Complete Proof Systems for Algebraic Simply-Typed Terms

Stavros S. Cosmadakis
New York University

Abstract

We show that reasoning by case analysis (on whether subprograms diverge or converge) is complete for proving PCF observational congruences of algebraic terms. The latter are applicative combinations of first-order variables and a constant Ω denoting a diverging program of base type. A restricted version of the logic is complete for proving equality of algebraic terms in the full continuous type hierarchy (equivalently, observational congruence in PCF with parallel conditional). We show that provability in the latter logic is in co-NP. We also give complete equational proof systems for a subclass of algebraic terms; provability in these systems is in linear time.

1 Introduction

The correctness of program transformations (used, for instance, by optimizing compilers) is generally established by a global analysis of the program. In some cases a local analysis, taking into account only the program fragment transformed, is sufficient. It seems important to understand these cases; one motivating consideration might be the optimization of separately compiled subroutines or modules. We are interested in developing a theory of correctness of such local transformations. We focus on formal techniques which might be used (a) to establish the correctness of local program transformations in a systematic way or (b) to automatically effect such transformations.

We take as a paradigm the language PCF [9, 7, 11, 3], a simple statically-typed functional language with higher-order types. It consists of a set of terms extending the simply-typed λ-calculus by arithmetic and recursion constructs. Local interchangeability of code fragments is formalized in PCF by a notion of congruence wrto *observable behavior*; the latter is taken to be the printable output of the program, or the fact that the program diverges. Two terms P, Q are *observationally congruent* (written $P =_{PCF} Q$) iff, for any program context $C[]$, $C[P], C[Q]$ have the same observable behavior. Sound principles for reasoning about observational congruence in PCF can be developed by syntactic analysis of program evaluation; or by finding principles that are sound wrto equality in *computationally adequate semantics*, where equality of meaning (of two terms) implies observational congruence. For example, the (β) and (η) axioms from the simply-typed λ-calculus are both sound for $=_{PCF}$. This can be established syntactically via the *Context Lemma* [7]; or semantically by the computational adequacy of the *full continuous type hierarchy*, $\mathcal{C}$ [9].

PCF can express all partial recursive functions, and this makes the relation $=_{PCF}$ non-r.e.. Consequently, most reasonable formal systems for proving PCF observational congruences will only succeed in deriving a subset of the valid ones. A proof system is *complete* for a class of terms if

LISP 94 - 6/94 Orlando, Florida USA
© 1994 ACM 0-89791-643-3/94/0006..$3.50

it can prove every valid observational congruence between terms in the class. For the class of *pure* λ-terms (containing no constants), Statman's *1-Section Theorem* [12, 13] gives a criterion for completeness of $=_{\beta\eta}$ (equational reasoning with the (β) and (η) axioms) wrto a given model of the simply-typed λ-calculus. Congruence classes of $=_{PCF}$ provide such a model, and the criterion shows that $=_{\beta\eta}$ is complete for $=_{PCF}$ in the case of pure terms (Meyer). Complete proof systems can also be obtained from *fully abstract* semantics, where equality of meaning *coincides* with observational congruence. For example, the full continuous type hierarchy, $\mathcal{C}$, is fully abstract for PPCF, the version of PCF which includes a *parallel conditional* [9, 11]; equality in $\mathcal{C}$, $=_{\mathcal{C}}$, is the same as observational congruence in PPCF, $=_{PPCF}$. Since $=_{\beta\eta}$ is complete for proving equality of pure terms in $\mathcal{C}$ [8, 10, 12] it follows that it is also complete for $=_{PPCF}$ in the case of pure terms.

In this paper we consider the problem of developing complete proof systems for terms with constants. In particular, we include a constant Ω^ι (of base type) denoting a *diverging* PCF program (all such programs are observationally congruent). It can be seen that $=_{\beta\eta}$ is no longer complete for $=_{\mathcal{C}}$ nor for $=_{PCF}$, even if the terms considered have a very simple structure; namely, they are applicative combinations of first-order variables and Ω. We call such terms *algebraic*.

To illustrate, consider the equation

$$z^{\iota\to\iota\to\iota}x^\iota(z\Omega\Omega) = zx\Omega. \qquad (1)$$

Clearly, 1 is not provable in $=_{\beta\eta}$. It can be shown to be valid in $\mathcal{C}$ by *case analysis*: either $z\Omega\Omega = \Omega$, or $z\Omega\Omega \neq \Omega$, in which case the *flatness* of the base type implies that z is constant. Such analysis is familiar in systems for reasoning about PCF programs [6].

We give a formalization of case analysis in a sequent-style logic, and show that it is complete for $=_{\mathcal{C}}$, for the class of algebraic terms. The main idea of the formalization and of the completeness proof (a Henkin-type argument) was originally suggested by J. Riecke. We show further that the logic has a *subterm* property: proofs can be constrained to mention only subterms (of the terms) of the equation to be proved. This gives a straightforward decision procedure for provability, with an exponential time upper bound. By further analyzing proofs in the logic, we show that provability is in co-NP.

By the computational adequacy of $\mathcal{C}$ for PCF, any equation valid in $\mathcal{C}$ is a valid PCF observational congruence. The converse fails ($\mathcal{C}$ is not fully abstract for PCF [9, 11]), even for algebraic terms: the equation

$$z^{\iota\to\iota\to\iota}\Omega(zx^\iota\Omega) = z\Omega\Omega \qquad (2)$$

is not valid in $\mathcal{C}$[1], but is a valid PCF observational congruence. The latter can be shown by case analysis, using a property of PCF-definable functions pointed out by Milner [7]: z (considered as a function of two arguments) is either constant or strict in some argument. This *strict-or-constant* principle is not valid in $\mathcal{C}$, but is valid in the models described in [2, 4]. We point out the proof-theoretic relevance of the strict-or-constant principle by describing how to include it in the sequent-style logic given for $=_{\mathcal{C}}$. We show that the extended logic is complete for $=_{PCF}$, for the class of algebraic terms.

In attempting to go beyond algebraic terms, it is not straightforward how techniques such as term models and logical relations [5, 10, 14] can be applied to prove completeness for the class of all λ-terms with Ω (wrto $=_{\mathcal{C}}$, $=_{PCF}$). We propose instead to develop combinatorial arguments, along the lines of [8, 12]. Algebraic terms become important in this context (apart from being a natural class to consider): to prove the 1-Section Theorem, Statman shows that, given a model $\mathcal{M}$ of the simply-typed λ-calculus, if the implication

$$P =_{\mathcal{M}} Q \;\Rightarrow\; P =_{\beta\eta} Q$$

holds for closed terms P, Q of type $(\iota\to\iota\to\iota)\to\iota\to\iota$, then it holds for all closed terms P, Q (and thus $=_{\beta\eta}$ is complete for proving equalities in $\mathcal{M}$). The $\beta\eta$-normal forms of type $(\iota\to\iota\to\iota)\to\iota\to\iota$ are $\lambda z^{\iota\to\iota\to\iota}.\lambda x^\iota.t$, where t is any term generated by the grammar

$$t \;::=\; x \mid (ztt).$$

[1] z can be the *parallel-or* function

Therefore, $=_{\beta\eta}$ is complete wrto $=_{\mathcal{M}}$ if $\mathcal{M}$ distinguishes these algebraic terms [12]. Still, complete proof systems (for algebraic terms with Ω) designed around case analysis do not seem to provide a lead towards extending Statman's result to λ-terms with Ω: it seems essential to the proof of the result that $=_{\beta\eta}$ is an *equational* proof system. We give complete equational proof systems (for $=_{PCF}$ and $=_{\mathcal{C}}$) for *simple* algebraic terms; these are generated by $z^{\iota\to\iota\to\iota}$, x^{ι}, Ω^{ι} (compare with the grammar above) in which x occurs at most once (compare with 2, the algebraic counterexample to full abstraction of $\mathcal{C}$ for PCF). Provability in either system is decidable in linear time.

2 Preliminaries

2.1 Simply-typed lambda calculus

Types range over ι (the base type) and $\sigma \to \tau$ (the type of functions from σ to τ.) The set of simply-typed terms over a set of constants *Const* is given by the following inductive definition:

- x^{σ} is a variable and a term of type σ;
- $c^{\sigma} \in \mathit{Const}$ is a term of type σ;
- $(P\ Q)$ is a term of type τ if P has type $\sigma \to \tau$ and Q has type σ;
- $(\lambda x^{\sigma}.P)$ is a term of type $\sigma \to \tau$ if P has type τ.

2.2 PCF

The syntax of PCF follows [9]; PPCF has parallel conditional as an additional constant. The full set of PCF constants, with their types, are

- $0, 1, 2, \ldots$ of type ι;
- succ, pred of type $\iota \to \iota$;
- Y^{σ} of type $(\sigma \to \sigma) \to \sigma$;
- cond of type $\iota \to \iota \to \iota$ (sequential conditional);
- pcond of type $\iota \to \sigma \to \sigma \to \sigma$ for any type σ (parallel conditional);

The syntax of PCF is the set of simply-typed terms over these constants. The structured rewrite rules of Figure 1 completely characterize an interpreter for this language.

Definition 1 *$P \sqsubseteq_{PCF} Q$ iff, for any program context $C[]$, if $C[P]$ evaluates to n then $C[Q]$ evaluates to n.*

$P =_{PCF} Q$ iff $P \sqsubseteq_{PCF} Q$ and $Q \sqsubseteq_{PCF} P$.

3 Completeness and decidability for algebraic terms

Algebraic terms contain the constant Ω^{ι} and variables of *first-order* type, $\iota \to \cdots \to \iota$. They are generated by the grammar

$$P \ ::= \ \Omega \mid x^{\iota} \mid (f^{\iota^k \to \iota} P_1 \cdots P_k)$$

An *atomic formula* is either an *approximation* $P \sqsubseteq Q$ or a *convergence* $P\downarrow$, where P, Q are algebraic terms. A *sequent* has the form $\varphi \vdash \gamma$, where φ is a set of atomic formulas and γ is an atomic formula. The intended meaning of the sequent $\varphi \vdash \gamma$ is that the *conjunction* of the formulas in φ implies the formula γ. As usual, an empty conjuction denotes "true".

3.1 A complete decidable logic for $=_{\mathcal{C}}$

The logic $\mathcal{L}_{\mathcal{C}}$ has axioms and rules for approximations (Figure 2) and rules for convergences (Figure 3). Divergence of P is expressed by the atomic formula $P \sqsubseteq \Omega$. Case analysis on whether a term diverges or converges is formalized by the rule $(div - or - conv)$.

Theorem 2 (Riecke) *A sequent is valid in the model $\mathcal{C}$ iff it is provable in the logic $\mathcal{L}_{\mathcal{C}}$.*

Definition 3 *A sequent w is* subterm-*provable in $\mathcal{L}_{\mathcal{C}}$ iff there is a proof of w which uses only subterms of the terms occurring in w.*

Theorem 4 *A sequent is valid in $\mathcal{C}$ iff it is subterm-provable in $\mathcal{L}_{\mathcal{C}}$.*

$$
\begin{array}{lcl@{\qquad}lcl}
(\lambda x.M)\ N & \to & M[x := N] & \mathsf{cond}\ 0\ M\ N & \to & M \\
\mathsf{succ}\ n & \to & n+1 & \mathsf{cond}\ (n+1)\ M\ N & \to & N \\
\mathsf{pred}\ 0 & \to & 0 & \mathsf{pcond}\ 0\ M\ N & \to & M \\
\mathsf{pred}\ (n+1) & \to & n & \mathsf{pcond}\ (n+1)\ M\ N & \to & N \\
(Y\ M) & \to & M\ (Y\ M) & \mathsf{pcond}\ M\ n\ n & \to & n
\end{array}
$$

$$
\frac{M \to M'}{(c\ M) \to (c\ M')} \qquad \frac{M \to M'}{(M\ N) \to (M'\ N)}
$$

$$
\frac{M \to M'}{\mathsf{cond}\ M\ N\ P \to \mathsf{cond}\ M'\ N\ P} \qquad \frac{M \to M'}{\mathsf{pcond}\ M\ N\ P \to \mathsf{pcond}\ M'\ N\ P}
$$

$$
\frac{N \to N'}{\mathsf{pcond}\ M\ N\ P \to \mathsf{pcond}\ M\ N'\ P} \qquad \frac{P \to P'}{\mathsf{pcond}\ M\ N\ P \to \mathsf{pcond}\ M\ N\ P'}
$$

Figure 1: Structured rewrite rules for PCF; c denotes either succ or pred.

$$
\begin{array}{ll}
(hyp) & \varphi, \gamma \vdash \gamma \\
(refl) & \varphi \vdash P \sqsubseteq P \\
(trans) & \dfrac{\varphi \vdash P \sqsubseteq Q \quad \varphi \vdash Q \sqsubseteq P'}{\varphi \vdash P \sqsubseteq P'} \\
(cong) & \dfrac{\varphi \vdash P_i \sqsubseteq Q_i}{\varphi \vdash f P_1 \cdots P_k \sqsubseteq f Q_1 \cdots Q_k} \\
(omega) & \varphi \vdash \Omega \sqsubseteq P
\end{array}
$$

Figure 2: Axioms and rules for approximations.

$$
\begin{array}{ll}
(consis) & \dfrac{\varphi \vdash \Omega\downarrow}{\varphi \vdash \gamma} \\
(conv) & \dfrac{\varphi \vdash P\downarrow \quad \varphi \vdash P \sqsubseteq Q}{\varphi \vdash Q\downarrow} \\
(div-or-conv) & \dfrac{\varphi, P\downarrow \vdash \gamma \quad \varphi, P \sqsubseteq \Omega \vdash \gamma}{\varphi \vdash \gamma} \\
(flat) & \dfrac{\varphi \vdash P\downarrow \quad \varphi \vdash P \sqsubseteq Q}{\varphi \vdash Q \sqsubseteq P}
\end{array}
$$

Figure 3: Rules for convergence and divergence.

Theorem 4 implies that provability in $\mathcal{L}_C$, subterm-provability in $\mathcal{L}_C$, and validity in $\mathcal{C}$ are equivalent.

Observe that the number of sequents using only subterms of a sequent w is exponential in the size of w. It can be shown from this that subterm-provability in $\mathcal{L}_C$ is in exponential time.

Definition 5 *A* complete set of assumptions *for a sequent w is a set Δ of basic formulas such that, for every subterm P of w, either $P \downarrow$ or $P \sqsubseteq \Omega$ occurs in Δ.*

Theorem 6 *A sequent $w = (\varphi \vdash \gamma)$ is subterm-provable in $\mathcal{L}_C$ iff, for every complete set Δ of assumptions for w, the sequent $\Delta, \varphi \vdash \gamma$ is subterm-provable in $\mathcal{L}_C$* without the rule *$(div - or - conv)$.*

Lemma 7 *Subterm-provability in $\mathcal{L}_C$* without the rule *$(div - or - conv)$ is in polynomial time.*

Theorem 8 *Subterm-provability in $\mathcal{L}_C$ is in co-NP.*

Proof: By Theorem 6, Lemma 7. ■

3.2 A complete logic for $=_{PCF}$

The *strict-or-constant* principle states that, for every PCF term f of type $\sigma_1 \to \cdots \to \sigma_k \to \iota$, either

$$f x_1 \cdots x_k =_{PCF} f y_1 \cdots y_k$$

or

$$f x_1 \cdots \Omega^{\sigma_i} \cdots x_k =_{PCF} \Omega^{\iota}$$

for some i; i.e., f is either constant or strict in some argument. In the context of the logic $\mathcal{L}_C$, the *strict-or-constant* principle can be formalized in a simple-minded way as follows: for each variable $f^{\iota^k \to \iota}$, a proof can use as axioms

either all sequents of the form

$$\emptyset \vdash f P_1 \cdots P_k \sqsubseteq f Q_1 \cdots Q_k$$

or, for some i, all sequents of the form

$$\emptyset \vdash f P_1 \cdots P_{i-1} \Omega \cdots P_k \sqsubseteq \Omega$$

Theorem 9 *A sequent is valid for $\sqsubseteq_{PCF}$ iff it is provable in the logic $\mathcal{L}_C$ with the* strict-or-constant *principle.*

4 Equational systems for simple terms

Simple terms contain the constant Ω^{ι} and variables $z^{\iota \to \iota \to \iota}$, x^{ι}. They are generated by the following grammar:

$$\begin{aligned} S_0 &::= \Omega \mid (z S_0 S_0) \\ S_1 &::= x \mid (z S_0 S_1) \mid (z S_1 S_0). \end{aligned}$$

Note that S_0 generates terms with no occurrences of x, and S_1 generates terms with exactly one occurrence of x.

Lemma 10 *The equations*

$$z t (z \Omega \Omega) = z t \Omega \quad (3)$$

$$z (z \Omega \Omega) t = z \Omega t \quad (4)$$

are sound for $=_C$, $=_{PCF}$.

Proof: Straightforward case analysis on divergence of $(z\Omega\Omega)$. By the flatness of the base type (and monotonicity of z), if $(z\Omega\Omega) \neq \Omega$ then z is constant. ■

Consider the subset of simple terms generated by the following grammar:

$$\begin{aligned} C_0 &::= \Omega \mid (z \Omega \Omega) \\ C_1 &::= x \mid (z \Omega C_1) \mid (z C_1 \Omega). \end{aligned}$$

Lemma 11 *Using equations 3, 4, every term generated by S_0 (resp. S_1) can be proved equal to a term generated by C_0 (resp. C_1).*

Proof: Straightforward induction on structure. ■

Lemma 11 motivates the following shorthand notation for terms:

$$L(P) \stackrel{\text{def}}{=} (z \Omega P)$$

$$\begin{aligned} L^0(P) &= x \\ L^{i+1}(P) &= L(L^i(P)) \end{aligned}$$

$$R(P) \stackrel{\text{def}}{=} (z P \Omega)$$

$$\begin{aligned} R^0(P) &= x \\ R^{i+1}(P) &= R(R^i(P)) \end{aligned}$$

4.1 A complete system for $=_C$

Lemma 12 *The equations*

$$\begin{aligned} RRL(t) &= RL(t) \\ LRL(t) &= RL^2(t) \\ RRL^2(t) &= RL^2(t) \\ LRL^2(t) &= RL^2(t) \end{aligned}$$

$$RL^i(t) = RL^2(t),\ i \geq 2$$

$$\begin{aligned} LLR(t) &= LR(t) \\ RLR^2(t) &= LR^2(t) \\ LLR^2(t) &= LR^2(t) \\ RLR^2(t) &= LR^2(t) \end{aligned}$$

$$LR^i(t) = LR^2(t),\ i \geq 2$$

are sound for $=_C$.

Lemma 13 *(i) Using the equations in Lemma 12, every term generated by* C_1 *can be proved equal to one of the following terms:*

$$L^i(x), RL(x), RL^2(x),$$

$$R^i(x), LR(x), LR^2(x).$$

(ii) No non-trivial equation between two of the terms

$$\Omega, (z\Omega\Omega),$$

$$L^i(x), RL(x), RL^2(x),$$

$$R^i(x), LR(x), LR^2(x)$$

is valid for $=_C$.

Theorem 14 *Equations 3, 4, and the equations in Lemma 12 are sound and complete for equality of simple terms in* C.

Proof: By Lemmas 10, 11, 12, 13. ■

4.2 A complete system for $=_{PCF}$

Lemma 15 *The equations*

$$RL(t) = (z\Omega\Omega) \quad (5)$$

$$LR(t) = (z\Omega\Omega) \quad (6)$$

are sound for $=_{PCF}$.

Proof: By the *strict-or-constant* principle. ■

Lemma 16 *(i) Using equations 5, 6, every term generated by* C_1 *can be proved equal to some* $L^i(x)$ *or some* $R^i(x)$.
(ii) No non-trivial equation between two of the terms

$$\Omega, (z\Omega\Omega), L^i(x), R^i(x)$$

is valid for $=_{PCF}$.

Theorem 17 *Equations 3, 4, 5, 6 are sound and complete for PCF observational congruence of simple terms.*

Proof: By Lemmas 10, 11, 15, 16. ■

Theorem 18 *Equality in* C *and PCF observational congruence are decidable in linear time for simple terms.*

Proof: By Theorems 14, 17. ■

Acknowledgments

We thank Albert Meyer, Jon Riecke and Ramesh Subrahmanyam for numerous discussions.

References

[1] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North-Holland, 1981. Revised Edition, 1984.

[2] G. Berry, and P.-L. Curien. Sequential algorithms on concrete data structures. Theoretical Computer Science 20, 1982.

[3] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*. Cambridge Univ. Press, 1985.

[4] R. Cartwright, and M. Felleisen. Observable sequentiality and full abstraction. Principles of Programming Languages, 1992.

[5] Harvey Friedman. Equality between functionals. In Rohit Parikh, editor, *Logic Colloquium '73*, volume 453 of *Lect. Notes in Math.*, pages 22–37. Springer-Verlag, 1975.

[6] Michael J.C. Gordon, Robin Milner, and C.P. Wadsworth. Edinburgh LCF: A Mechanical Logic of Computation, volume 78 of *Lect. Notes in Computer Sci.* Springer-Verlag, 1979.

[7] Robin Milner. Fully abstract models of the typed lambda calculus. *Theoretical Computer Sci.*, 4:1–22, 1977.

[8] Gordon D. Plotkin. λ-definability and logical relations. Technical Report SAI-RM-4, University of Edinburgh, School of Artificial Intelligence, 1973.

[9] Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Sci.*, 5:223–257, 1977.

[10] Gordon D. Plotkin. Notes on completeness of the full continuous type hierarchy. Unpublished manuscript, MIT, November 1982.

[11] V.Yu. Sazonov. Expressibility of functions in D. Scott's LCF language. *Algebra i Logika*, 15:308–330, 1976. (Russian).

[12] Richard Statman. Completeness, invariance and lambda-definability. *J. Symbolic Logic*, 47:17–26, 1982.

[13] Richard Statman. Equality between functionals revisited. In L.A. Harrington, *et al.*, editor, *Harvey Friedman's Research on the Foundations of Mathematics*, volume 117 of *Studies in Logic*, pages 331–338. North-Holland, 1985.

[14] Ramesh Subrahmanyam, Dan Dougherty. Completeness theorem for the set-theoretic coproduct model. Manuscript, 1993.