last}\,1$$
$$\mathbf{inf} = \neg\mathbf{fin}$$
$$S(X_0,..., X_k; Y) = \mathbf{f}X_0 \wedge \mathbf{nf}X_1 \wedge \cdots \wedge \mathbf{n}^k\mathbf{f}X_k \wedge \mathbf{n}^k\mathbf{n}Y$$
$$S(X_0,..., X_k) = \mathbf{f}X_0 \wedge \mathbf{nf}X_1 \wedge \cdots \wedge \mathbf{n}^k\mathbf{f}X_k \wedge \mathbf{n}^k\bar{\mathbf{n}}0.$$

The operators $\bar{\mathbf{n}}$ and $\mathbf{n}$ are duals. Whereas $\mathbf{n}$ says, "there exists a proper suffix, and the longest one satisfies X," its dual $\bar{\mathbf{n}}$ says, "*if* there exists a proper suffix, then the longest one satisfies X." Thus $\mathbf{n}$ is existential in nature and $\bar{\mathbf{n}}$ is universal. These operators are related to the **nexttime** operator of Temporal Logic.

The operator $\mathbf{L}_0$ (for *length* 0) says that if there exists a proper suffix, then it satisfies 0; i.e., there does not exist a proper suffix. This says that the path is of length 0. Similarly, $\mathbf{L}_k$ says that the path is of length k.

The operator some applied to X says that there exists a suffix satisfying X, and $\mathbf{all}X$ says that all suffixes satisfy X. $\mathbf{last}X$ says that there is a last state in the path, i.e., the path is of finite length, and the last state satisfies X. **fin** (**inf**) says that the path is of finite (infinite) length.

$S(X_0,..., X_k; Y)$ says that the ith of the first $k + 1$ states of the path satisfies X_i and the remainder of the path satisfies Y. $S(X_0,..., X_k)$ says that the path is of length k and the ith state satisfies X_i.

The constructs of other process logics can be expressed in PL. For example, Pratt's [27] $\alpha \perp X$ and $\alpha \mathbin{\lrcorner\!\ulcorner} X$ are expressed as

$$[\alpha](\mathbf{some}X) \qquad \text{and}$$
$$[\alpha](\mathbf{all} \neg X \vee \mathbf{all}X \vee (\neg X \wedge ((\neg X)\mathbf{suf}(\mathbf{all}X)))),$$

respectively. Nishimura's αX is expressed as $\mathbf{f}([\alpha]X)$, and $[\alpha]X$ of PDL as $[\alpha](\mathbf{fin} \supset \mathbf{last}X)$. Also, using results of [7, 10], program-free formulas of PL are simply a notational variant of TL [23, 24].

Several properties relevant to real programming systems can be expressed, such as freedom from deadlock, liveness, and mutual exclusion [7, 16].

3. The Decision Problem for PL

In this section we shall prove that the validity problem for PL is decidable. This is done by encoding PL into *SnS*, the second order theory of n successors, shown decidable in Rabin [31]. Parikh used the same technique to show that SOAPL is decidable [22].

To be slightly more precise, the decidability of PL will follow from a lemma stating that for any formula X of PL, there is a formula X' of the language of *SnS*, for some appropriate n depending on X, such that X is satisfiable iff X' is true in the standard model of *SnS*. The key observation leading to this lemma is that X is satisfiable iff it is true in a tree-like model in which states do not repeat.

Let X be an arbitrary formula of PL involving primitive programs $a_1,..., a_n$ and propositional letters $P_1,..., P_m$. We shall be using the second order language L_{n+3} of $n+3$ successor functions $a_1,..., a_n$, b, c, d, where successor a applied to formula x will be written xa. The language allows individual variables $x, y,...$, and set variables $A, B,...$. Quantification over both types of variables is permitted and the language includes Boolean connectives and a membership relation $x \in A$. For the standard semantics and details the reader is referred to Rabin [31]. The set of sentences of L_n true in the standard semantics is called the *second order theory of n successors*, and is denoted *SnS*.

Our main goal is to construct a formula X' of L_{n+3} which is in $S(n+3)S$ iff there is a model M and path p in M such that $p \models X$. The formula X' will be of the form

$$\exists U, B_1,..., B_n, C_1,..., C_m, A(\mathbf{model}(U, B_1,..., B_n, C_1,..., C_m) \wedge \mathbf{path}(A) \wedge \Phi_X(A)). \qquad (*)$$

Here, U will encode the state space S of M, B_i the interpretation in M of primitive program a_i, C_j that of proposition letter P_j, and A the path p. The subformulas **model** and **path** will guarantee this kind of behavior. The formula $\Phi_X(A)$ which, in general, will involve the B's and C's, will have a free variable A and will be defined by induction on the structure of X.

Let $M = (S, \models, R)$ be a model; without loss of generality we can assume that S is countable. Let $\Sigma = \{a_1,..., a_n, b, c, d\}$, where b, c, and d are new symbols, used in the encoding into $S(n+3)S$. It is possible to find a nonempty set (possibly finite) $T = \{s_0, s_1,...,\} \subseteq S$ such that any $s \in S$ is on some path $p \in R_\alpha$ for some $\alpha \in (\bigcup a_i)^*$ and $\mathit{first}(p) \in T$. In other words, every state in S is accessible by programs from states in T. Without loss of generality, assume that T is minimal with this property. Certainly, for any $s \in S$ there are at most countably many paths in R_{a_i} starting at s; assume they are numbered. Now define a partial function $\delta: \Sigma^* \to S$ as follows:

$$\delta(b^i) = s_i \qquad \text{for } i \geqslant 0 \text{ such that } s_i \in T \text{ (in particular } \delta(\lambda) = s_0)$$

$$\delta(xa_i d^j c^k) = \quad \text{the}(k+1)\text{th state on the } j\text{th path of } R_{a_i}$$

$$\text{which starts at } \delta(x); \quad \text{for } 1 \leqslant i \leqslant n,\ j > 0,\ k \geqslant 0.$$

In particular, $\delta(xa_i d^j) = \delta(x)$. We write $\delta(x)\downarrow = s$ if $\delta(x)$ is defined and equal to s. Set $S' = \{x \mid \delta(x) \text{ is defined}\}$. Now define a partial function $\gamma: 2^{\Sigma^*} \to S^* \cup S^\omega$ as follows: $\gamma(A)$ is defined iff $A = \{x_1, x_2, \ldots,\}$ (possibly finite), and for all i, $x_i \in S'$ and $x_i < x_{i+1}$ (where $x < y$ iff x is a proper prefix of y) and for no $y \in S'$ is it the case that $x_i < y < x_{i+1}$. We then set $\gamma(A) = (\delta(x_1), \delta(x_2), \ldots,)$.

EXAMPLE. Let $R_{a_1} = \{(s_0, s, s')\}$, $R_{a_2} = \{(s', s_0)\}$; take $T = \{s_0\}$. Then $\delta(\lambda) = s_0$, $\delta(a_1 dc) = s$, and $\delta(a_1 dcc) = s'$. Also, if $x = (a_1 dcca_2 dc)^i$ for any i, then $\delta(x) = s_0$, $\delta(xa_1 dc) = s$, and $\delta(xa_1 dcc) = s'$. If $A_0 = \{a_1 dc, a_1 dcc, a_1 dcca_2 dc, a_1 dcca_2 dca_1 dc\}$, then $\delta(A_0)\downarrow = (s, s', s_0, s)$.

Given the model M define sets B_i^M, $1 \leqslant i \leqslant n$ and C_j^M, $1 \leqslant j \leqslant m$, encoding the meanings of a_i and P_j as follows:

Let $B_i^M = \bigcup A$-{first element of A}, where the union is over all sets A such that $\gamma(A)\downarrow \in R_{a_i}$, and the first element of A is the unique x_1 such that $A = \{x_1, x_2, \ldots,\}$ in the definition of $\gamma(A)$ above.

In the case of A_0 in the example above we have simply $B_i^M = \bigcup_x \{x, xa_1 dc, xa_1 dcc\}$, for all $x = (a_1 dcca_2 dc)^i$, $i \leqslant 0$.

Let $C_j^M = \{x \mid x \in S', \delta(x) \models P_j\}$. Before defining the formulas **model, path** and $\Phi_X(A)$, we establish some abbreviations for concepts definable in L_{n+3}.

(1) $x \leqslant y$ (x is an initial segment of y):

$$\forall A(x \in A \wedge \forall z((z \in A \supset \bigwedge_i za_i \in A \wedge \bigwedge_{v \in \{b,c,d\}} zv \in A) \supset y \in A)).$$

(2) $\mathbf{linor}(A)$ (A is linearly ordered by $\leqslant$):

$$\forall x \forall y(x \in A \wedge y \in A \supset (x \leqslant y \vee y \leqslant x)).$$

(3) $\mathbf{first}(A, x)$ (x is the first element of linearly ordered A):

$$\mathbf{linor}(A) \wedge \forall y(y \in A \supset x \leqslant y) \wedge x \in A.$$

(4) $\mathbf{suffix}(A, B)$ (the linearly ordered B is a proper suffix of the linearly ordered A):

$$\mathbf{linor}(A) \wedge \mathbf{linor}(B) \wedge \exists x(x \in A \wedge \neg\mathbf{first}(A, x)$$
$$\wedge \forall y((x \leqslant y \wedge y \in A) \equiv y \in B)).$$

(5) $\mathbf{single}(A, x)$ ($A = \{x\}$):

$$x \in A \wedge \forall y(y \in A \supset x = y).$$

(6) $\mathbf{pair}(A, x, y)$ ($A = \{x, y\}$):

$$x \in A \wedge y \in A \wedge \forall z(z \in A \supset (z = x \vee z = y)).$$

(7) **conc**(A, B, C) (A, B, and C are linearly ordered, and either A is infinite and $C = A$, or A is finite and its last element is the first in B and C represents the concatenation of A and B):

$$\begin{aligned}&\mathbf{linor}(A) \wedge \mathbf{linor}(B) \wedge \mathbf{linor}(C) \wedge \exists x(x \in A)\\ &\quad \wedge (\forall x(x \in A \supset \exists y(y \geqslant x \wedge y \in A \wedge \neg y = x) \wedge \forall x(x \in A \equiv x \in C))\\ &\quad \vee (\exists x(x \in A \wedge \forall y(y \in A \supset y \leqslant x) \wedge \mathbf{first}(B, x)\\ &\quad \wedge \forall z \forall y(x \in A \wedge y \in B \supset x \leqslant y)\\ &\quad \wedge \forall z(z \in C \equiv (z \in A \vee z \in B)))).\end{aligned}$$

(8) **next**(A, x, y) (x and y are consecutive in linearly ordered A):

$$\mathbf{linor}(A) \wedge x \in A \wedge y \in A \wedge \forall z(z \in A \supset ((x \neq z \wedge x \leqslant z) \equiv y \leqslant z)).$$

(9) **segment**(A, B, x, y) (B is the subset of the linearly ordered A enclosed by x and y):

$$\mathbf{linor}(A) \wedge x \in A \wedge y \in A \wedge \forall z(z \in B \equiv (z \in A \wedge x \leqslant z \leqslant y)).$$

(10) **rest**(A, B, x) (B is the infinite suffix of the linearly ordered A starting at x):

$$\begin{aligned}&\mathbf{linor}(A) \wedge x \in A, \wedge \forall y(y \in A \supset \exists z(z \in A \wedge z \geqslant y \wedge \neg z = y))\\ &\quad \wedge \forall y(y \in A \wedge y \geqslant x \equiv y \in B).\end{aligned}$$

(11) **a-extension**(x, y) (for $a \in \Sigma, x = ya^i, i \geqslant 0$):

$$x \geqslant y \wedge \forall A((x \in A \wedge \forall z(za \in A \supset z \in A)) \supset y \in A).$$

Now define

$$\begin{aligned}&\mathbf{path}(A)\colon\ \mathbf{linor}(A) \wedge \forall x \forall y \forall z((x \leqslant y \wedge y \leqslant z \wedge x \in A \wedge z \in A)\\ &\quad \wedge (y \in U \vee \exists w(w = \lambda \wedge b\text{-extension } (y, w)) \supset y \in A))\\ &\quad \wedge \forall x(x \in A \supset (x \in U \vee \exists w(w = \lambda \wedge b\text{-extension}(x, w)))),\end{aligned}$$

where $w = \lambda$ is defined, e.g., by $\bigwedge \forall x(\neg w = xa)$, with the conjunction taken over all $a \in \Sigma$. In words, a path is a *dense* linearly ordered set A with elements either from the set U described below, or of the form b^i, $i \geqslant 0$.

Now define the formula **model**$(U, B_1,..., B_n, C_1,..., C_m)$ which will assert that the sets B_i encode atomic programs as B_i^M for some model M, and similarly for the C_j. The set U is asserted to include the elements in all the B_i. Thus, U together with the set $\{b^i \mid i \geqslant 0\}$, can encode the domain S of a model M. Define **model** as:

$$U \neq \varnothing \wedge U = \bigcup_{l<i<n} B_i$$
$$\wedge \bigwedge_{1<j<m} \forall x(x \in C_j \supset (x \in U \vee \exists w(w = \lambda \wedge b\text{-extension}(x, w))))$$
$$\wedge \bigwedge_{1<i<n} \forall x((xc \in B_i \supset x \in B_i) \wedge (xdd \in B_i \supset xd \in B_i)$$
$$\wedge (x \in B_i \supset \exists z(\textbf{start-state}_i(x, z)))),$$

where $\textbf{start-state}_i(x, z)$ is defined as:

$$\forall u(u = za_i d \supset \exists v(d\text{-extension}(v, u) \wedge \forall y(y = vc \supset c\text{-}extension(x, y))))$$
$$\wedge (z \in U \vee \exists w(w = \lambda \wedge b\text{-extension}(z, w))).$$

Thus, B_i is asserted to be *closed* under shorter paths of a_i and under less paths, and each $x \in B_i$ is asserted to be of the form $za_i d^j c^k$ for some $j > 0$, $k \geqslant 0$, where z is either in U or of the form b^j, $j \geqslant 0$.

Now, by induction on the complexity of the formula X of PL, we define the L_{n+3} formula $\Phi_X(A)$, depending on U and the B_i and C_j. One clause of the definition involves an L_{n+3} formula $\Psi_\alpha(A)$ for each program α in PL. These formulas are also defined below.

$$\Phi_{P_i}(A) = \forall x(\textbf{first}(A, x) \supset x \in C_i)$$
$$\Phi_{\neg X}(A) = \neg \Phi_X(A)$$
$$\Phi_{X \vee Y}(A) = \Phi_X(A) \vee \Phi_Y(A)$$
$$\Phi_{X \operatorname{suf} Y}(A) = \exists B\ (\textbf{suffix}(A, B) \wedge \Phi_Y(B)$$
$$\wedge \forall C\ (\textbf{suffix}(A, C) \wedge \textbf{suffix}(C, B) \supset \Phi_X(C)))$$
$$\Phi_{fX}(A) = \forall B\ \forall x\ (\textbf{single}(B, x)$$
$$\wedge \textbf{first}(A, x) \supset \Phi_X(B))$$
$$\Phi_{\langle\alpha\rangle X}(A) = \exists B(\textbf{path}(B) \wedge \Psi_\alpha(B) \wedge \forall x(\textbf{first}(B, x) \supset (x \in A$$
$$\wedge \forall y(y \in A \supset y \leqslant x))$$
$$\wedge \forall C\ (\textbf{conc}(A, B, C) \supset \Phi_X(C))).$$

Now, define $\Psi_\alpha(A)$ by induction on α for PL programs:

$$\Psi_{a_i}(A) = \exists z \exists B(\neg z \in B \wedge \textbf{linor}(B) \wedge B \subset B_i \wedge \forall x(x \in A \equiv (x = z \vee x \in B))$$
$$\wedge \textbf{first}(A, z) \wedge \forall x((x \in B \wedge xc \in B_i) \supset xc \in B$$
$$\wedge (xc \in B \wedge x \in B_i) \supset x \in B)),$$
$$\Psi_{\alpha \cup \beta}(A) = \Psi_\alpha(A) \vee \Psi_\beta(A),$$
$$\Psi_{\alpha\beta}(A) = \exists B\ \exists C\ (\textbf{conc}(B, C, A) \wedge \Psi_\alpha(B) \wedge \Psi_\beta(C)),$$

$$\Psi_{\alpha^*}(A) = \exists B(\forall x(x \in B \supset x \in A) \wedge \mathbf{linor}(B) \wedge \forall x\, \forall y\, (\mathbf{next}(B, x, y)$$
$$\supset \forall C\, (\mathbf{segment}(A, C\, x, y) \supset \Psi_\alpha(C))) \wedge \exists x\, (x \in B \wedge \mathbf{first}(A, x))$$
$$\wedge \forall x\, (x \in A \wedge \forall y\, (x \leqslant y \supset \neg(y \in A)) \supset x \in B)$$
$$\wedge \forall x\, \forall C\, (\mathbf{rest}(A, C, x) \wedge x \in B$$
$$\wedge \forall y\, (y \in B \supset y \leqslant x) \supset \Psi_\alpha(C)).$$

LEMMA 3.1. *Let $U \subseteq \Sigma^*$. For each $A \subseteq U$, $\gamma(A)$ is defined iff* **path**(A) *is true.*

Proof. Immediate. ∎

Now let M be given. We can talk about the truth in $S(n+3)\,S$ of formulas involving U, B_i, C_j by adopting the convention that they are taken, respectively, to be $U^M = \bigcup_i B_i^M$, B_i^M and C_j^M. With this convention we have Lemmas 3.2 and 3.3 which are proven tediously but easily by induction on the structure of α and X, respectively. We provide proofs for one case in each lemma.

LEMMA 3.2. *For every program α, and for every path p in M,*

(1) *for every $A \subseteq \Sigma^*$, if $\gamma(A)\!\downarrow\, = p$ and $\Psi_\alpha(A)$ holds, then $p \in R_\alpha$.*

(2) *if $p \in R_\alpha$, then for every x such that $\delta(x)\!\downarrow\, =$* **first**$(p)$, *there exists $A \subseteq \Sigma^*$ such that $\gamma(A)\!\downarrow\, = p$, $x \in A$, and $\Psi_\alpha(A)$ holds.*

Proof. We prove the lemma for the case $\alpha\beta$.

(1) Let $\gamma(A)\!\downarrow\, = p$ and $\Psi_{\alpha\beta}(A)$ hold. Then $\exists B, C$ s.t. $\Psi_\alpha(B)$ and $\Psi_\beta(C)$, and $A = BC$. By the definition of γ we have $\gamma(B)\!\downarrow\, = q$ and $\gamma(C)\!\downarrow\, = r$ for some q, r and $qr = p$. By the inductive hypothesis, part (1), we obtain $q \in R_\alpha$, $r \in R_\beta$ and hence $p = qr \in R_{\alpha\beta}$.

(2) Let $p \in R_{\alpha\beta}$, so that $p = qr$ with $q \in R_\alpha$ and $r \in R_\beta$. By the inductive hypothesis on α we have that for every x such that $\delta(x)\!\downarrow\, =$ **first**(q), there is $B \subseteq \Sigma^*$ such that $\gamma(B)\!\downarrow\, = q$, $x \in B$, and $\Psi_\alpha(B)$ holds. Now B must be finite. Let y be the largest element in B. Clearly, $\delta(y) =$ **last**$(q) =$ **first**(r). Apply the inductive hypothesis to β with $\delta(y)\!\downarrow\, =$ **first**(r) and $r \in R_\beta$, to obtain some $C \subseteq \Sigma^*$ such that $\gamma(C)\!\downarrow\, = r$, $y \in C$, and $\Psi_\beta(C)$ holds. So we have that for every x such that $\delta(x)\!\downarrow\, =$ **first**$(q) =$ **first**(p) there is $A \subseteq \Sigma^*$, namely, the set $A = B \cup C$, such that $x \in A$. In order to show that $\Psi_{\alpha\beta}(A)$ holds all we need to show is that **conc**(B, C, A) holds. But this follows from the facts that y is the largest element in B, $y \in C$, $\delta(y) =$ **first**$(r) =$ **last**(q) and $\gamma(B) = q$, $\gamma(C) = r$. ∎

LEMMA 3.3. *For every X and for every $A \subseteq \Sigma^*$, if $\gamma(A)\!\downarrow\, = p$ for some path p in M, then $p \models X$* iff *$\Phi_X(A)$ holds.*

Proof. We prove the lemma for the case $\langle\alpha\rangle X$. Assume $\gamma(A)\!\downarrow\, = p$.

(1) Let $p \models \langle \alpha \rangle X$. Then there is $q \in R_\alpha$ such that pq is defined and $pq \models X$. By Lemma 3.2(2) for each x such that $\delta(x)\downarrow = \mathbf{first}(q)$, there is $B \subseteq \Sigma^*$ such that $\gamma(B)\downarrow = q$, $x \in B$, and $\Psi_\alpha(B)$ holds. By Lemma 3.1 we also have that $\mathbf{path}(B)$ holds. By the inductive hypothesis, from $pq \models X$ we may infer that for any C such that $\gamma(C)\downarrow = pq$, we have $\Phi_X(C)$ holding. In particular, for any C such that $\mathbf{conc}(A, B, C)$ holds, we would have $\Phi_X(C)$ holding too. Now choose x to be the largest element in A, hence $\delta(x) = \mathbf{last}(p) = \mathbf{first}(q)$. The B existing by Lemma 3.2 is that chosen to satisfy $\Phi_{\langle \alpha \rangle X}(A)$.

(2) Let $\Phi_{\langle \alpha \rangle X}(A)$ hold. Then there is $B \subseteq \Sigma^*$ such that $\gamma(B)\downarrow = q$ for some q, $\Psi_\alpha(B)$ holds, and pq is defined. Furthermore, $C = A \cup B$ satisfies $\Phi_X(C)$. By Lemma 3.2(1) we have $q \in R_\alpha$. Since $\gamma(C)\downarrow = pq$, we conclude by the inductive hypothesis that $pq \models X$. Hence $p \models \langle \alpha \rangle X$. ∎

LEMMA 3.4. *Let X be any formula of* PL *involving atomic programs from among $\{a_1, \ldots, a_{n-1}\}$ and atomic propositions from among $\{P_1, \ldots, P_m\}$. Then X is satisfiable iff formula* $(*)$ *is in $S(n+3)S$.*

Proof. Note that a_n does not appear in X.

Only if: Let $M', p \models X$. Consider the model M obtained from M' by letting $R_{a_n} = \{p\}$. This ensures the existence of some A with $\gamma(A)\downarrow = p$. Clearly, since X does not refer to a_n, we have $M, p \models X$. If U, B_i and C_j are taken to be respectively, U^M, B_i^M, and C_j^M as defined earlier, and A is taken to be some set such that $\gamma(A)\downarrow = p$, then by Lemma 3.1, $\mathbf{path}(A)$ holds, by Lemma 3.3, $\Phi_X(A)$ holds, and the formula **model** holds by the construction of U^M, B_i^M, and C_j^M.

If: Let $(*)$ be a true sentence of L_{n+3}, and let U^0, B_i^0, C_j^0, and A be the sets asserted in $(*)$ to exist. Define the model $M = (S, \models, R)$ by:

$$S = U^0 \cup \{b^i \mid i \geqslant 0\}$$

$$u \models P_j \qquad \text{iff} \quad u \in C_j,$$

$$R_{a_i} = \{(z, za_i dcc, \ldots) \mid \text{all elements in } \mathbf{path} \text{ are in } B_i \text{ except possibly } z,$$
$$z \in S, \text{ and the sequence is either infinite or maximal}$$
$$\text{finite satisfying the conditions}\}.$$

It is easy to see that the formula **model** forces B_i^0 and C_j^0 to be B_i^M and C_j^M. By Lemma 3.3 we obtain that $M, p \models X$ for $p = \gamma(A)$.

By Rabin's [31] result that SnS is decidable, we conclude:

THEOREM 3.1. *The validity problem for* PL *is decidable.*

We do not know of an elementary bound on the complexity of deciding validity of PL. In fact, it is possible that Meyer's [17] result on the complexity of the first order theory of linear order could be used to show that PL is nonelementary.

4. A Deductive System for PL

The following proof system is purely equational. Some of the axiom schemas below are given in the form $X \supset Y$, but this is equivalent to the equation $(X \wedge Y) \approx X$. There is one rule of inference, namely substitution of equals for equals. We write $X \equiv Y$ if $\vdash X \approx Y$ and $X \leqslant Y$ if $\vdash X \supset Y$, where $\vdash$ denotes provability in this system. A formula X is *consistent* if not $X \equiv 0$.

Definition 4.1.

(1) All axioms of PDL (see [5, 15, 21]), including propositional logic.

(2) $\mathbf{f}(X \vee Y) \approx \mathbf{f}X \vee \mathbf{f}Y$.

(3) $\mathbf{f}\neg X \approx \neg \mathbf{f}X$.

(4) $(X\mathbf{suf}Y) \vee (X\mathbf{suf}Z) \approx X\mathbf{suf}(Y \vee Z)$.

(5) $(X\mathbf{suf}Y) \wedge (Z\mathbf{suf}W) \approx (X \wedge Z)\ \mathbf{suf}(Y \wedge W) \vee (X \wedge Z)\ \mathbf{suf}(X \wedge W \wedge (X\mathbf{suf}Y)) \vee (X \wedge Z)\ \mathbf{suf}(Y \wedge Z \wedge (Z\mathbf{suf}W))$.

(6) $\neg(X\mathbf{suf}Y) \approx \neg(1\mathbf{suf}Y) \vee (\neg Y)\ \mathbf{suf}(\neg X \wedge \neg Y)$.

(7) $X\mathbf{suf}Y \approx \mathbf{n}Y \vee \mathbf{n}(X \wedge (X\mathbf{suf}Y))$.

(8) $X\mathbf{suf}(X \wedge (X\mathbf{suf}Y)) \approx \mathbf{n}(X \wedge (X\mathbf{suf}Y))$.

(9) $X\mathbf{suf}(X \wedge (X\mathbf{suf}Y)) \approx X\mathbf{suf}(X \wedge \mathbf{n}Y)$.

(10) $\mathbf{fn}1 \approx 0$.

(11) $\neg X \wedge \mathbf{f}X \supset \mathbf{n}1$.

(12) $P \approx \mathbf{f}P$ for primitive propositions P.

(13) $\mathbf{f}X \wedge \langle\alpha\rangle Y \approx \langle\alpha\rangle(\mathbf{f} \wedge Y)$.

(14) $\mathbf{n}\langle\alpha\rangle X \approx \mathbf{n}1 \wedge \langle\alpha\rangle\, \bar{n}X$.

(15) $\langle\alpha\rangle\, 1 \supset \mathbf{fin}$.

(16) $(\mathbf{n}X \supset X)\ \mathbf{suf}X \supset \mathbf{n}X$.

Axiom 4.1(16) is called the *path induction axiom*. It is not hard to see that all the above axioms hold in all path models, therefore the system is sound.

The following are some elementary theorems of PL which follow easily from the axioms of Definition 4.1. After each one we indicate in parentheses the axioms and previous parts of the theorem used in the proof.

Theorem 4.2.

(1) $\mathbf{f}0 \approx 0$, $\mathbf{f}1 \approx 1$, $\mathbf{f}(X \wedge Y) \approx \mathbf{f}X \wedge \mathbf{f}Y$ (4.1(2), (3)).

(2) $(X\mathbf{suf}Z) \vee (Y\mathbf{suf}Z) \supset (X \vee Y)\ \mathbf{suf}Z$ (4.1(4), (5)).

(3) $(X\mathbf{suf}Z) \wedge (Y\mathbf{suf}Z) \approx (X \wedge Y)\ \mathbf{suf}Z$ (4.1(4), (5)).

(4) $1\mathbf{suf}0 \approx 0$ (4.1(6)).

(5) $X\mathbf{suf}0 \approx 0, \mathbf{n}0 \approx 0$ (4.2(2), (4)).

(6) $\mathbf{n}(X \vee Y) \approx \mathbf{n}X \vee \mathbf{n}Y$ (4.1(4)).

(7) $\mathbf{n}(X \wedge Y) \approx \mathbf{n}X \wedge \mathbf{n}Y$ (4.1(5)).

(8) $\mathbf{n}\neg X \approx \mathbf{n}1 \wedge \neg\mathbf{n}X$ (4.2(5)–(7)).

(9) $X\mathbf{suf}(X \wedge \mathbf{n}Y) \approx \mathbf{n}(X \wedge (X\mathbf{suf}Y))$ (4.1(8), (9)).

(10) $1\mathbf{suf}(\mathbf{n}X) \approx \mathbf{n}(1\mathbf{suf}X)$ (4.2(9)).

(11) $\mathbf{L}_0 \supset (X \approx \mathbf{f}X)$ (4.1(3), (11)).

(12) $\mathbf{f}X \approx \mathbf{f}(\mathbf{L}_0 \wedge X)$ (4.1(2), (10); 4.2(1)).

(13) $\langle\alpha\rangle \mathbf{L}_0 \supset \mathbf{L}_0$ (4.1(1), (14); 4.2(5)).

(14) $\mathbf{ff}X \approx \mathbf{f}X$ (4.2(11), (12)).

(15) $X\mathbf{suf}X \approx \mathbf{n}X$ (4.1(7); 4.2(6)).

(16) $(\neg X)\,\mathbf{suf}X \approx 1\mathbf{suf}X$ (4.1(6); 4.2(5)).

(17) $X\mathbf{suf}Y \approx (X \wedge \neg Y)\,\mathbf{suf}Y$(4.2(2), (3), (16)).

(18) $\langle\alpha\rangle(\mathbf{L}_0 \wedge X) \approx \mathbf{L}_0 \wedge X \wedge \langle\alpha\rangle \mathbf{L}_0$ (4.1(1), (13); 4.2(11), (13)).

(19) $\langle\alpha^*\rangle \mathbf{L}_0 \approx \mathbf{L}_0$ (4.1(1); 4.2(13)).

(20) $\mathbf{n}(\langle\alpha\rangle X) \approx \mathbf{n}1 \wedge \langle\alpha\rangle \mathbf{n}X$ (4.1(1), (14); 4.2(8), (13)).

(21) $\mathbf{L}_0 \wedge \langle\alpha^*\rangle \mathbf{n}X \approx \mathbf{L}_0 \wedge \langle\alpha\rangle \mathbf{n}\langle\alpha^*\rangle X$ (4.1(1); 4.2(19), (20)).

(22) $S(X_0,..., X_k, X_{k+1},..., X_m; Y) \approx S(X_0,..., X_k; S(X_{k+1},..., X_m; Y))$ (4.2(7)).

(23) $\mathbf{L}_k \wedge \langle\alpha\rangle \mathbf{n}^k Y \supset \mathbf{n}^k(\langle\alpha\rangle Y)$ (4.2(7), (20)).

(24) $\langle\alpha\rangle \mathbf{L}_k \supset \bigvee_{0 \leqslant i \leqslant k} \mathbf{L}_i$ (4.2(18), (20)).

The extended Fischer–Ladner Closure

The *extended Fischer–Ladner closure* of a formula W, denoted EFL(W) or just EFL if W is understood, is the set of formulas that are relevant to the semantics of W. EFL(W) is analogous to the *Fischer–Ladner closure* FL(W) used in [5, 15, 21, 26,] for PDL. Formally, FL(W) (or just FL) is defined to be the smallest set of formulas containing W and closed under the following rules:

if	$X \vee Y \in$ FL,	then	$X \in$ FL and $Y \in$ FL,
if	$\neg X \in$ FL,	then	$X \in$ FL,
if	$\langle\alpha\rangle X \in$ FL,	then	$X \in$ FL,
if	$\langle\alpha \cup \beta\rangle X \in$ FL,	then	$\langle\alpha\rangle X \in$ FL and $\langle\beta\rangle X \in$ FL,
if	$\langle\alpha\beta\rangle X \in$ FL,	then	$\langle\alpha\rangle\langle\beta\rangle X \in$ FL,
if	$\langle\alpha^*\rangle X \in$ FL,	then	$\langle\alpha\rangle X \in$ FL and $\langle\alpha\rangle\langle\alpha^*\rangle X \in$ FL.

Then FL(W) is finite [5], and is in fact linear in the size of W. The completeness of PDL was established in [15] by constructing, for a given consistent W, a model whose states were *atoms* of FL(W) (i.e., *consistent* conjunctions of elements of FL such that each $X \in$ FL or its negation appears in the conjunction). In order to obtain the completeness of PL we need to extend the concept of Fischer–Ladner closure. Let $X(x)$ be a formula with one free occurrence of variable x ranging over formulas of PL. Now $X(Y)$ denotes $X(x)$ with variable x replaced by formula Y. Then $X(Y)$ is a formula of PL. Define the relation $\rightarrow$ by: $W \rightarrow W'$ if there is a formula $X(x)$ such that $W = X(Y)$, $W' = X(Y')$, and Y' is a Boolean combination of subformulas of Y. $\rightarrow^*$ denotes the reflexive closure of $\rightarrow$.

DEFINITION 4.3. The *extended Fischer–Ladner closure* of W, denoted EFL(W), is the smallest set of formulas containing FL(W), **n1**, and **fin**, and closed under $\rightarrow$.

Although EFL(W) contains infinitely many formulas, we prove below that it is finite up to equivalence modulo $\equiv$, although the number of equivalence classes can be nonelementary in the size of W.

Define $L(X) = \{Y \mid X \rightarrow^* Y\}$. If L is a set of formulas of PL, define BC(L) to be the set of Boolean combinations of elements of L. Let $\oplus$ be any operator symbol in the language of PL, including the modal operators $\langle \alpha \rangle$. Operator $\oplus$ can be of any arity (including 0), but for simplicity of notation we shall write $\oplus$ as a binary operator.

LEMMA 4.4. $L(\oplus XY) = BC(L(X) \cup L(Y) \cup \{\oplus ZW \mid Z \in L(X),\ W \in L(Y)\})$.

Proof ($\supseteq$). Since $\oplus XY \rightarrow X$, we have that $L(X) \subseteq L(\oplus XY)$. Similarly $L(Y) \subseteq (\oplus XY)$. Since $\oplus XY \rightarrow^* \oplus ZW$ whenever $X \rightarrow^* Z$ and $Y \rightarrow^* W$, we have that

$$\{\oplus ZW \mid Z \in L(X),\ W \in L(Y)\} \subseteq L(\oplus XY).$$

Thus it remains to show $L(\oplus XY)$ is closed under BC. But any set $L(Z)$ is closed under BC, since if $B(Y_1, \ldots, Y_n)$ is any Boolean combination of the formulas Y_i, where $Z \rightarrow^* Y_i$, then $Z \rightarrow B(Z, \ldots, Z) \rightarrow^* B(Y_1, \ldots, Y_n)$.

($\subseteq$). Certainly

$$\oplus XY \in BC(L(X) \cup L(Y) \cup \{\oplus ZW \mid Z \in L(X),\ W \in L(Y)\}),$$

and any application of $\rightarrow$ to a Boolean combination of members of $L(X)$, $L(Y)$, and $\{\oplus ZW \mid Z \in L(X),\ W \in L(Y)\}$ results in a formula of the same form. Since $L(\oplus XY)$ is the least set containing $\oplus XY$ and closed under $\rightarrow$, the result follows. ∎

LEMMA 4.5. *Up to equivalence modulo* $\equiv$, EFL(W) *is finite.*

Proof. It suffices to show that $L(X)$ is finite modulo $\equiv$, where X is some formula containing all elements of FL(W), **fin**, and $\mathbf{L}_0$ as subformulas. We do this by induction on the structure of X. For primitive P, all elements of $L(P)$ are propositionally equivalent to P, $\neg P$, 0, or 1. For X of the form $\oplus YZ$, by induction hypothesis $L(Y)$ and $L(Z)$ are finite modulo $\equiv$, so by Lemma 4.4 and the fact that a finitely generated Boolean algebra is finite, $L(\oplus YZ)$ is finite as well. ∎

Let W be a consistent formula. An *atom* of EFL(W) is any $\leqslant$-minimal consistent element of EFL(W). The symbols $A, B, C, A_0, \ldots,$ will always be used to denote atoms of some EFL(W). By the previous lemma, any EFL(W) has only finitely many atoms up to $\equiv$-equivalence. Because EFL(W is closed under Boolean combinations, for any $X \in$ EFL(W) and atom A, either $A \leqslant X$ or $A \leqslant \neg X$, and any X in EFL(W) is equivalent to the join of all $A \leqslant X$. This says that there is at least one atom $A \leqslant W$, since W is consistent.

5. Completeness of PL

In this section we prove the completeness of the axiom system for PL given in the last section. We first assume the axiom **fin** and restrict our interpretations to paths of finite length only, and later indicate how to extend the result to the general case. We define a *special form* for formulas and show that each formula X is equivalent to infinitely many formulas in special form, called the *refinements* of X, and show that $X \equiv Y$ iff X and Y have a common refinement (Lemma 5.7). This part is purely syntactic. Next we show that (under the axiom **fin**)

$$X \equiv \bigvee \{S(A_0, \ldots, A_k) | S(A_0, \ldots, A_k) \leqslant X, k \geqslant 0\}$$

for any X, where the A_i are atoms of EFL(X) and $S(A_0, \ldots, A_k)$ is the formula defined in Definition 2.2, and $\bigvee$ denotes the infinite join or least upper bound with respect to the relation $\leqslant$. This is done by proving that

$$1\,\mathbf{suf}X \equiv \bigvee_k \mathbf{n}^k X$$

$$\langle \alpha \rangle (X \wedge 1\,\mathbf{suf}Y) \equiv \bigvee_k \langle \alpha \rangle (X \wedge \mathbf{n}^k Y).$$

Finally, the technique of [15] is applied. A path model is built with states $\{A_i | A_i \wedge \mathbf{L}_0$ is consistent$\}$, and it is shown that the path $(A_0, \ldots, A_k)$ satisfies X in this model iff $S(A_0, \ldots, A_k) \leqslant X$.

A *partition* is a finite set π of consistent but pairwise inconsistent formulas with $\bigvee \pi \equiv 1$. If π and ρ are partitions, then so is their *coarsest common refinement*

$$\pi \wedge \rho = \{X \wedge Y | X \in \pi, Y \in \rho, X \wedge Y \text{ consistent}\}.$$

For any subset σ of a partition, define

$$(\alpha)\,\sigma = (\textstyle\bigwedge_{A \in \sigma} \langle \alpha \rangle A) \wedge [\alpha](\textstyle\bigvee_{A \in \sigma} A).$$

The formula $(\alpha)\,\sigma$ says that program α enables every $A \in \sigma$, but no $A \notin \sigma$. Note that $(\alpha)\varnothing = [\alpha]\,0$ and that if π is a partition, then so is the set $\{(\alpha)\,\sigma | \sigma \subseteq \pi\}$.

Now we define by mutual recursion the four concepts:

(i) $(\mathbf{n}, \mathbf{L})$-*partition*,

(ii) $(\langle\,\rangle, P)$-*partition*,

(iii) $\mathbf{L}_0$-*special form*, and

(iv) *special form.*

Special forms are like normal forms in that they give a subset of formulas obeying certain syntactic restrictions, but nevertheless represent all formulas up to provable equivalence. We hesitate to use the term *normal form* because the representation is not unique. $\mathbf{L}_0$-special forms are meaningful because they represent the result of attempting to show how the satisfiability of a formula in a path depends on the satisfiability of a set of other formulas in states of that path. The *other formulas* in that set are $\mathbf{L}_0$-special forms.

(i) An $(\mathbf{n}, \mathbf{L})$-*partition* is a partition of the form $\{nX_1,..., nX_k, L_0\}$.

(ii) A partition π is a $(\langle\,\rangle, P)$-*partition* if there exist $(\mathbf{n}, \mathbf{L})$-partitions $\pi_1,..., \pi_k$, distinct primitive program letters $a_1,..., a_k$, and distinct primitive proposition letters $P_1,..., P_n$ such that π is the set of all consistent terms of the form

$$\langle a_1\rangle \sigma_1 \wedge \cdots \wedge \langle a_k\rangle \sigma_k \wedge Q_1 \wedge \cdots \wedge Q_n,$$

where each Q_j is either P_j or $\neg P_j$ and each σ_i is a subset of π_i.

(iii) An $\mathbf{L}_0$-*special form* is a term of the form $\mathbf{L}_0 \wedge \bigvee\sigma$, where σ is a subset of a $(\langle\,\rangle, P)$-partition.

(iv) A *special form* is a term of the form

$$\bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z),$$

where $\{\mathbf{n}X_1,..., \mathbf{n}X_k, \mathbf{L}_0\}$ is an $(\mathbf{n}, \mathbf{L})$-partition and all Y_i and Z are in $\mathbf{L}_0$-special form. For example, the primitive proposition P is equivalent to the special form $(\mathbf{n}1 \wedge \mathbf{f}P) \vee (\mathbf{L}_0 \wedge P)$, and 1 is equivalent to the special form $(\mathbf{n}1 \wedge \mathbf{f}1) \vee (\mathbf{L}_0 \wedge 1)$. (Strictly speaking, the $\mathbf{f}P$ and $\mathbf{f}1$ in the above examples should really be $\mathbf{f}(\mathbf{L}_0 \wedge P)$ and $\mathbf{f}(\mathbf{L}_0 \wedge 1)$, but we omit the $\mathbf{L}_0$ in light of Theorem 4.2(12). This abuse, and others like it that may appear in the sequel, are for notational convenience and should-cause no misunderstanding.)

Let σ be a subset of an $(\mathbf{n}, \mathbf{L})$-partition. By Theorem 4.2(18), if $\mathbf{L}_0 \in \sigma$, then $\mathbf{n}1 \wedge \langle\alpha\rangle \sigma$ is inconsistent; otherwise, by Theorem 4.2(20),

$$\mathbf{n}1 \wedge \langle\alpha\rangle \sigma \equiv \mathbf{n}\langle\alpha\rangle \sigma^1,$$

where

$$\sigma^1 = \{A \mid \mathbf{n}A \in \sigma\}.$$

The coarsest common refinement $\pi \wedge \rho$ of any two $(\mathbf{n}, \mathbf{L})$-partitions π and ρ can be made into an $(\mathbf{n}, \mathbf{L})$-partition using Theorem 4.2(7). This partition is denoted by $\pi \triangle \rho$. Similarly, any two $(\langle\,\rangle, P)$-partitions π and ρ have a common refinement $\pi \triangle \rho$ that is a $(\langle\,\rangle, P)$-partition and is coarsest among all such refinements, obtained by forming $\pi \wedge \rho$, taking the coarsest common refinements of the $(\mathbf{n}, \mathbf{L})$-partitions in the definitions of π and ρ, and using the PDL axioms. By a *refinement* of an $(\mathbf{n}, \mathbf{L})$- or $(\langle\,\rangle, P)$-partition π we shall mean any partition $\pi \triangle \rho$, where ρ is a partition of the same type.

If the $\mathbf{L}_0$-special form X is defined in terms of the $(\langle\,\rangle, P)$-partition π, and if ρ is a refinement of π, then there is an $\mathbf{L}_0$-special form Y equivalent to X and defined in terms of ρ, obtained by replacing $\bigvee\sigma$ with $\bigvee\tau$, where τ is the unique subset of ρ such that $\bigvee\sigma \equiv \bigvee\tau$. Such a Y is called a *refinement* of X. Similarly, a *refinement* of special form X is an equivalent special form Y such that the $\mathbf{L}_0$-special forms and $(\mathbf{n}, \mathbf{L})$-partition appearing in the definition of Y are refinements of those in the definition of X. Any $\mathbf{L}_0$-special form or special form is equivalent to all its refinements.

Now we associate with each X a special form X' equivalent to X. This is done by induction on term structure. A special form P' for primitive P has been given above, and below we give special forms for $X \wedge Y$, $\neg X$, $X \mathbf{suf} Y$, $\mathbf{f}X$, and $\langle a\rangle X$, provided X and Y are already in special form. Note that X' is uniquely defined relative to given special form representations of all the maximal proper subformulas of X, but in general there are infinitely many special form representations for the subformulas of X and hence infinitely many possible X'. In the process of defining the following special forms, we shall simultaneously be proving by induction on formula structure that $X \equiv X'$, and if $X' = \bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z)$, then the X_i, Y_i, and any W that occurs in Z in the form $\langle a\rangle\, \mathbf{n}W$ are equivalent to formulas in EFL(X).

First we define $(X \wedge Y)'$, where X and Y are the special forms

$$\bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee Z, \qquad \bigvee_j(\mathbf{n}U_j \wedge \mathbf{f}V_j) \vee W.$$

Now $X \wedge Y$ is equivalent to

$$\bigvee_{i,j}(\mathbf{n}(X_i \wedge U_j) \wedge \mathbf{f}(Y_i \wedge V_j)) \vee (Z \wedge W),$$

obtained by converting to disjunctive normal form, using Theorem 4.2(1), (7), and deleting inconsistent terms. Now $Z \wedge W$ can be put into $\mathbf{L}_0$-special form by taking the coarsest common refinements of the $(\langle\,\rangle, P)$-partitions defining Z and W and using the PDL axioms to combine terms of the form $(a)\,\sigma$ and $(a)\,\tau$. The same holds for the $Y_i \wedge V_j$. This process uses Theorem 4.2(7) and the PDL axioms. We take $(X \wedge Y)'$ to be the resulting formula.

If X is the special form $\bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z)$, where $Y_i = \mathbf{L}_0 \wedge \bigvee\sigma_i$ and $Z = \mathbf{L}_0 \wedge \bigvee\sigma_0$, each $\sigma_i \subseteq \pi_i$ where the π_i where the π_i are $(\langle\,\rangle, P)$-partitions, then take $\sim\sigma_i$ to be the complement of σ_i in π_i and take

$$(\neg X)' = \bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}(\mathbf{L}_0 \wedge \bigvee \sim \sigma_i) \vee (\mathbf{L}_0 \wedge \bigvee \sim \sigma_0).$$

To show $\neg X \equiv (\neg X)'$, note that since $\{\mathbf{n}X_1, \ldots, \mathbf{n}X_k, L_0\}$ is a partition,

$$\neg X \equiv \neg(\bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z)) \equiv \bigvee_i(\mathbf{n}X_i \wedge \neg\mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge \neg Z)$$

by purely propositional reasoning. Then

$$\neg\mathbf{f}Y_i \equiv \mathbf{f}(L_0 \wedge \neg\bigvee\sigma_i), \qquad \mathbf{L}_0 \wedge \neg Z \equiv \mathbf{L}_0 \wedge \neg\bigvee\sigma_0$$

by Axiom 4.1(3) and Theorem 4.2(12); finally, $\neg\bigvee\sigma_i \equiv \bigvee \sim \sigma_i$ by the PDL axioms. This gives $(\neg X)'$.

$$(X\mathbf{suf}Y)' \quad \text{is defined to be} \quad (\mathbf{n}X_1 \wedge \mathbf{f}1) \vee (\mathbf{n}X_2 \wedge \mathbf{f}1) \vee (\mathbf{n}X_3 \wedge \mathbf{f}0) \vee (\mathbf{L}_0 \wedge 0),$$

where $X_1 = Y$, $X_2 = X \wedge (X\mathbf{suf}Y)$, and $X_3 = \mathbf{n}(\neg X_1 \wedge \neg X_2)$. This uses Axiom 4.1(7).

We leave the definition of $(\mathbf{f}X)'$ as an exercise. This case uses Theorem 4.2(11), (12).

The hardest case is $(\langle\alpha\rangle X)'$. If $X = \bigvee_i(\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z)$, then

$$\begin{aligned}\langle\alpha\rangle X &\equiv \bigvee_i((\langle\alpha\rangle \mathbf{n}X_i) \wedge \mathbf{f}Y_i) \vee \langle\alpha\rangle(\mathbf{L}_0 \wedge Z)\\ &\equiv \bigvee_i((\langle\alpha\rangle \mathbf{n}X_i) \wedge \mathbf{n}1 \wedge \mathbf{f}Y_i)\\ &\quad \vee \bigvee_i((\langle\alpha\rangle \mathbf{n}X_i) \wedge \mathbf{L}_0 \wedge \mathbf{f}Y_i) \vee \langle\alpha\rangle(\mathbf{L}_0 \wedge \bigvee\sigma),\end{aligned} \tag{5.1}$$

where $Z = \mathbf{L}_0 \wedge \bigvee\sigma$. By Theorem 4.2(18), $\langle\alpha\rangle(\mathbf{L}_0 \wedge \bigvee\sigma) \equiv \mathbf{L}_0 \wedge \bigvee\sigma \wedge \langle\alpha\rangle \mathbf{L}_0$. But $\langle\alpha\rangle \mathbf{L}_0$ can be reduced to a positive Boolean combination of formulas of the form $\langle a\rangle \mathbf{L}_0$, where the a are primitive programs that occur in α, using Theorem 4.2(18) and the PDL axioms. Thus the third term in (5.1) is equivalent to a term of the form $\mathbf{L}_0 \wedge \bigvee(\bigwedge\langle a\rangle \mathbf{L}_0 \wedge \sigma)$. Now each $\langle a\rangle \mathbf{L}_0$ is incorporated into σ. If σ contains a term of the form $(a)\tau$ and τ does not contain $\mathbf{L}_0$, then the result is inconsistent. If σ contains $(a)\tau$ and $\mathbf{L}_0$ is in τ, then the result is $(a)\tau$. If σ does not contain any term of the form $(a)\tau$, then a refinement has to be taken.

The second term of (5.1) is equivalent to $\bigvee_i \mathbf{L}_0 \wedge Y_i \wedge \langle\alpha\rangle \mathbf{n}X_i$ by Theorem 4.2(11). The Y_i are already in $\mathbf{L}_0$-special form. The α in the term $\langle\alpha\rangle \mathbf{n}X_i$ is now broken up using the PDL axioms. $\langle\beta \cup \gamma\rangle \mathbf{n}X_i$ becomes $\langle\beta\rangle \mathbf{n}X_i \vee \langle\gamma\rangle \mathbf{n}X_i$. $\langle\beta\gamma\rangle \mathbf{n}X_i$ becomes

$$\langle\beta\rangle(\mathbf{L}_0 \wedge \langle\gamma\rangle \mathbf{n}X_i) \vee \langle\beta\rangle(\mathbf{n}1 \wedge \langle\gamma\rangle \mathbf{n}X_i). \tag{5.2}$$

The first term of (5.2) is equivalent to $\langle\beta\rangle \mathbf{L}_0 \wedge \langle\gamma\rangle \mathbf{n}X_i$ using Theorem 4.2(18). The $\langle\beta\rangle \mathbf{L}_0$ is decomposed as above and the process is applied recursively to decompose the γ. By Theorem 4.2(20), the second term of (5.2) is equivalent to $\langle\beta\rangle \mathbf{n}\langle\gamma\rangle X_i$, and the process is applied recursively to decompose β. Finally, if α is of the form β^*, by Theorem 4.2(21), the $\mathbf{L}_0 \wedge \langle\beta^*\rangle \mathbf{n}X_i$ appearing in the second term of (5.1) can be replaced by $\mathbf{L}_0 \wedge \langle\beta\rangle \mathbf{n}\langle\beta^*\rangle X_i$, and the procedure can be applied recursively. This process continues until all α appearing outside the scope of an $\mathbf{n}$ are primitive. Then the second and third term of (5.1) together form a Boolean combination of $\mathbf{L}_0$ and terms of the form $\langle a \rangle \mathbf{n}X$ and $\langle a \rangle \mathbf{L}_0$. Each $\langle a \rangle \mathbf{n}X$ is replaced by

$$\bigvee \{(a)\, \sigma \mid \sigma \subseteq \pi, nX \in \sigma\},$$

where $\pi = \{\mathbf{n}X, \mathbf{n}\neg X, \mathbf{L}_0\}$, and each $\langle a \rangle \mathbf{L}_0$ is replaced by

$$\bigvee \{(a)\, \sigma \mid \sigma \subseteq \{\mathbf{L}_0, \mathbf{n}1\}, \mathbf{L}_0 \in \sigma\}$$

and the Δ-meet of all these partitions is taken. This results in an equivalent $\mathbf{L}_0$-special form. Finally, by Theorem 4.2(20), the first term in the expression (5.1) is equivalent to

$$\bigvee_i (\mathbf{n}\langle \alpha \rangle X_i \wedge \mathbf{f}Y_i).$$

The set $\{\mathbf{n}\langle \alpha \rangle X_1, \ldots, \mathbf{n}\langle \alpha \rangle X_n, \mathbf{L}_0\}$ is not necessarily an $(\mathbf{n}, \mathbf{L})$-partition, but we can make it so by taking the Δ-meet of all the $(\mathbf{n}, \mathbf{L})$-partitions $\{\mathbf{n}\langle \alpha \rangle \mathbf{X}_i, \mathbf{n}\neg\langle \alpha \rangle X_i, \mathbf{L}_0\}$. The resulting formula is in special form and is taken to be $(\langle \alpha \rangle X)'$.

We shall write $X \rightarrow^r Y$ iff Y is a refinement of some X', where X' is defined according to the above construction. We have already proved in the above construction

LEMMA 5.3. *If $X \rightarrow^r Y$, then $X \equiv Y$.*

LEMMA 5.4. *If $X \equiv Y$ is an instance of an axiom, then there is a special form Z such that $X \rightarrow^r Z$ and $Y \rightarrow^r Z$.*

Proof. This is quite straightforward to check for almost all the axioms, but lack of a good notaton makes some cases tedious, especially the PDL axioms. We argue the case of the PDL axiom

$$\langle \alpha^* \rangle X \equiv X \vee \langle \alpha\alpha^* \rangle X.$$

If $X \equiv \bigvee_i (\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z)$, then in the process of deriving $(\langle \alpha^* \rangle X)'$, $\langle \alpha^* \rangle X$ reduces to

$$\bigvee_i (\mathbf{n}\langle \alpha^* \rangle X_i \wedge \mathbf{f}Y_i) \vee \bigvee_i (\mathbf{L}_0 \wedge \langle \alpha \rangle \mathbf{n}\langle \alpha^* \rangle X_i \wedge Y_i) \vee (\mathbf{L}_0 \wedge Z). \tag{5.5}$$

Similarly $X \vee \langle \alpha\alpha^* \rangle X$ reduces to

$$\begin{aligned} &\bigvee_i (\mathbf{n}X_i \wedge \mathbf{f}Y_i) \vee (\mathbf{L}_0 \wedge Z) \vee \bigvee_i (\mathbf{n}\langle \alpha\alpha^* \rangle X_i \wedge \mathbf{f}Y_i) \\ &\qquad \vee \bigvee_i (\mathbf{L}_0 \wedge \langle \alpha\alpha^* \rangle \mathbf{n}X_i \wedge Y_i) \vee (\mathbf{L}_0 \wedge Z \wedge \langle \alpha\alpha^* \rangle \mathbf{L}_0). \end{aligned} \tag{5.6}$$

The first term of (5.5) is equivalent to the join of the first and third terms of (5.6), by the PDL axiom above and Theorem 4.2(6). Thus the first term of (5.5) and the join of the first and third terms of (5.6) will produce the same result if we refine (5.5) and (5.6) by the $(\mathbf{n}, \mathbf{L})$-partitions $\{\mathbf{n}X_i, \mathbf{n}\neg X_i, \mathbf{L}_0\}$ and $\{\mathbf{n}\langle \alpha\alpha^* \rangle X_i, \mathbf{n}\neg\langle \alpha\alpha^* \rangle X_i, \mathbf{L}_0\}$.

We can discard the fifth term of (5.6) since it is covered by the second term. But the second term of (5.6) is identical to the third term of (5.5). Finally, the second term of (5.5) and the fourth term of (5.6) can be shown to have a common refinement using Theorem 4.2(18), (20), (21). ∎

LEMMA 5.7. *If $X \equiv Y$, then there exists a Z such that $X \to^r Z$ and $Y \to^r Z$.*

Proof. If $X \equiv Y$ is an instance of an axiom, then the result follows from Lemma 5.4. If $\oplus WZ \equiv \oplus YZ$ by virtue of the fact that $X \equiv Y$, then by induction on term structure there exists a W such that $X \to^r W$ and $Y \to^r W$. Since W is a special form representation of both X and Y, $\oplus XZ \to^r (\oplus WZ)'$ and $\oplus YZ \to^r (\oplus WZ)'$. Finally, if $X \equiv Z$ by virtue of the fact that there is a Y such that $X \equiv Y \equiv Z$, then by induction there are U, V such that $X \to^r U$, $Y \to^r U$, $Y \to^r V$, and $Z \to^r V$. Then $X \to^r (U \triangle V)$ and $Z \to^r (U \triangle V)$. ∎

As an immediate corollary of this result, we have

LEMMA 5.8. (i) *If $\mathbf{n}X$ and $\mathbf{f}Y$ are both consistent, then $\mathbf{n}X \wedge \mathbf{f}Y$ is;*

(ii) *if $\mathbf{n}X$ is consistent, then $\mathbf{L}_0 \wedge \langle a \rangle \mathbf{n}X$* is;

(iii) *if X is consistent, then $\mathbf{n}X$ is.*

Proof. (i) If $\mathbf{n}X \wedge \mathbf{f}Y \equiv 0$, then by the previous lemma, there exists a Z such that $\mathbf{n}X \wedge \mathbf{f}Y \to^r Z$ and $0 \to^r Z$. But by definition of $\to^r$, $0 \to^r Z$ iff $Z = 0$, and $\mathbf{n}X \wedge \mathbf{f}Y \to^r 0$ iff either $\mathbf{n}X \to^r 0$ or $\mathbf{f}Y \to^r 0$. Cases (ii) and (iii) are similar. ∎

In the following, W is a consistent formula of PL and $A, B, C, \ldots,$ denote atoms of EFL(W).

LEMMA 5.9. (i) *If $\mathbf{f}A \wedge \mathbf{n}B \wedge C$ is consistent, then $\mathbf{f}A \wedge \mathbf{n}B \leqslant C$;*

(ii) *if $S(A_0, \ldots, A_k) \wedge C$ is consistent, then $S(A_0, \ldots, A_k) \leqslant C$;*

(iii) *if $S(A_0, \ldots, A_k; B) \wedge C$ is consistent, then $S(A_0, \ldots, A_k; B) \leqslant C$.*

Proof. (i) Let $X \in$ EFL(W) such that $C \leqslant X$. Using the representation X', $\mathbf{f}A \wedge \mathbf{n}B$ is consistent with some subformula $\mathbf{f}Z \wedge \mathbf{n}W$ occurring in X', where Z and W are in EFL(X) $\subseteq$ EFL(W). Since A and B are atoms, $A \leqslant Z$ and $B \leqslant W$, thus $\mathbf{f}A \wedge \mathbf{n}B \leqslant \mathbf{f}Z \wedge \mathbf{n}W \leqslant X$. Since C is the meet of all such X, $\mathbf{f}A \wedge \mathbf{n}B \leqslant C$. Cases (ii) and (iii) follow from (i). ∎

Lemma 5.9 implies that $C \wedge \mathbf{L}_k \equiv \bigvee S(A_0, \ldots, A_k)$, where the join is over all sequences $(A_0, \ldots, A_k)$ such that $S(A_0, \ldots, A_k) \wedge C$ is consistent.

LEMMA 5.10. (i) *For any formula Y,*

$$\mathbf{some}\ Y \equiv \bigvee\nolimits_{k>0} \mathbf{n}^k Y$$

*in the sense that for any Z, $Z \wedge$ **some** Y is consistent iff $Z \wedge \mathbf{n}^k Y$ is consistent for some $k \geqslant 0$.*

(ii) *For any Y, $\langle\alpha\rangle(X \wedge \textbf{some } Y) \equiv \bigvee_{k>0}\langle\alpha\rangle(X \wedge \mathbf{n}^k Y)$.*

Proof. (i) The direction $\geqslant$ is immediate from the axioms. For the direction $\leqslant$, let Z be any formula of PL such that $W = Z \wedge \textbf{some } Y$ is consistent. Form EFL(W) with atoms A, B,.... Let

$$U = \bigvee \{C \in \mathrm{EFL}(W) \mid \exists k C \wedge \mathbf{n}^k Y \text{ consistent}\}.$$

Then $Y \leqslant U$, and we claim also that $\mathbf{n}U \leqslant U$, for if $\neg U \wedge \mathbf{n}U$ were consistent, then $A \wedge \mathbf{n}C$ would be consistent for some $A \leqslant \neg U$ and $C \leqslant U$, and then $\mathbf{f}B \wedge \mathbf{n}C$ would be consistent for some $\mathbf{f}B \wedge \mathbf{n}C \leqslant A$, by Lemma 5.9. But $C \wedge \mathbf{n}^k Y$ is consistent since $C \leqslant U$, so $\mathbf{f}B \wedge \mathbf{n}(C \wedge \mathbf{n}^k Y)$ is by Lemma 5.8, and $\mathbf{f}B \wedge \mathbf{n}(C \wedge \mathbf{n}^k Y) \leqslant A \wedge \mathbf{n}^{k+1} Y$, thus the latter is consistent, which contradicts the fact that $A \leqslant \neg U$. Then

$$\begin{aligned} Z \wedge \textbf{some } Y &\leqslant Y \vee (1\,\mathbf{suf} Y), \\ &\leqslant U \vee ((\mathbf{n}U \supset U)\,\mathbf{suf} U), \\ &\leqslant U \vee \mathbf{n}U \qquad \text{by the path induction axiom} \\ &\leqslant U. \end{aligned}$$

Then there is an atom $C \leqslant Z \wedge U$, therefore $Z \wedge \mathbf{n}^k Y$ is consistent for some k. The proof of (ii) is similar. ∎

Using Lemmas 5.9 and 5.10 we get

LEMMA 5.11. (i) $\mathbf{fin} \equiv \bigvee_{k>0} \mathbf{L}_k$ and $\langle\alpha\rangle\mathbf{fin} \equiv \bigvee_{k>0}\langle\alpha\rangle\, \mathbf{L}_k$;

(ii) $X \wedge \mathbf{fin} \equiv \bigvee S(A_0,..., A_k)$ *and* $\langle\alpha\rangle(X \wedge \mathbf{fin}) \equiv \bigvee \langle\alpha\rangle\, S(A_0,..., A_k)$, *where the join is over all $S(A_0,..., A_k)$ consistent with X.*

It is interesting to note that the proof of Lemma 5.10 gives a bound on the least k such that $Z \wedge \mathbf{n}^k Y$ is consistent: $k \leqslant |\mathrm{EFL}(Z \wedge \textbf{some } Y)|$.

The remainder of the proof mimics the completeness proof for PDL given in [15]. We first prove the result in the absence of infinte paths, and indicate later how to extend the result to the general case. Accordingly, we assume the axiom **fin** and restrict interpretations to path models with only finite paths.

Let W be a consistent formula of PL and let A, B, C, A_0,..., denote atoms of EFL(W). We shall construct a path model M and a finite path p in M with $p \models W$. The states of M will be the atoms A of EFL(W) such that $A \leqslant \mathbf{L}_0$. Paths in this model consist of sequences of states $(A_0,..., A_k)$. The reader should bear in mind that, as in the proof in [15], the atoms A play two roles: formulas in the language of PL and states in the model M. The particular role is to be determined from context. The interpretation of the primitive formulas is given by:

$$(A) \models P \qquad \text{iff} \quad A \leqslant P.$$

The interpretation of primitive programs is given by:

$$(A_0,\ldots,A_k)\in R_a \quad \text{iff} \quad \mathbf{L}_0 \wedge \langle a\rangle\, S(A_0,\ldots,A_k) \text{ is consistent.}$$

The following three lemmas are analogous to [15, Lemmas 1–3].

LEMMA 5.12. *Let α be any program. If $\mathbf{L}_0 \wedge \langle\alpha\rangle\, S(A_0,\ldots,A_k)$ is consistent, then $(A_0,\ldots,A_k)\in R_\alpha$.*

Proof. The proof is by structural induction on α. The basis is by definition and the case $\alpha=\beta\cup\gamma$ is trivial. For the case $\alpha=\beta\gamma$, suppose $\mathbf{L}_0\wedge\langle\beta\gamma\rangle\, S(A_0,\ldots,A_k)$ is consistent. Then $\mathbf{L}_0\wedge\langle\beta\rangle(\langle\gamma\rangle\, S(A_0,\ldots,A_k))$ is consistent. But

$$\langle\beta\rangle(\langle\gamma\rangle\, S(A_0,\ldots,A_k)) \equiv \textstyle\bigvee_{0\leqslant i\leqslant k}\langle\beta\rangle(S(A_0,\ldots,A_i)\wedge\langle\gamma\rangle\, S(A_0,\ldots,A_k)),$$

therefore, $\langle\beta\rangle(S(A_0,\ldots,A_i)\wedge\langle\gamma\rangle\, S(A_0,\ldots,A_k))$ is consistent for some $0\leqslant i\leqslant k$. Thus $\mathbf{L}_0\wedge\langle\beta\rangle\, S(A_0,\ldots,A_i)$ and $S(A_0,\ldots,A_i)\wedge\langle\gamma\rangle\, S(A_0,\ldots,A_k)$, and therefore, $\mathbf{L}_0\wedge\langle\gamma\rangle\, S(A_i,\ldots,A_k)$, are consistent. The result for this case then follows from two applications of the induction hypothesis.

For the case $\alpha=\beta^*$, suppose $\mathbf{L}_0\wedge\langle\beta^*\rangle\, S(A_0,\ldots,A_k)$ is consistent. The induction axiom of PDL says

$$\begin{aligned}\langle\beta^*\rangle\, S(A_0,\ldots,A_k)\leqslant S(A_0,\ldots,A_k)\vee\langle\beta^*\rangle(&\neg S(A_0,\ldots,A_k)\\ &\wedge\langle\beta\rangle\, S(A_0,\ldots,A_k))\end{aligned}$$

and Theorem 4.2(20) and elementary manipulations yield

$$\leqslant S(A_0,\ldots,A_k)\vee\langle\beta^*\rangle(\textstyle\bigvee_{0\leqslant i<k} S(A_0,\ldots,A_i)\wedge\langle\beta\rangle\, S(A_0,\ldots,A_k)).$$

If $k=0$, then $(A_0)\in R_{\beta^*}$ and we are done. Otherwise,

$$\mathbf{L}_0\wedge\langle\beta^*\rangle(\textstyle\bigvee_{0\leqslant i<k} S(A_0,\ldots,A_i)\wedge\langle\beta\rangle\, S(A_0,\ldots,A_k))$$

is consistent, so there must be an $i<k$ such that $\mathbf{L}_0\wedge\langle\beta^*\rangle\, S(A_0,\ldots,A_i)$ and $S(A_0,\ldots,A_i)\wedge\langle\beta\rangle\, S(A_0,\ldots,A_k)$, and hence $\mathbf{L}_0\wedge\langle\beta\rangle\, S(A_i,\ldots,A_k)$, are consistent. By the induction hypothesis, $(A_i,\ldots,A_k)\in R_\beta$, and then we repeat the process in order to break up $\mathbf{L}_0\wedge\langle\beta^*\rangle\, S(A_0,\ldots,A_i)$. In this fashion we obtain a finite list $p_0,\ldots,p_m$ of elements of R_β such that $p_0\,p_1\cdots p_m=(A_0,\ldots,A_k)$. ∎

LEMMA 5.13. *Let $\langle\alpha\rangle X\in \mathrm{EFL}(W)$. Then $S(A_0,\ldots,A_k)\leqslant\langle\alpha\rangle X$ iff there exist $A_{k+1},\ldots,A_m$ such that $(A_k,\ldots,A_m)\in R_\alpha$ and $S(A_0,\ldots,A_k,\ldots,A_m)\leqslant X$.*

Proof. $(\rightarrow)$.

$$\begin{aligned}&S(A_0,\ldots,A_k)\leqslant\langle\alpha\rangle X\\ &\qquad\rightarrow S(A_0,\ldots,A_k)\wedge\langle\alpha\rangle B \quad \text{is consistent for some} \quad B\leqslant X\\ &\qquad\rightarrow S(A_0,\ldots,A_k)\wedge\langle\alpha\rangle\, S(A_0,\ldots,A_{k-1};C) \quad \text{consistent}\end{aligned}$$

and $S(A_0,\ldots,A_{k-1};C)\leqslant B$, by Lemma 5.9

$$\rightarrow S(A_k)\wedge\langle\alpha\rangle C \quad \text{consistent,}$$
$$\rightarrow \exists A_{k+1},\ldots,A_m(A_k,\ldots,A_m)\in R_\alpha \quad \text{and} \quad S(A_k,\ldots,A_m)\leqslant C$$

by Lemmas 5.9, 5.11, and 5.12,

$$\begin{aligned}&\rightarrow (A_k,\ldots,A_m)\in R_\alpha \quad \text{and} \quad S(A_0,\ldots,A_k,\ldots,A_m)\\ &= S(A_0,\ldots,A_{k-1};\,S(A_k,\ldots,A_m))\\ &\leqslant S(A_0,\ldots,A_{k-1};C)\leqslant X.\end{aligned}$$

($\leftarrow$) This is by induction on the structure of α. We argue the case $\alpha=\beta^*$. If $S(A_0,\ldots,A_k,\ldots,A_m)\leqslant X$ and $(A_k,\ldots,A_m)\in R_{\beta^*}$, then $(A_k,\ldots,A_m)\in R_{\beta^n}$ for some n. Then there exist $k=k_0\leqslant\cdots\leqslant k_p=m$ with $(A_{k_i},\ldots,A_{k_{i+1}})\in R_\beta$. Since

$$S(A_0,\ldots,A_{k_{p-1}},\ldots,A_{k_p})\leqslant X\leqslant\langle\beta^*\rangle X \quad \text{and} \quad (A_{k_{p-1}},\ldots,A_{k_p})\in R_\beta,$$

by the induction hypothesis

$$S(A_0,\ldots,A_{k_{p-1}})\leqslant\langle\beta\rangle\langle\beta^*\rangle X\leqslant\langle\beta^*\rangle X.$$

Proceeding backward through all k_i in this fashion, we get $S(A_0,\ldots,A_k)\in\langle\beta^*\rangle X$. ∎

LEMMA 5.14. *Let* $X\in \mathrm{EFL}(W)$. *Then*

$$S(A_0,\ldots,A_k)\leqslant X \qquad \text{iff} \quad (A_0,\ldots,A_k)\vDash X.$$

Proof. The proof is by induction on the structure of X, using Lemma 5.13 for the case $X=\langle\alpha\rangle Y$. We argue the case $X=Y\,\mathbf{suf}Z$.

$S(A_0,\ldots,A_k)\leqslant Y\mathbf{suf}Z$ iff $S(A_0,\ldots,A_k)\wedge Y\mathbf{suf}Z$ is consistent, by Lemma 5.9,

iff $S(A_0,\ldots,A_k)\wedge\mathbf{n}^iZ$ is consistent for some $1\leqslant i\leqslant k$
and $S(A_0,\ldots,A_k)\wedge\mathbf{n}^jY$ is consistent for all $1\leqslant j<i$,

iff $S(A_i,\ldots,A_k)\leqslant Z$ and $S(A_j,\ldots,A_k)\leqslant Y$ for all $1\leqslant j<i$,

iff $(A_i,\ldots,A_k)\vDash Z$ and $(A_j,\ldots,A_k)\vDash Y$ for all $1\leqslant j<i$,

iff $(A_0,\ldots,A_k)\vDash Y\mathbf{suf}Z$. ∎

THEOREM 5.15. *The axiom system* PL + **fin** is complete.

Proof. Since $W\wedge\mathbf{fin}$ is consistent, by Lemma 5.11 there is a consistent $S(A_0,\ldots,A_k)\leqslant W$. By Lemma 5.14, $(A_0,\ldots,A_k)\vDash W$ in the model M. ∎

Now we indicate how to extend the proof of completeness to encompass infinite paths. Discard the axiom **fin** and suppose $W\wedge\mathbf{inf}$ is consistent. If X is any formula such that $X\wedge\mathbf{inf}$ is consistent, then there is an atom B_0 of EFL(W) such that $X\wedge\mathbf{inf}\wedge B_0$ is consistent. By Lemma 5.9, there are atoms A_0 and B_1 such that

$\mathbf{f}A_0 \wedge \mathbf{n}B_1 \leqslant B_0$ and $X \wedge \mathbf{inf} \wedge \mathbf{f}A_0 \wedge \mathbf{n}B_1$ is consistent, i.e., $X \wedge \mathbf{inf} \wedge S(A_0, B_1)$ is consistent. Continuining in this fashion we can construct a countably infinite sequence

$$B_0 \geqslant S(A_0; B_1) \geqslant S(A_0, A_1; B_2) \geqslant \cdots$$

of formulas such that each $X \wedge \mathbf{inf} \wedge S(A_0,..., A_k; B_{k+1})$ is consistent. In order to satisfy $W \wedge \mathbf{inf}$, our first instinct is to construct this sequence for $X = W$ and use the infinite path $(A_0, A_1,...)$. This path can fail to satisfy W, however. For example, if $W = P\mathbf{suf} \neg P$, then this construction can yield the infinite path $(P, P, P,...)$.

Let us call an infinite sequence

$$B_0 \geqslant S(A_0; B_1) \geqslant S(A_0, A_1; B_2) \geqslant \cdots$$

of consistent formulas *standard* if whenever $X\mathbf{suf}Y \in \mathrm{EFL}(W)$ and $B_0 \leqslant X\mathbf{suf}Y$, there exists a $k \geqslant 1$ such that $B_k \leqslant Y$. This definition excludes the counterexample above.

LEMMA 5.16. *If $X \wedge \mathbf{inf}$ is consistent, then there exists a standard sequence σ such that $X \wedge S$ is consistent for all $S \in \sigma$.*

Proof. Assume without loss of generality that $X \leqslant \mathbf{inf}$. Let $B_0 \wedge X$ be consistent and let $X_i\mathbf{suf}Y_i$, $1 \leqslant i \leqslant m$, be the set of all $X_i\mathbf{suf}Y_i \in \mathrm{EFL}(W)$ such that $B_0 \leqslant X_i\mathbf{suf}Y_i$. Since $B_0 \wedge X \wedge \bigwedge_{1 \leqslant i \leqslant m} X_i\mathbf{suf}Y_i$ is consistent, m applications of Lemma 5.10 yield $k_1,..., k_m$ such that $B_0 \wedge X \wedge \bigwedge_{1 \leqslant i \leqslant m} n^{k_i}Y_i$ is consistent. Now do the construction of

$$B_0 \geqslant S(A_0; B_1) \geqslant S(A_0, A_1; B_2) \geqslant \cdots$$

as above, with $B_0 \wedge X \wedge \bigwedge_{1 \leqslant i \leqslant m} n^{k_i}Y_i$ in place of X. The resulting sequence σ is standard, since $B_{k_i} \leqslant Y_i$. ∎

Now let the definition of M be modified to allow α to contain infinite paths. If σ is the sequence

$$B_0 \geqslant S(A_0; B_1) \geqslant S(A_0, A_1; B_2) \geqslant \cdots,$$

let p_σ denote the path $(A_0, A_1,...,)$ in M. Lemmas 5.13 and 5.14 are augmented with the following extra cases to handle the infinite paths:

LEMMA 5.17. *Let $\langle \alpha \rangle X \in \mathrm{EFL}$, $X \leqslant \mathbf{inf}$. Then $S(A_0,..., A_k) \leqslant \langle \alpha \rangle X$ iff there exists a standard σ such that $p_\sigma = (A_k,...,) \in R_\alpha$ and $S(A_0,..., A_{k-1}; S) \leqslant X$ for all $S \in \sigma$.*

LEMMA 5.18. *Let $X \in \mathrm{EFL}(W)$ and let σ be a standard sequence. Then $S \leqslant X$ for all $S \in \sigma$ iff $p_\sigma \vDash X$.*

The proofs of these lemmas are straightforward modifications of the proofs of Lemmas 5.13 and 5.14. Thus we have

THEOREM 5.19. *The axiom system* PL *is complete.*

6. Directions for Further Work

The Axiom $P = \mathbf{f}P$

The axiom $P = \mathbf{f}P$ makes the completeness and decidability proofs go through, since it allows the special form representation X'. This restriction is undesirable, however, since we would like to be able to substitute any path formula for the primitive P, not just those satisfying $P = \mathbf{f}P$. We would like to see a construction leading to Lemma 5.10 which bypasses this restriction.

The Test and Reverse Operators

We have not accounted for tests (?) or the reverse operator ($^-$) of DL. These operators in some cases make arguments simpler. Contrary to first thoughts, the reverse operator *does* make sense in the presence of infinite paths, if we define

$$p \models \langle \alpha^- \rangle X \qquad \text{iff} \quad \exists q \in R_\alpha(\exists r(p = rq \text{ and } r \models X)).$$

This semantics is quite different from the semantics of $^-$ in PDL, where $^-$ is a unary operator on programs. Here it is not an operator on programs, and α^- only makes sense in the context of a PL formula $\langle \alpha^- \rangle X$ Thus a better notation than $\langle \alpha^- \rangle X$ is needed. Nevertheless, under this semantics, the two PDL axioms for $^-$ are satisfied.

Expressive Completeness

In the presence of the axiom $P = \mathbf{f}P$, every path formula ultimately expresses poperties of *states* and how they interact, as with TL or NL. In [7, 10] it was proved that TL, and hence PL, is expressively complete for all such formulas, in the sense that any reasonable formula of states (meaning anything in the first order theory of linear order) can be expressed. In the absence of the axiom $P \equiv \mathbf{f}P$, however, PL is unable to express all reasonable *path* formulas. For example, without $P = \mathbf{f}P$, the operators **f** and **suf** are not sufficient to express the property **chop** defined by: $p \models \mathbf{chop}(X, Y)$ if there exist q, r with $p = qr$ and $q \models X$, $r \models Y$. (This is because program-free PL can be encoded in deterministic PDL, which is elementary [3], while program-free PL with **chop** is nonelementary.) Is there a good definition of *reasonable* path formula, and if so, what primitives in addition to **f** and **suf** are needed to make the system expressively complete?

Complexity of PL

As shown above, PL is no harder to decide than SnS, but it is not known whether PL is elementary. It is known that PL with **chop** is nonelementary, and PL with **chop** and without $P = \mathbf{f}P$ is undecidable [4]. This question could be answered in the negative if an efficient encoding of weak $S1S$ or the first order theory of linear order with first element into PL could be found. PL does encode the first order theory of linear order with first element [7, 10] but the only known encoding is nonelementary [1]. We thank Karl Abrahamson for pointing this out to us.

Algebraic and Topological Interpretation of PL

Reasoning in an algebraic context is often cleaner than in the framework of pure logic, since irrelevant syntactic details are suppressed. For example, Boolean algebra captures the essence of propositional logic at a better level of abstraction. The algebraic structure of PDL has been studied in the form of dynamic algebra [11–14, 29, 30] and has been found to aid insight and in some cases simplify proofs. Many of the results of this paper have natural algebraic and topological interpretations:

Let L be the Boolean algebra of formulas of PL modulo the PL axioms of Section 4, and let

$$\mathbf{n}L = \{\mathbf{n}X \mid X \in L\}, \qquad \mathbf{f}L = \{\mathbf{f}X \mid X \in L\}.$$

In Theorem 4.2(6)–(8) it is stated that $\mathbf{n}L$ is a Boolean subring of L with top element $\mathbf{n}1$ and that the map $\mathbf{n}: L \to \mathbf{n}L$ is a homomorphism, and in Lemma 5.8(iii), that it is an isomorphism; in Definition 4.1(2), (3) that $\mathbf{f}L$ is a Boolean subalgebra of L and that $\mathbf{f}$ is a homomorphism; in Theorem 4.2(14), that $\mathbf{f}$ is a projection $L \to \mathbf{f}L$ and in Definition 4.1(10) and Theorem 4.2(11) that $\mathbf{f}$ is the Boolean ideal generated by Im $\mathbf{n}$, and in Theorem 4.2(5) that $L \cong \mathbf{n}L$. By the construction of X' and Lemma 5.8(i), L is isomorphic to the direct sum of $\mathbf{n}L$ and $\mathbf{f}L$. Results involving joins and meets, such as Lemmas 5.10 and 5.17, have a natural topological interpretation involving density.

Acknowledgments

We are grateful to J. Stavi and D. Peleg for correcting errors in an early version of Section 3.

References

1. K. Abrahamson, correspondence.
2. L. Banachowski, A. Kreczmar, G. Mirkowska, H. Rasiowa and A. Salwicki, "An introduction to Algorithmic Logic," *in* "Mathematical Foundation of Computer Science," (Mazurkiewicz and Pawlak, Eds.), Banach Center Publications, Warsaw, 1977.
3. M. Ben-Ari, J. Halpern, and A. Pnueli, Finite models of deterministic propositional dynamic logic, *in* "Proc. 8th Colloq. on Automata, Languages, and Programming," Springer Lecture Notes in Comp. Sci., 1981.
4. A. K. Chandra, J. Halpern, A. Meyer and R. Parikh, Equations between regular terms and an application to Process Logic, *in* "Proc. 13th ACM Symp. on Theory of Computing," May 1981.
5. M. J. Fischer and R. E. Ladner, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (2) (1979), 194–211.
6. D. Gabbay, Axiomatizations of Logics of Programs, manuscript, November 1977.
7. D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, On the temporal analysis of fairness, *in* "Proc. 7th ACM Symp. on Principles of Programming Languages," January 1980, pp. 163–173.
8. D. Harel, First-Order Dynamic logic, "Lecture Notes in Computer Science 68" (Goos and Hartlmanis. Ed.), Springer–Verlag, Berlin, 1979.

9. D. HAREL, Two results on process logic, *Inform. Process. Lett.* **8** (4) (1979), 195–198.
10. H. W. KAMP, Tense logics and the theory of linear order, PhD. Thesis, UCLA, California, 1968.
11. D. KOZEN, A representation theorem for models of *-free PDL, *in* "Proc. 7th Int. Colloq. on Automata, Languages, and Programming, Lecture Notes in Computer Science Vol. 85, (1980), pp. 351–362, Springer–Verlag, New York/Berlin.
12. D. KOZEN, "On the Duality of Dynamic Algebras and Kripke Models, " Report RC7893, IBM Research, Yorktown Heights, New York, October 1979.
13. D. KOZEN, "On the Representation of Dynamic Algebras," Report RC7898. IBM Research, Yorktown Heights, New York,October 1979.
14. D. KOZEN, "On the Representation of Dynamic Algebras II," Report RC8290, IBM Research, Yorktown Heights, New York, May 1980.
15. D. KOZEN AND R. PARIKH, An elementary proof of the completeness of PDL, *Theoret. Comput. Sci.* **14** (1) (1981), 113–118.
16. Z. MANNA AND A. PNUELI, The modal logic of programs, *in* "Proc. 6th ICALP," Lecture Notes in Computer Sci., Vol. 74, 1979, pp. 385–409, Springer-Verlag, New York/Berlin.
17. A. R. MEYER, Weak monadic second order theory of successor is not elementary recursive, *in* "Proc. Logic Colloquium," Lecture Notes in Mathematics, Vol. 453, pp. 132–151, Springer–Verlag, New York/Berlin, (1975)
18. A. R. MEYER, Ten thousand and one logics of programming, *Bull. European Assoc. Theoret. Comput. Sci.* **10** (1980), 11–29.
19. A. R. MEYER AND R. PARIKH, Definability in dynamic logic, *in* "Proc. 12th ACM Symp. on Theory of Computing," April 1980, pp. 1–7.
20. H. NISHIMURA, Descriptively complete process logic, *Acta Inform.* **14** (4) (1980), 359–369.
21. R. PARIKH, A completeness result for PDL, *in* "Symp. on Math. Found. of Comp. Sci.," Zakopane, Warsaw, May 1978, Springer–Verlag, New York/Berlin.
22. R. PARIKH, A decidability result for second order process logic," *in* Proc. 19th FOCS, October 1978, pp. 177–183.
23. A. PNUELI, The temporal logic of programs, *in* "Proc. 18th FOCS, October 1977, pp. 46–57.
24. A. PNUELI, The temporal semantics of concurrent programs, *in* "Proc. Int. Symp. on Semantics of Concurrrent Programs," Evian, France, July 1979.
25. V. R. PRATT, Semantical considerations on Floyd–Hoare logic, *in* "Proc. 17th IEEE Symp. on Foundations of Comp. Sci.," Oct. 1976, pp. 109–121.
26. V. R. PRATT, A practical decision method for Propositional Dynamic Logic, *in* "Proc. 10th ACM Symp. on Theory of Computing," May 1978, pp. 326–337.
27. V. R. PRATT, Process logic, *in* "Proc. 6th ACM Symp. on Principles of Programming Languages," January 1979, pp. 93–100.
28. V. R. PRATT, A near optimal method for reasoning about action, *J. Comput. System Sci.* **20** (2) (1980), 231–254.
29. V. R. PRATT, Models of program logics, *in* "Proc. 20th IEEE Symp. on Foundations of Comp. Sci.," October 1979, pp. 115–122.
30. V. R. PRATT, Dynamic algebras and the nature of induction, *in* "Proc. 12th ACM Symp. on Theory of Computing," May 1980, pp. 22–28.
31. M. O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141** (1969), 1–35.
32. A. SALWICKI, Formalized algorithmic languages, *Bull. Acad. Polon. Sci. Ser. Math. Astronom. Phys.* **18** (5) (1970).
33. K. SEGERBERG, A completeness theorem in the modal logic of programs, *Notices Amer. Math. Soc.* **24** (6) (1977), A–552.
34. J. TIURYN, "Logic of Effective Definitions," manuscript, Inst. of Math., Warsaw Univ., July 1979.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium