On the Completeness of Dynamic Logic

Daniel Leivant

Computer Science Department,
Indiana University,
Bloomington, IN 47405, USA
leivant@cs.indiana.edu

Abstract. The impossibility of semantically complete deductive calculi for logics for imperative programs has led to the study of two alternative approaches to completeness: "local" semantic completeness on the one hand (Cook's relative completeness, Harel's Arithmetical completeness), and completeness with respect to other forms of reasoning about programs, on the other. However, local semantic completeness is problematic on several counts, whereas proof theoretic completeness results often involve *ad hoc* ingredients, such as formal theories for the natural numbers.

The notion of inductive completeness, introduced in [18], provides a generic proof theoretic framework which dispenses with extraneous ingredients, and yields local semantic completeness as a corollary.

Here we prove that (first-order) Dynamic Logic for regular programs (DL) is inductively complete: a DL-formula φ is provable in (the first-order variant of) Pratt-Segerberg deductive calculus **DL** iff φ, is provable in first-order logic from the inductive theory for program semantics. The method can be adapted to yield the *schematic* relative completeness of **DL**: if $\mathcal{S}$ is an expressive structure, then every formula true in $\mathcal{S}$ is provable from the axiom-*schemas* that are valid in $\mathcal{S}$. Harel's Completeness Theorem falls out then as a special case.

Keywords: Dynamic logic, inductive completeness, relative completeness, arithmetical completeness.

1 Introduction

1.1 Background

Logics of programs inherently exceed first-order logic, because program semantics is defined in terms of iterative processes, which can be formalized using second-order formulas [15], existential fixpoints [3], or explicit reference to the natural numbers (e.g. [6]), but not by a first-order theory. Consequently, the delineation of logics of programs cannot parallel the characterization of first-order logic by soundness and completeness for general ("uninterpreted") validity.

The early attempts to refer instead to *local completeness,* i.e. completeness with respect to one structure at a time, led to Cook's Relative Completeness Theorem. However, notwithstanding the persistent centrality of Cook's relative completeness [5] in research on logics of programs, that notion has foundational and practical drawbacks

L. de Alfaro (Ed.): FOSSACS 2009, LNCS 5504, pp. 78–91, 2009.

[17]. For one, relative completeness fails to demarcate the boundaries of logics of programs: there are, for example, proper extensions of Hoare's Logic that are sound and relatively complete!

An alternative approach is to match logics of programs with formalisms for *explicit* reasoning about programs, such as second-order logic. A common objection to second-order logic, that it is non-axiomatizable for its intended semantics, is off the mark here: On the one hand there are natural proof calculi for second-order logic that are sound and complete for a natural non-standard semantics (Henkin's structures); on the other hand, the same objection applies to arithmetic! Indeed, it turns out that Hoare's Logic matches second-order logic with first-order set-existence, and Dynamic Logic matches second-order logic with "computational" (i.e. strict-Π^1_1) set-existence [15,17].

Here we consider an approach dual to explicit second-order rendition of program semantics, namely an *implicit* but *first-order* rendition. We consider the inductive definition of program semantics, and invoke the well-known framework of first-order theories for such definitions [13,14,7,19,8]. We show that Dynamic Logic matches reasoning about programs in the inductive theory for regular programs.

Since this approach is strictly first-order, it is particularly accessible conceptually and expositorily, and directly amenable to automated theorem proving tools. It also meshes generically with the long-standing tradition of defining program semantics inductively. Whenever an inductive definition is given for a programming language, one obtains the corresponding first-order inductive-definition theory, and can tackle the question of completeness of a proposed logic of programs for that theory.

A match between Dynamic Logic and inductive theories was observed already by the Hungarian school of "nonstandard dynamic logic" (see e.g. [6,23]). However, the first-order theories they considered invoke the natural numbers as an auxiliary data-type. Our present approach calls for no such extraneous data-types, and is more generic, in that it applies directly to all programming constructs with inductively defined semantics, even when they lack a simple iterative definition in terms of the natural numbers.

1.2 Regular Programs

Regular programs distill and separate the essential components of imperative programs, and are therefore particularly amenable to logical analysis. Given a vocabulary V (i.e. "similarity type", "signature"), the *atomic V-programs* are *assignments* $x := \mathbf{t}$ of V-terms to variables, and *tests* $?\varphi$, where φ is a quantifier-free V-formula.[1] Compound *V-programs* are generated from atomic V-programs by composition, union, and Kleene's $*$ (nondeterministic iteration). Deterministic **while** programs are definable in terms of regular programs: $\texttt{(if } \varphi \texttt{ then } \alpha \texttt{ else } \beta\texttt{)}$ is an abbreviation for $(?\varphi;\ \alpha \cup (?\neg\varphi);\ \beta$, and $\texttt{(while } \varphi \texttt{ do } \alpha\texttt{)}$ for $(?\varphi;\ \alpha)^*;\ ?(\neg\varphi)$. We refer to [11] for background and detail.

Given a V-structure $\mathcal{S}$, the *states* are the $\mathcal{S}$-environments, i.e. partial functions from the set of variables to elements of $\mathcal{S}$. Each V-program α is interpreted semantically as a binary relation over states, denoted $\xrightarrow{\alpha}$, or $\xrightarrow[\mathcal{S}]{\alpha}$ when $\mathcal{S}$ is not obvious from the context.

[1] Our discussion below remains unchanged if tests are generalized to arbitrary first-order formulas. Tests for arbitrary DL formulas, known as "rich tests", can also be accommodated with minor changes.

These relations are defined by recurrence on α: $\eta \xrightarrow{x:=\mathbf{t}} \eta'$ iff $\eta' = \eta[x \leftarrow [\![\mathbf{t}]\!]_\eta]$; $\eta \xrightarrow{?\varphi} \eta'$ iff $\eta' = \eta$ and $\mathcal{S}, \eta \models \varphi$; $\xrightarrow{\alpha;\beta}$ is the composition of $\xrightarrow{\alpha}$ and $\xrightarrow{\beta}$; $\xrightarrow{\alpha\cup\beta}$ is the union of $\xrightarrow{\alpha}$ and $\xrightarrow{\beta}$; and $\xrightarrow{\alpha^*}$ is the $(\xrightarrow{\alpha})^*$, i.e. the reflexive-transitive closure of $\xrightarrow{\alpha}$.

1.3 Dynamic Logic Formulas

Given a vocabulary V, the DL V-formulas are generated inductively, just like first-order V-formulas, but with the added clause: If φ is a formula, and α a program, then $[\alpha]\varphi$ is a formula. The operator $\langle\alpha\rangle$, dual to $[\alpha]$, can be defined by $\langle\alpha\rangle\,\varphi \;\equiv_{\rm df}\; \neg[\alpha]\neg\varphi$.

The common convention, of not distinguishing between (assignable) program variables on the one hand and logical variables on the other, lightly complicates the interaction of assignments and quantifiers: We must posit that a quantifier cannot bind a variable that is assigned-to in its scope. For example, $\forall x\ [x := 1](x = 1)$ is not a legal DL formula, whereas $\forall x\ [y := x](y = x)$ is. Also, the definition of "free occurrence of variable x in formula φ" is amended to exclude x occurring in the scope of the modal operators $[\alpha]$ and $\langle\alpha\rangle$, when x is assigned-to in α.[2]

The definition of the satisfaction relation $\mathcal{S}, \eta \models \varphi$ is as usual by recurrence on φ. In particular, $\mathcal{S}, \eta \models [\alpha]\varphi$ iff $\mathcal{S}, \eta' \models \varphi$ whenever $\eta \xrightarrow{\alpha} \eta'$. Consequently, $\mathcal{S}, \eta \models \langle\alpha\rangle\,\varphi$ iff $\mathcal{S}, \eta' \models \varphi$ for some environment η' where $\eta \xrightarrow{\alpha} \eta'$.

1.4 Complexity

The set of valid DL formulas is highly complex. In fact, even DL formulas of a seemingly modest appearance have a decision problem more complex than that of first-order arithmetic:

Proposition 1
1. The validity problem for DL formulas is Π^1_1.
2. Even the validity problem for formulas of the form $\exists \boldsymbol{x}.[\alpha]\,\varphi$, where α is a deterministic program and φ is quantifier-free, is Π^1_1-hard.

Proof. (1) is easy, and proved in [11, Theorem 13.1], where (2) is also stated and proved, but for φ first-order and α non-deterministic. On closer inspection the proof there, which uses a Π^1_1-hard tiling problem, yields the result with φ quantifier-free.

However, already basic properties of Π^1_1 easily imply the refinement stated here (with α deterministic). Recall that every Π^1_1 formula φ is equivalent over $\mathbb{N}$ to a formula of the form

$$\forall f\ \exists x.\ g(x) = 0 \tag{1}$$

where f ranges over unary functions, and g is defined uniformly from f by primitive-recursion. More precisely, if D_g is the conjunction of the recurrence equations defining g from f, and $\boldsymbol{u}$ the variables free in D_g, then (1) is expressed by

$$\forall f, g\,(\,\bar{D}_g \;\rightarrow\; \exists x.\ g(x) = 0\,) \tag{2}$$

where $\bar{D}_g$ is the universal closure of D_g.

[2] This issue is central to [12], but is inconsequential here.

The truth of (2) in $\mathbb{N}$ is equivalent to the validity in all structures (over the vocabulary in hand) of the informal statement

$$\bar{D}_g \ \wedge \ (\forall v.\text{“}f(v) \text{ is the denotation of a numeral”}) \quad \rightarrow \quad \exists x.\ g(x) = 0 \quad (3)$$

where the *numerals* are the terms 0, $s(0)$, $s(s(0))$ (with s intended to denote the successor function).[3]

Now, with p intended to denote the predecessor function, let ψ be the conjunction of the three formulas

$$p(0) = 0$$
$$\forall y\ p(s(y)) = y$$
$$\text{and} \qquad \forall y, w\ (\ p(y) = p(w)\ \wedge\ y \neq 0\ \wedge w \neq 0\ \rightarrow\ y = w\ )$$

Thus, if ψ holds in a structure, and N is the program

$$z := f(v);\ \texttt{while}\ \ z \neq 0\ \ \texttt{do}\ \ z := p(z)\ \ \texttt{end}$$

then N terminates in k steps iff $f(v)$ is the denotation of the numeral $s^k(0)$. Therefore, (3) can be expressed in DL as

$$\bar{D}_g \ \wedge \ \psi \ \wedge \ (\forall v.\, \langle N \rangle \texttt{true}) \ \rightarrow \ \exists x\, g(x) = 0 \qquad (4)$$

Since v and z are the only variables in N, quantifiers over other variables commute with $[N]$, and so (4) can be converted into an equivalent formula of the form stated in the Proposition. ■

1.5 Axiomatics

One way of addressing the challenge presented by of Proposition 1 is to consider an infinitary deductive system. So-called omega-rules go back to the 1930's and their adaptation to logics of programs is due to Mirkowska [20]. In [11, Theorem 14.7] that proof is adapted to Dynamic Logic.

Since no effective axiomatization of DL can be complete for validity, the dual task is all the more interesting: articulate a *natural* axiomatization of DL, and delineate it in terms of a familiar deductive system. This is analogous to the situation with Hoare's Logic: the validity problem for Partial-correctness assertions is Π^0_2-complete [11, Theorem 13.5], implying that Hoare's Logic is not complete for validity, and that alternative completeness properties are needed to delineate the logic and explicate its significance.

A natural deductive calculus **DL** for Dynamic Logic is obtained by augmenting first-order logic with the rules of Table 1. This formalization is due primarily to Pratt [22,9]. The assignment rule is Hoare's, and the others are related to Segerberg's Propositional Dynamic Logic for regular programs [24]. This is closely related to the formalism 14.12 of [11], with the Convergence Rule omitted. Note that the quantifier rules of first-order logic can never be used to instantiate a variable x in an assignment $x := \mathbf{t}$, due to our syntactic restriction above on the formation of DL formulas.

[3] The denotations of the numerals form a copy of $\mathbb{N}$ in a structure satisfying Peano's Third and Fourth Axioms, which we could include here trivially. However, (3) remains true in structures that identify the values of some distinct numerals!

Table 1. Pratt-Segerberg's Dynamic Logic

First-order logic		
Modality:	Generalization:	$\dfrac{\vdash \varphi}{\vdash [\alpha]\varphi}$
	Distribution:	$[\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$
Atomic Programs:	Assignment:	$[x := \mathbf{t}]\,\varphi \leftrightarrow \{\mathbf{t}/x\}\varphi$ $\mathbf{t}$ free for x in φ
Basic constructs:	Test:	$[?\chi]\varphi \leftrightarrow (\chi \to \varphi)$
	Composition:	$[\alpha;\beta]\varphi \leftrightarrow [\alpha][\beta]\varphi$
	Union:	$[\alpha \cup \beta]\varphi \leftrightarrow [\alpha]\varphi \wedge [\beta]\varphi$
Iteration:	Invariance (Folding):	$\dfrac{\vdash \varphi \to [\alpha]\,\varphi}{\vdash \varphi \to [\alpha^*]\,\varphi}$
	Unfolding:	$[\alpha^*]\varphi \to \varphi \wedge [\alpha][\alpha^*]\varphi$

From the Invariance Rule we obtain the Schema of Induction:

$$\psi \wedge [\alpha^*](\psi \to [\alpha]\,\psi) \to [\alpha^*]\,\psi)$$

Indeed, taking $\varphi \equiv \psi \wedge [\alpha^*](\psi \to [\alpha]\psi)$, we have $\vdash \varphi \to [\alpha]\varphi$ using the remaining axioms and rules, and so $\varphi \to [\alpha^*]\varphi$ by Iteration. The Induction template above readily follows.

Also, the converse of the Unfolding Schema is easily provable: Taking $\psi \equiv \varphi \wedge [\alpha;\alpha^*]\varphi$, we have $\vdash \psi \to [\alpha]\psi$, by Unfolding and the modality rules, yielding $\psi \to [\alpha^*]\psi$ by Invariance, while $[\alpha^*]\psi \to [\alpha^*]\varphi$ by Iteration and modality rules.[4]

A *V-theory* is a set of closed first-order *V*-formulas. Given a *V*-theory $\mathbf{T}$ we write $\mathbf{DL}(\mathbf{T})$ for the deductive formalism **DL** augmented with the formulas in $\mathbf{T}$ as axioms. We refer to $\mathbf{T}$ as the *background theory.*

A straightforward induction on proofs establishes the soundness of **DL**:

Theorem 1. (Soundness of DL) *Let* $\mathbf{T}$ *be a V-theory,* φ *a DL V-formula. Suppose* $\mathbf{DL}(\mathbf{T}) \vdash \varphi$*. Then* φ *is true in every model of* $\mathbf{T}$.

2 Inductive Completeness

2.1 Inductive Definition of Program Semantics

Generic methods for associating to a given collection of inductive (i.e. generative) definitions first-order inductive theories are well-known. The inductive definition of the semantics of regular programs has a particularly simple form, using atomic production rules, i.e. natural-deduction inferences with atomic premises and conclusion, as follows.

[4] Note that this argument uses Invariance for modal formulas, and is not available when the Invariance Rule is restricted to first-order formulas.

For a list $\boldsymbol{x} = (x_1, \ldots, x_n)$ of variables, let $\mathbf{P}[\boldsymbol{x}]$ consist of the regular V-programs with all assigned variables among $x_1 \ldots x_n$. Note that if α is such a program, then so are all its subprograms. Given a V-structure $\mathcal{S}$, each program $\alpha \in \mathbf{P}[\boldsymbol{x}]$ defines a $2n$-ary relation $[\![\alpha]\!]_{\mathcal{S}}$ on the universe $|\mathcal{S}|$ of $\mathcal{S}$, that holds between $\boldsymbol{a}, \boldsymbol{b} \in |\mathcal{S}|^n$ iff α has a complete execution that starts with $\boldsymbol{x}$ bound to $\boldsymbol{a}$, and terminates with $\boldsymbol{x}$ bound to $\boldsymbol{b}$.

For $n \geqslant 1$, let $\hat{V}^n$ be the expansion of the underlying vocabulary V with $2n$-ary relational identifiers $\mathtt{M}^n_\alpha$ for each $\alpha \in \mathbf{P}[\boldsymbol{x}]$. The intent is that $\mathtt{M}^n_\alpha$ denotes, in each V-structure $\mathcal{S}$, the relation $[\![\alpha]\!]_{\mathcal{S}}$ above. We omit the superscript n throughout when in no danger of confusion.

An inductive definition of $[\![\alpha]\!]$, uniform for all V-structures, is given by generative clauses that can be rendered by the following atomic rule-templates, which can be construed as natural-deduction rules.

ASSIGNMENT
$$\frac{}{\mathtt{M}_{x_i:=\mathbf{t}[\boldsymbol{x}]}(\boldsymbol{u}, \boldsymbol{u}_{i\leftarrow\mathbf{t}})}$$
where $\boldsymbol{u}_{i\leftarrow\mathbf{t}}$ is $u_1 \ldots u_{i-1}, \mathbf{t}[\boldsymbol{u}], u_{i+1} \ldots u_n$

TEST
$$\frac{\varphi[\boldsymbol{u}]}{\mathtt{M}_{?\varphi}(\boldsymbol{u}, \boldsymbol{u})}$$

COMPOSITION
$$\frac{\mathtt{M}_\beta(\boldsymbol{u}, \boldsymbol{w}) \quad \mathtt{M}_\gamma(\boldsymbol{w}, \boldsymbol{v})}{\mathtt{M}_{\beta;\gamma}(\boldsymbol{u}, \boldsymbol{v})}$$

BRANCHING
$$\frac{\mathtt{M}_\beta(\boldsymbol{u}, \boldsymbol{v})}{\mathtt{M}_{\beta\cup\gamma}(\boldsymbol{u}, \boldsymbol{v})} \qquad \frac{\mathtt{M}_\gamma(\boldsymbol{u}, \boldsymbol{v})}{\mathtt{M}_{\beta\cup\gamma}(\boldsymbol{u}, \boldsymbol{v})}$$

ITERATION
$$\frac{}{\mathtt{M}_{\beta^*}(\boldsymbol{u}, \boldsymbol{u})} \qquad \frac{\mathtt{M}_\beta(\boldsymbol{u}, \boldsymbol{w}) \quad \mathtt{M}_{\beta^*}(\boldsymbol{w}, \boldsymbol{v})}{\mathtt{M}_{\beta^*}(\boldsymbol{u}, \boldsymbol{v})}$$

2.2 Expressing Program Properties

It is easy to see that, modulo the intended reading of the identifiers $\mathtt{M}_\alpha$, the expressive power of the $\hat{V}$-formulas is identical to the expressive power of DL formulas over the base vocabulary V. To avoid clutter we posit that all programs are in $\mathbf{P}[\boldsymbol{x}]$, i.e. all assigned-variables are among $x_1 \ldots x_n$.

Each DL V-formula φ can be expressed as a $\hat{V}$-formula $\varphi^\sharp$, defined by structural recurrence on φ. For φ modal-free we take of course $\varphi^\sharp$ to be φ itself. If φ is $[\alpha]\varphi_0$, then $\varphi^\sharp$ is $\forall v_1 \ldots v_n\ \mathtt{M}_\alpha(\boldsymbol{x}, \boldsymbol{v}) \to \{\boldsymbol{v}/\boldsymbol{x}\}\varphi_0^\sharp$. Finally, we let $\sharp$ commute with the first-order logical operations; for instance, $(\varphi_0 \wedge \varphi_1)^\sharp$ is $\varphi_0^\sharp \wedge \varphi_1^\sharp$, and $(\forall u\varphi)^\sharp$ is $\forall u(\varphi^\sharp)$ (recall that assigned-to variables are not quantified.)

Conversely, each $\hat{V}$-formula ψ is expressible as a DL V-formula $\psi^\natural$, defined by structural recurrence on ψ. If ψ is a V-formula, we defined $\psi^\natural$ to be ψ. If ψ is $\mathtt{M}_\alpha(\mathbf{t}, \mathbf{s})$ then $\psi^\natural$ is $\boldsymbol{x} = \mathbf{t} \to \langle\alpha\rangle(\boldsymbol{x} = \mathbf{s})$. Again, we let $\natural$ commute with connectives and quantifiers.

Observe that these interpretations are *sound* in the following sense.

Theorem 2. *For every V-structure $\mathcal{S}$, if $\hat{\mathcal{S}}$ is the $\hat{V}$-expansion of $\mathcal{S}$ in which each $\mathtt{M}_\alpha$ is interpreted as the denotational semantics of α, then for every DL V-formula φ, $\hat{\mathcal{S}} \models \varphi \leftrightarrow \varphi^\sharp$, and for every $\hat{V}$-formula ψ, $\hat{\mathcal{S}} \models \psi \leftrightarrow \psi^\natural$.*

Since $\hat{\mathcal{S}}$ is trivially conservative over $\mathcal{S}$ for DL V-formulas, we conclude

Corollary 1. *For every DL formula φ, $\mathcal{S} \models \varphi$ iff $\hat{\mathcal{S}} \models \varphi^\sharp$.*

We prove below (Proposition 4) that the equivalence $\varphi \leftrightarrow \varphi^{\sharp\natural}$ is in fact provable in **DL**.

2.3 The Inductive Theory of Regular Programs

The generative rules above, for inductively defining program semantics, bound the interpretation of the relation-identifiers $\mathtt{M}_\alpha$ from below. Bounding inductively generated sets from above, namely as the *minimal* relations closed under the generative clauses, is a second-order condition which has no first-order axiomatization (except for degenerated cases). However, we can approximate that delineation as the minimal one among a given collection of *definable* relations. Namely, the *deconstruction* template for $\mathtt{M}_\alpha$ states that $\mathtt{M}_\alpha$ is contained in every definable relation closed under the generative rules for $\mathtt{M}_\alpha$. This is analogous to the familiar deconstruction for the set $\mathbb{N}$ of natural numbers: With N as unary relational-identifier, the generative clauses are

$$\frac{}{N(\mathtt{0})} \quad \text{and} \quad \frac{N(x)}{N(\mathtt{s}(x))}$$

yielding the Deconstruction template

$$\frac{N(x) \quad \varphi[\mathtt{0}] \quad \begin{array}{c}\varphi[z]\\ \cdots \\ \varphi[\mathtt{s}z]\end{array}}{\varphi[x]}$$

(assumption $\varphi[z]$ is discharged,
z not free in other open assumptions)

that is, the natural-deduction rule of induction on $\mathbb{N}$ [19].

Analogously, the DECONSTRUCTION Rule for the iteration construct $*$ should be

$$\frac{\mathtt{M}_{\beta^*}(\boldsymbol{s},\boldsymbol{t}) \quad \varphi[\boldsymbol{u},\boldsymbol{u}] \quad \begin{array}{c}\mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w}) \quad \varphi[\boldsymbol{w},\boldsymbol{v}]\\ \cdots \\ \varphi[\boldsymbol{u},\boldsymbol{v}]\end{array}}{\varphi[\boldsymbol{s},\boldsymbol{t}]}$$

(assumptions $\mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w})$ and $\varphi[\boldsymbol{w},\boldsymbol{v}]$ are discharged,
$\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}$ not free in other open assumptions)

The formula φ above is the *eigen-formula* of the inference.

A related, more practical, rule is

$$\textsc{Invariance}\qquad \frac{\begin{array}{c}\mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w})\\ \cdots\\ \mathtt{M}_{\beta^*}(\boldsymbol{s},\boldsymbol{t}) \quad \psi[\boldsymbol{u}] \to \psi[\boldsymbol{w}]\end{array}}{\psi[\boldsymbol{s}] \to \psi[\boldsymbol{t}]}$$

(assumption $\mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w})$ is discharged
$\boldsymbol{u},\boldsymbol{w}$ not free in other open assumptions)

Put differently,

$$\frac{\forall \boldsymbol{u},\boldsymbol{w} \quad \psi[\boldsymbol{u}] \wedge \mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w}) \to \psi[\boldsymbol{w}]}{\forall \boldsymbol{y},\boldsymbol{z} \quad \psi[\boldsymbol{y}] \wedge \mathtt{M}_{\beta^*}(\boldsymbol{y},\boldsymbol{z}) \to \psi[\boldsymbol{z}]}$$

However, we have

Proposition 2. *The rules* DECONSTRUCTION *and* INVARIANCE *are equivalent.*

Proof. Posit DECONSTRUCTION, and assume the premises of INVARIANCE. Then the three premises of DECONSTRUCTION hold with $\varphi[\boldsymbol{x},\boldsymbol{y}]$ taken as $\psi[\boldsymbol{x}] \to \psi[\boldsymbol{y}]$. We thus obtain $\psi[\boldsymbol{s}] \to \psi[\boldsymbol{t}]$, as required.

Conversely, posit INVARIANCE, and assume the premises of DECONSTRUCTION. Then the premises of INVARIANCE hold with $\psi[\boldsymbol{x}]$ taken to be $\neg\varphi[\boldsymbol{x},\boldsymbol{t}]$. Thus INVARIANCE yields $\psi[\boldsymbol{s}] \to \psi[\boldsymbol{t}]$, i.e. $\varphi[\boldsymbol{t},\boldsymbol{t}] \to \varphi[\boldsymbol{s},\boldsymbol{t}]$. Since we have $\varphi[\boldsymbol{t},\boldsymbol{t}]$ by the second premise of DECONSTRUCTION (recall that $\boldsymbol{u}$ is not free in assumptions), we obtain $\varphi[\boldsymbol{s},\boldsymbol{t}]$, as required. ■

Note that deconstruction rules for the remaining program constructs are degenerate, in the sense that they are equivalent to explicit definitions. For example, the Deconstruction of composition, combined with the Composition Rule, yield an explicit definition of $\mathtt{M}_{\beta;\gamma}$. More generally, $\mathtt{M}_\alpha$ can be explicitly defined in terms of components of α, for all non-loop programs α:

- $\mathtt{M}_{x_i:=\mathbf{t}}(\boldsymbol{u},\boldsymbol{v}) \;\leftrightarrow\; (v_i = \mathbf{t}[\boldsymbol{u}] \wedge \bigwedge_{j\neq i} v_j = u_j)$.
- $\mathtt{M}_{?\varphi}(\boldsymbol{u},\boldsymbol{v}) \;\leftrightarrow\; (\varphi \wedge \boldsymbol{v} = \boldsymbol{u})$
- $\mathtt{M}_{\beta;\gamma}(\boldsymbol{u},\boldsymbol{v}) \leftrightarrow \exists \boldsymbol{w}\, \mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{w}) \wedge \mathtt{M}_\gamma(\boldsymbol{w},\boldsymbol{v})$
- $\mathtt{M}_{\beta\cup\gamma}(\boldsymbol{u},\boldsymbol{v}) \;\leftrightarrow\; \mathtt{M}_\beta(\boldsymbol{u},\boldsymbol{v}) \vee \mathtt{M}_\gamma(\boldsymbol{u},\boldsymbol{v})$

We write $\mathbf{Ind}^n$ for the inductive theory given by the universal closure of the formulas above (recall that our $\mathtt{M}_\alpha$'s are for programs α with variables among $x_1,\ldots,x_n$). We omit the superscript n when in no danger of confusion.

2.4 Inductive Soundness of DL

Clearly, the deductive calculus **DL** is semantically sound (Theorem 1). Only slightly less trivial is the observation that it is sound for **Ind**:

Theorem 3. *If* $\mathbf{DL}(\mathbf{T}) \vdash \varphi$ *then* $\mathbf{T} + \mathbf{Ind} \vdash \varphi^\sharp$.

Proof. Induction on proofs in $\mathbf{DL}(\mathbf{T})$.

Consider, for example, an instance of the INVARIANCE Rule, deriving $\psi \to [\alpha^*]\ \psi$ from $\psi \to [\alpha]\ \psi$. By IH we have

$$\mathbf{T} + \mathbf{Ind} \vdash \psi^\sharp[\boldsymbol{u}] \wedge \mathtt{M}_\alpha(\boldsymbol{u}, \boldsymbol{v}) \ \to\ \psi^\sharp[\boldsymbol{v}]$$

which by the INVARIANCE Rule of **Ind** yields

$$\mathbf{T} + \mathbf{Ind} \vdash \psi^\sharp[\boldsymbol{u}] \wedge \mathtt{M}_{\alpha^*}(\boldsymbol{u}, \boldsymbol{v}) \ \to\ \psi^\sharp[\boldsymbol{v}]$$

For another example, consider the Generalization rule, deriving $\vdash [\alpha]\varphi$ from $\vdash \varphi$. By IH the premise implies the provability in $\mathbf{T} + \mathbf{Ind}$ of $\varphi^\sharp$, from which the provability of $([\alpha]\varphi)^\sharp$, i.e. $\mathtt{M}_\alpha(\boldsymbol{x}, \boldsymbol{u}) \to \{\boldsymbol{u}/\boldsymbol{x}\}\varphi^\sharp$, follows trivially. ■

2.5 Inductive Completeness of DL

The main interest in the inductive theory **Ind** is its relation to **DL**, namely the inductive completeness of **DL**:

Theorem 4. *For all V-theories* $\mathbf{T}$*, if φ is a DL V-formula, and* $\mathbf{T} + \mathbf{Ind} \vdash \varphi^\sharp$*, then* $\mathbf{DL}(\mathbf{T}) \vdash \varphi$.

The proof of Theorem 4 will use the interpretation $\psi \mapsto \psi^\natural$ defined above.

Proposition 3. *Let* $\mathbf{T}$ *be a V-theory, and ψ a $\hat{V}$-formula. If* $\mathbf{T} + \mathbf{Ind} \vdash \psi$*, then* $\mathbf{DL}(\mathbf{T}) \vdash \psi^\natural$.

Proof. The proof is by induction on natural-deduction derivations of $\mathbf{T} + \mathbf{Ind}$. More precisely, if φ is provable in $\mathbf{T} + \mathbf{Ind}$ from assumptions Γ, then $\varphi^\natural$ is provable from $\Gamma^\natural$ in $\mathbf{DL}(\mathbf{T})$.

It is easy to verify that the rules of **Ind** translate correctly. Consider, for example, the Iteration rules. If ψ is $\mathtt{M}_{\beta^*}(\boldsymbol{u}, \boldsymbol{u})$ then $\psi^\natural$ is $\boldsymbol{x} = \boldsymbol{u} \to \langle \beta^* \rangle\, \boldsymbol{x} = \boldsymbol{u}$, which is readily provable in **DL** using Unfolding. To tackle the second Iteration rule of **Ind**, suppose that ψ is $\mathtt{M}_{\beta^*}(\boldsymbol{u}, \boldsymbol{u})$, and is derived from $\mathtt{M}_\beta(\boldsymbol{u}, \boldsymbol{w})$ and $\mathtt{M}_{\beta^*}(\boldsymbol{w}, \boldsymbol{u})$. Assume now that the premises translate correctly, i.e. the formulas

$$\boldsymbol{x} = \boldsymbol{u} \to \langle \beta \rangle\, \boldsymbol{x} = \boldsymbol{w} \tag{5}$$

and

$$\boldsymbol{x} = \boldsymbol{w} \to \langle \beta^* \rangle\, \boldsymbol{x} = \boldsymbol{v} \tag{6}$$

are given. From (6) we get

$$\langle \beta \rangle\ \boldsymbol{x} = \boldsymbol{w} \to \langle \beta \rangle\, \langle \beta^* \rangle\, \boldsymbol{x} = \boldsymbol{v}$$

which readily yields in **DL**

$$\langle \beta \rangle\ \boldsymbol{x} = \boldsymbol{w} \to \langle \beta^* \rangle\, \boldsymbol{x} = \boldsymbol{v}$$

Combined with (5) we get $\psi^\natural$.

It is also easy to verify that the inference rules of first order logic preserve the translation. ■

Lemma 1. *Let $\boldsymbol{x}$ be the assigned-to variables in a DL-formula $\varphi \equiv \varphi(\boldsymbol{x})$. The following is provable in* **DL***.*

$$[\alpha]\varphi \quad \leftrightarrow \quad \forall \boldsymbol{v} \; (\; \langle \alpha \rangle \, (\boldsymbol{x} = \boldsymbol{v}) \rightarrow \varphi(\boldsymbol{v}))$$

The proof is straightforward by induction on α. Only the iteration case $\alpha = \beta^*$ is non-trivial.

Proposition 4. *For all DL-formulas φ the following is provable in* **DL***:*

$$(\varphi^\sharp)^\natural \quad \leftrightarrow \quad \varphi$$

Proof. We use structural induction on φ. The only non-trivial case is where φ is of the form $[\alpha]\psi$. Then

$$\begin{array}{lll} (\varphi^\sharp)^\natural \equiv (\forall \boldsymbol{v} \; (\mathtt{M}_\alpha(\boldsymbol{x}, \boldsymbol{v}) \rightarrow \varphi^\sharp(v)))^\natural & & \\ \leftrightarrow \forall \boldsymbol{v} \; (\langle \alpha \rangle (\boldsymbol{x} = \boldsymbol{v}) \rightarrow \varphi^{\sharp\natural}(v)) & & \\ \leftrightarrow [\alpha]\varphi^{\sharp\natural} & \text{(Lemma 1)} & \\ \leftrightarrow [\alpha]\varphi(v) & \text{(IH and the Distribution Rule)} & \blacksquare \end{array}$$

Proof of Theorem 4. Suppose $\mathbf{T} + \mathbf{Ind} \vdash \varphi^\sharp$. Then, by Proposition 3, $\mathbf{DL}(\mathbf{T}) \vdash (\varphi^\sharp)^\natural$, from which $\mathbf{DL}(\mathbf{T}) \vdash \varphi$ follows by Proposition 4. $\blacksquare$

Note that this proof is different from the proof of the corresponding result for Hoare's Logic [18, Theorem 2.4], stating the soundness and completeness of Hoare's Logic for a theory $\mathbf{Ind}_0$, obtained by restricting the Deconstruction Rule to eigen-formulas φ in the vocabulary V. However, the latter theorem can be obtained from the proof presented here, using the conservation theorem [16, Theorem 7], which states that Dynamic Logic is conservative over Hoare's Logic when Invariance is restricted to first-order formulas.

3 Relative Completeness and Arithmetical Completeness

3.1 Failure of Relative Completeness for Termination Assertions

Relative completeness in the sense of Cook fails for **DL**, since there is no way to infer that a termination assertion of the form $\langle \alpha^* \rangle \mathtt{true}$ is true in a given structure from any collection of first-order formulas true in the structure. This reflects a gap between the semantics of program convergence, which is anchored in the intended data (e.g. $\mathbb{N}$), and the semantics of data in the background theory, which might include non-standard elements. To illustrate, consider the termination assertion

$$P \; \rightarrow \; \langle \, (x := p(x))^* \rangle \; (x = 0) \tag{7}$$

where P is some finite axiomatization of arithmetic,[5] that includes the definition of p as cut-off predecessor: $p(0) = 0$, $p(s(x)) = x$. Since there are non-standard models

[5] Take, for example, Peano's Arithmetic with induction up to some fixed level Σ_n in the arithmetical hierarchy.

of P, with elements that are not denotations of numerals, the assertion (7) is not valid. It follows that (7), although trivially true in the intended structure, cannot be proved in any DL formalism which is sound for all structures, such as **DL**.

This remains true even if we augment **DL** with axioms for inductive data, such as Peano's Arithmetic, since the semantics of program convergence will remain different from the semantics of counting in the grafted first-order theory.

Of course, (7) can be proved by induction on the formula

$$\boldsymbol{\varphi}[n] \quad \equiv \qquad (x = \bar{n}) \to \langle\, (x := p(x))^* \rangle\, (x{=}0)$$

but $\boldsymbol{\varphi}$ is not a first-order formula, and so this instance of induction is not part of the background theory. As long as the background theory has no direct access to modal formulas, it cannot be used to derive termination assertions whose truth depends on the inductive data in hand.

3.2 Schematic Relative Completeness

The remarks above suggest a way to modify Cook's notion of relative completeness, so as to apply to Dynamic Logic. Define a *V-schema* to be a closed first-order formula φ in V augmented with additional identifiers (place-holders) for relations. For example, the Schema of Induction over $\mathbb{N}$ is

$$P(0) \ \wedge\ (\forall x\ P(x) \to P(s(x))) \ \to\ \forall x\ P(x)$$

Here 0 and s (denoting the successor function) are part of the given vocabulary, whereas P is a new, unary, place-holder. Note that a V-formula is trivially also a V-schema.

A *DL-instance* of a schema φ is the result of replacing in φ each such place-holder identifier, of arity k say, by a k-ary DL-predicate, i.e. $\lambda x_1 \ldots x_k.\, \psi$, where ψ is a DL-formula (and bound variables in ψ are renamed to avoid scoping clashes). For example, if $\psi[u, v, w]$ is a formula with variables u, v, w free, then the instance of Induction for $\lambda v.\psi$ is

$$\psi[u, 0, w] \ \wedge\ (\forall x\ \psi[u, x, w] \to \psi[u, s(x), w]) \ \to\ \forall x\ \psi[u, x, w]$$

A schema φ is *true* in a V-structure $\mathcal{S}$ if it is true regardless of the interpretation of each k-ary place-holders as a k-ary relation over the universe $|\mathcal{S}|$ of $\mathcal{S}$; i.e., if it is true in every expansion of $\mathcal{S}$. For example, the schema of induction above is true in the structure $\mathbb{N}$ (with zero and successor). If $\mathcal{S}$ is a V-structure, then the *schematic theory of* $\mathcal{S}$ consists of the V-schemas true in $\mathcal{S}$. We write $\mathbf{DL}(\mathcal{S})$ for the formalism **DL** augmented with all DL-instances of *schemas* true in $\mathcal{S}$.

We continue to refer to Cook's notion of expressiveness: a V-structure $\mathcal{S}$ is expressive if for every program α there is a V-formula ξ_α equivalent in $\mathcal{S}$ to $\langle\alpha\rangle(\boldsymbol{x} = \boldsymbol{v})$.

Theorem 5. **DL** *is (schematic) relatively complete in the following sense: for every expressive structure* $\mathcal{S}$ *and DL formula* φ*, if* $\mathcal{S} \models \varphi$*, then* φ *is provable in* $\mathbf{DL}(\mathcal{S})$.

A proof of Theorem 5 will be given elsewhere. The core idea is to emulate the proof above of Theorem 4, with the formulas ξ_α replacing $\mathtt{M}_\alpha$. Each formula ξ_{β^*} satisfies the

Iteration Rule for β, read as a schema, and so the schematic theory of a structure makes it possible to use the formulas ξ_α in place of $\mathtt{M}_\alpha$.

In [18, §2.5] we showed that Cook's notion of relative completeness is the local projection (to expressive structures) of the inductive completeness theorem proved there for Hoare's Logic. Analogously, Theorem 5 provides a notion of relative completeness which is the projection of inductive completeness of **DL**.

3.3 Arithmetical Completeness

The termination of imperative programs is commonly proved by the Variance Method: one attaches to each instance of a looping construct in the program (such as a **while** loop or a recursive procedure) a parameter ranging over the field A of a well-founded relation $\succ$, and shows that each cycle reduces that parameter under $\succ$:

$$\frac{\vdash \boldsymbol{\varphi} \wedge a = x \rightarrow \langle \boldsymbol{\alpha} \rangle \boldsymbol{\varphi} \wedge a \succ x}{\vdash \boldsymbol{\varphi} \wedge A(x) \rightarrow \langle \boldsymbol{\alpha}^* \rangle \boldsymbol{\varphi}}$$

(a not assigned-to in $\boldsymbol{\alpha}$)

Taking $\succ$ to be the natural order on $\mathbb{N}$, the Variance Rule yields the Convergence Rule of [10]:

$$\frac{\vdash \boldsymbol{\varphi}(\mathbf{s}x) \rightarrow \langle \boldsymbol{\alpha} \rangle \boldsymbol{\varphi}(x)}{\vdash \boldsymbol{\varphi}(x) \wedge N(x) \rightarrow \langle \boldsymbol{\alpha}^* \rangle \boldsymbol{\varphi}(\mathbf{0})}$$

(x not assigned-to in $\boldsymbol{\alpha}$
N interpreted as $\mathbb{N}$)

Note that this rule fuses the interpretation of counting in the background theory and in the program semantics, thus forcing the numeric variables to range precisely over the natural numbers. In particular, the rule is sound only for structures in which N is interpreted as $\mathbb{N}$, structures dubbed *arithmetical* in [10].

The rationale of [10] for the Convergence Rule was ostensibly to establish a completeness property for DL, analogous to Cook's Relative Completeness Theorem for Hoare-style logics. However, Cook's notion of relative completeness is itself problematic, and the arithmetic completeness of [10] faces additional pitfalls. One is the fact that the Convergence Rule is sounds only for a special class of structures, which is itself not first-order axiomatizable.

Also, whereas Hoare's Logic for Partial-correctness assertions is based on a formal separation between rules for programs (Hoare's rules) and rules for data (the background theory), the essential feature of the Convergence Rule is that it fuses the two. When programs and data are fused, and programs and their semantics are codable by data (as is the case in arithmetic structures), the very rationale for factoring out rules for programs from axioms for data is weakened, and one might arguably reason directly about programs in a first-order theory, as done for example in [1]. Needless to say, proving program termination by the Variance Method is of immense practical importance, and our contention is simply that it is a mathematical tool (referring to particular structures, i.e. well-orderings) rather than a logical principle. The misfit of the Convergence Rule in the rest of the axiomatics of DL is indeed manifest in the rather arbitrary choice of the natural numbers as inductive measure.

The schematic relative completeness of DL clarifies the status of the Convergence Rule, and more generally of the notion of arithmetical completeness. As observed above, our ability to use induction on $\mathbb{N}$ to prove termination (and more complex) assertions in DL is hindered by the restriction of induction to first-order formulas. By referring to data-induction as a schema, which can be instantiated to any DL formula, we recover the freedom of action that we have in reasoning about DL formulas, and which Harel's Convergence Rule is providing in an *ad hoc* fashion, and only for arithmetical structures. Indeed, the Convergence Rule is merely a syntactic variant of the schema of induction for $\mathbb{N}$. The reference to $\mathbb{N}$, however, is off target, as our result on schematic relative completeness shows: when $\mathtt{M}_{\alpha^*}$ is expressed in a structure by a first-order formula ξ_α, as it should in an expressive structure, then Invariance (or — equivalently — Induction) for ξ_α is the relevant true schema, and there is no need to invoke the natural numbers.

True, schematic relative completeness refers to valid schemas of the structure in hand, and recognizing these is complete-Π^1_1 (for the natural numbers). But the entire framework of Cook's relative completeness is highly non-effective to begin with.

4 Summary and Directions

Our inductive theories provide a generic framework for explicit reasoning about programs (i.e. without using the modal operators of logics of programs). They build directly on the inductive definition of program semantics, as opposed to frameworks proposed in the past, based on natural numbers (see e.g. [25]). This approach provides a close match between the semantic definition of programming languages, a specification languages for program behavior, and formal verification theories for them. Correspondingly, the notion of inductive completeness establishes a natural match between a proposed (modal) logic of programs and the programming language to which it refers.

Moreover, inductive theories for programming languages provide a setting in which tools of automated deduction can be applied directly, notably various methods for identifying inductive assertions in proofs.

The methods and results presented here can be applied to richer programming paradigms, such as recursive procedures, parallelism, and object-oriented programming (compare [21] and [2]). Of course, programming constructs that use dynamic memory allocation, say recursive procedures with local variables, would require more expressive logical constructs, such as relations with undetermined arities (or, equivalently, direct reference to lists or streams). However, these do not seem to raise any conceptual difficulty. In particular, we would expect to obtain inductively complete logics for programming languages for which relative completeness (even for Hoare's logic, let alone Dynamic Logic) is excluded by [4].

References

1. Andreka, H., Nemeti, I., Sain, I.: A complete logic for reasoning about programs via nonstandard model theory, Parts I and II. Theoretical Computer Science 17, 193–212, 259–278 (1982)
2. Beckert, B., Platzer, A.: Dynamic logic with non-rigid functions. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS, vol. 4130, pp. 266–280. Springer, Heidelberg (2006)

3. Blass, A., Gurevich, Y.: The underlying logic of Hoare logic. Current Trends in Theoretical Computer Science, 409–436 (2001)
4. Clarke, E.: Programming language constructs for which it is impossible to obtain good Hoare-like axioms. J. ACM 26, 129–147 (1979)
5. Cook, S.A.: Soundness and completeness of an axiom system for program verification. SIAM J. Computing 7(1), 70–90 (1978)
6. Csirmaz, L.: Programs and program verification in a general setting. Theoretical Computer Science 16, 199–210 (1981)
7. Feferman, S.: Formal theories for transfinite iterations of generalized inductive definitions and some subsystems of analysis. In: Intuitionism and Proof Theory, pp. 303–326. North-Holland, Amsterdam (1970)
8. Feferman, S., Sieg, W.: Iterated inductive definitions and subsystems of analysis. In: Iterated Inductive Definitions and Subsystems of Analysis: Recent Proof-Theoreitc Studies. LNM, vol. 897, pp. 16–77. Springer, Berlin (1981)
9. Harel, D., Meyer, A., Pratt, V.: Computability and completeness in logics of programs. In: Proceedings of the ninth symposium on the Theory of Computing, Providence, pp. 261–268. ACM, New York (1977)
10. Harel, D.: First-Order Dynamic Logic. LNCS, vol. 68. Springer, Heidelberg (1979)
11. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
12. Honsell, F., Miculan, M.: A natural deduction approach to dynamic logics. In: Berardi, S., Coppo, M. (eds.) TYPES 1995. LNCS, vol. 1158, pp. 165–182. Springer, Heidelberg (1996)
13. Kreisel, G.: Generalized inductive definitions. Reports for the seminar on foundations of analysis, Stanford, vol. 1 §3 (1963)
14. Kreisel, G.: Mathematical logic. In: Saaty, T. (ed.) Lectures on Modern Mathematics, vol. III, pp. 95–195. John Wiley, New York (1965)
15. Leivant, D.: Logical and mathematical reasoning about imperative programs. In: Conference Record of the Twelfth Annual Symposium on Principles of Programming Languages, pp. 132–140. ACM, New York (1985)
16. Leivant, D.: Partial correctness assertions provable in dynamic logics. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 304–317. Springer, Heidelberg (2004)
17. Leivant, D.: Matching explicit and modal reasoning about programs: A proof theoretic delineation of dynamic logic. In: Twenty-first Symposium on Logic in Computer Science (LiCS 2006), Washington, pp. 157–166. IEEE Computer Society Press, Los Alamitos (2006)
18. Leivant, D.: Inductive completeness of logics of programs. In: Proceedings of the Workshop on Logical Frameworks and Meta-Languages (to appear, 2008)
19. Martin-Löf, P.: Hauptsatz for the intuitionistic theory of iterated inductive definitions. In: Fenstad, J.E. (ed.) Proceedings of the Second Scandinavian Logic Symposium, pp. 63–92. North-Holland, Amsterdam (1971)
20. Mirkowska, G.: On formalized systems of algorithmic logic. Bull. Acad. Polon. Sci. 19, 421–428 (1971)
21. Nishimura, H.: Arithmetical completeness in first-order dynamic logic for concurrent programs. Publ. Res. Inst. Math. Sci. 17, 297–309 (1981)
22. Pratt, V.: Semantical considerations on Floyd-Hoare logic. In: Proceedings of the seventeenth symposium on Foundations of Computer Science, Washington, pp. 109–121. IEEE Computer Society, Los Alamitos (1976)
23. Sain, I.: An elementary proof for some semantic characterizations of nondeterministic Floyd-Hoare logic. Notre Dame Journal of Formal Logic 30, 563–573 (1989)
24. Segerberg, K.: A completeness theorem in the modal logic of programs (preliminary report). Notics of the American Mathematical Society 24(6), A–552 (1977)
25. Szalas, A.: On strictly arithmetical completeness in logics of programs. Theoretical Computer Science 79(2), 341–355 (1991)