科技资料

1991 IEEE 6th Annual Symposium on Logic in Computer Science

Proceedings

Sixth Annual IEEE Symposium on
LOGIC IN COMPUTER SCIENCE

July 15-18, 1991 Amsterdam, The Netherlands

Sponsored by
IEEE Technical Committee on
Mathematical Foundations of Computing
CWI, Amsterdam
Vrije Universiteit, Amsterdam

In cooperation with
Association for Computing Machinery
Association for Symbolic Logic
European Association for Theoretical Computer Science

1951-1991

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society Press, or the Institute of Electrical and Electronics Engineers, Inc.

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY PRESS

Published by the
IEEE Computer Society Press
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Computer Society Press Order Number 2230
Library of Congress Number 89-641304
IEEE Catalog Number 91CH3025-4
ISBN 0-8186-2230-X (paper)
ISBN 0-8186-2231-8 (microfiche)
ISBN 0-8186-2232-6 (case)

Additional copies can be ordered from

IEEE Computer Society Press
Customer Service Center
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264

IEEE Service Center
445 Hoes Lane
PO Box 1331
Piscataway, NJ 08855-1331

IEEE Computer Society
13, avenue de l'Aquilon
B-1200 Brussels
BELGIUM

IEEE Computer Society
Ooshima Building
2-19-1 Minami-Aoyama
Minato-ku, Tokyo 107
JAPAN

Editorial production: Wally Hutchins
Printed in the United States of America by Braun-Brumfield, Inc.
Cover graphic by Alvy Ray Smith

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

These Proceedings are dedicated to the memory of

David M.R. Park, 1935-1990

a deep and original thinker in programming theory,
and a faithful friend.

Foreword

This volume is the Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science (LICS). The symposium encourages international participation of Computer Scientists influenced by mathematical logic and of Logicians influenced by computer science. Previous LICS symposia were held in Cambridge, Massachusetts; Ithaca, New York; Edinburgh, Scotland; Asilomar, California; and Philadelphia, Pennsylvania—each time attracting several hundred enthusiastic participants. The Seventh LICS is scheduled for June 22-25, 1992, on the campus of the University of California, Santa Cruz.

LICS'91 is cosponsored by the IEEE-TC on Mathematical Foundations of Computing, CWI, Amsterdam, and the Vrije Universiteit, Amsterdam, in cooperation with the Association for Computing Machinery–SIGACT, the Association for Symbolic Logic, and the European Association for Theoretical Computer Science.

LICS'91 has been subsidized by

Institutional Sponsors

Institut National de Recherche
en Informatique et en Automatique (INRIA)
Dutch National Facility for Informatics (NFI)
Dutch Royal Academy of Sciences (KNAW)
The Free University of Amsterdam (VU)
The University of Amsterdam
The City of Amsterdam
European Research Consortium
for Informatics and Mathematics (ERCIM)
KLM Royal Dutch Airlines

Donations by these Sponsors make it possible for the LICS Organizers to subsidize student attendance, student author awards, invited speakers, and attendance by researchers without other travel grants.

On behalf of the Organizing Committee and all the LICS'91 participants, I sincerely thank these sponsors for their donations. I also thank the Program Chair, Gilles Kahn, the Conference Co-Chairs, Jan Willem Klop and Roel de Vrijer, and the Publicity Chair, Daniel Leivant, for their many months of effort. We look forward to another fruitful symposium.

Albert R. Meyer
LICS General Chair

Cambridge, Massachusetts
April 1991

Preface

The LICS Symposium aims for wide coverage of theoretical and practical issues in computer science that relate to logic in a broad sense, including algebraic, categorical and topological approaches.

Representative topics mentioned in this year's call for papers include: *abstract data types, automated deduction, concurrency, constructive mathematics, data base theory, finite model theory, knowledge representation, lambda and combinatory calculi, logical aspects of computational complexity, logics in artificial intelligence, logic programming, modal and temporal logics, program logic and semantics, rewrite rules, software specification, type systems, verification.*

The 40 contributed papers in this volume were selected by the Program Committee from a total of 167 submissions; several additional submissions arrived too late to be considered. Selection criteria included originality, quality, relevance to Computer Science, and suitability for conference presentation. A brief synopsis of each extended abstract was mailed to every member of the program committee, with five members designated as primary readers. Some members of the committee chose to consult additional reviewers whose names are listed on the following page. Constructive reviews were sent to all submitting authors whenever available.

Although LICS submissions were read carefully, conference selection is not a formal refereeing process. Many of the papers describe ongoing research and it is anticipated that authors will publish more polished and complete versions in scientific journals.

On behalf of the Program Committee, I thank all authors who chose to submit their papers to LICS'91. Many excellent submissions could not be accepted because of size limitations on the symposium. I would also like to thank the members of the program committee and the additional reviewers for their untiring efforts in reading and evaluating the large number of excellent submissions received this year. I would like to thank Prof. Christine Paulin, who helped me in deciding who should be the primary reviewers. Further, I wish to thank Lydia Vergamini who managed the surprisingly large flow of information generated in evaluating so many papers.

Gilles Kahn
1991 Program Chair

Program Committee

Serge Abiteboul
Samson Abramsky
Krzysztof Apt
Jan Bergstra
Val Breazu-Tannen
Sam Buss
Robert Constable
Pierre-Louis Curien
Nachum Dershowitz
Peter Dybjer
Ursula Goltz
Gilles Kahn (Chair)
Giuseppe Longo
Grigori Mints
Andy Pitts
Simona Ronchi
Masahiko Sato
Ehud Shapiro
Bernhard Steffen

Additional Reviewers

Martin Abadi
Michele Abrusci
Luca Aceto
Dieter Ackermann
Hassan Ait-Kaci
Roberto Amadio
Pierre America
Jean-Marc Andréoli
Toshiyasu Arai
Dennis Arnon
Andrea Asperti
Lennart Augustsson
Leo Bachmair
Jos Baeten
von Bahel
Franco Barbanera
Erik Barendsen
Stefano Berardi
Gérard Berry
B. Bertolino
Alberto Bertoni
Eike Best
Marc Bezem
Nicole Bidoit
Bard Bloom
Roland N. Bol
Gérard Boudol
Steve Brookes
P.J. de Bruin
Luca Cardelli
Felice Cardone
Ilaria Castellani
Didier Caucal
Serenella Cerrito
Rance Cleaveland
Eric de la Clergerie
Philippe Codognet
George Collins
Loic Colson
Mario Coppo
Thierry Coquand
Corradini
Roberto di Cosmo
Gerardo Costa
Bruno Courcelle
Guy Cousineau
Roy Crole
Pierpaolo Degano
Rocco de Nicola
Joëlle Despeyroux
Mariangiola Dezani
Pietro Digianantonio
Gilles Dowek
Thomas Ehrard
Allen Emerson
Uffe Engberg
Lars-Henrik Eriksson
François Fages
Luis Farinas Del Cerro
Amy Felty
Gian-Luigi Ferrari
Nissim Francez
Torkel Franzén
Daniel Fredholm
Laurent Fribourg
Ulf Friedrichsdorf
Ken-Etsu Fujita
Jean Gallier
Peter Gärdenfors
Rob Gerth
Giorgio Ghelli
Paola Giannini
P. Giovanetti
Jean-Yves Girard
Andreas Goerdt
Wolfgang Goerick
Robert Gold
Rajeev Gore
Shigeki Goto
Norbert Gotz
Ed Griffor
Jan Friso Groote
Carl Gunter
Elsa Gunter
Tatsuya Hagino
Masami Hagiya
Thomas Hallgren
Bob Harper
Takashi Hattori
Ralf Heckmann
Andreas Heise
Leen Helmink
Matthew Hennessy
Tom Henzinger
Roger Hindley
André Hirschowitz
Jieh Hsiang
Gérard Huet
Richard Hull
Hardi Hungar
Heinrich Hussmann
Martin Hyland
Furio Honsell
Klaus Indermark
Bart Jacobs
Barry Jay
Thomas Jensen
Herman Ruge Jervell
Jean-Pierre Jouannaud
L.V. Kale
Yukioshi Kameyama
Samuel Kamin
Deepak Kapur
Burghard von Karger
Kent Karlsson
Kasangian
Astrid Kiehn
Michael Kifer
Jan Willem Klop
Steven Klusener
Jürgen Knoop
Satoshi Kobayashi
Henri Korver
C.P.J. Koymans
H. Kuchen
Yves Lafont
Kim Larsen
C. Laneve
Pierre Lescanne
James Lipton
Gabrile Lolli
Rita Loogen
Zhao Hui Luo
Nancy Lynch
Pasquale Malacaria
Leo Marcus
Alberto Martelli
Maurizio Martelli

Simone Martini
Andrea Masini
Michel Mauny
G. Mauri
Georges McNulty
José Meseguer
J-J.Ch. Meyer
Dale Miller
John Mitchell
Subrata Mitra
Eugenio Moggi
Ugo Montanari
Shinichi Morishita
Yael Moskowitz
Horst Müller
Satoru Miyano
Alan Mycroft
Hayao Nakahara
Hiroshi Nakagawa
Tobias Nipkow
Bengt Nordström
Hiroshi Nunokawa
Mitsuhiro Okada
Ernst-Rüdiger Olderog
Barbara Paech
Valeria de Paiva
Erik Palmgren
Remo Pareschi
Peter Padawitz
Michel Parigot
Christine Paulin
Lawrence Paulson
Wojciech Penczek
Leonard Pitt
David Plaisted
Amir Pnueli
Axel Poigné
A. Policriti
Randy Pollack
Alban Ponse
I.V. Ramakrishnan
Uday Reddy
Wolfgang Reisig
Jeff Remmel
Peter Revesz
Jon Riecke
Edmund Robinson
François Rouaix
Gudula Rünger
Michael Rusinowitch
Jan Rutten
Yehoshua Sagiv
Takafumi Sakurai
Hiroyuki Sato
Taisuke Sato
André Scedrov
Richard Scherl
Arno Schmitt
Helmut Seidl
Svetozar Serafimovski
Juichi Shinoda
Kurt Sieber
Michael Siegel
F. Simon
Robert de Simone
Jan Smith
Scott Smolka
Ugo Solitro
Eugene Stark
Jacques Stern
Thomas Streicher
Alvaro Tasistro
Makoto Tatsuta
Satish Thatte
Wolfgang Thomas
Andrew Thomason
Atsushi Togashi
Christophe Tollu
Yoshihiko Toyama
Hideki Tsuiki
Daniele Turi
Silvio Valentini
J. van de Wiele
Peter Van Emde Boas
Moshe Vardi
Betty Venneri
Victor Vianu
Albert Visser
Walter Vogler
Fes-Jan de Vries
Scott Weinstein
Carsten Weise
Glynn Winskel
Marianne Winslett
Akihiro Yamamoto
Naoki Yonezaki

Conference Organization

General Chair

Prof. Albert R. Meyer
MIT Lab. for Computer Science

1991 Conference Co-chair

Prof. Roel de Vrijer
Free University of Amsterdam

1991 Conference Co-chair

Dr. Jan Willem Klop
CWI, Amsterdam

1991 Program Chair

Gilles Kahn
INRIA, Sophia-Antipolis

Publicity Chair

Prof. Daniel Leivant
Carnegie-Mellon University

Organizing Committee

Martin Abadi
Jon Barwise
Ashok Chandra
Robert Constable (General Chair Elect)
Erwin Engeler
Jean Gallier
Joseph Goguen
David Gries
Yuri Gurevich
Susumu Hayashi
David S. Johnson
Gilles Kahn
Jan Willem Klop
Dexter Kozen
Daniel Leivant
Zohar Manna
Albert R. Meyer (General Chair)
Grigori Mints
John C. Mitchell
Yiannis Moschovakis
Christos Papadimitriou
Rohit Parikh
Gordon D. Plotkin
Grzegorz Rozenberg
Dana Scott
Jerzy Tiuryn
Roel de Vrijer

Table of Contents

Session 10
Chair: E. Shapiro

Session 11
Chair: K. Apt

Session 12
Chair: M. Sato

Session 13
Chair: U. Goltz

Session 1

Chair:
G. Longo

A Foundational Delineation of Computational Feasibility

Daniel Leivant*

Abstract

A function over $\{0,1\}^*$ is in P-Time iff it is computed by a program which can be proved correct in second-order logic with set-existence (comprehension) restricted to positive quantifier-free formulas. This set-existence principle captures formally the view of infinite totalities as evolving, not completed, entities.

1 Introduction

1.1 Feasibility and P-Time

Feasible computing has been identified for long with computability within deterministic polynomial time, primarily on practical and circumstantial grounds: P-Time functions are easily defined and computed, and are closed under many natural operations; and most known worst-case lower-bounds are either bounded by polynomials of small degrees, which are clearly feasible, or are at least exponential, and clearly non-feasible. The central importance of P-Time has been contested as of late, notably because feasible probabilistic classes might subsume P-Time in their practical significance, and because bounds such as $n^{\log\log n}$ are more feasible in practice than say n^{1000}. At the same time, the fundamental nature of P-Time has been reaffirmed repeatedly by various characterizations and stability results. For example, relations computable in P-Time over enumerated finite structures are the same as the ones computable by recursion equations [Saz80,Gur83] or by pure uninterpreted logic programs [Pap85], or by alternating multihead automata [CKS81,Gur87]; they are also the same as the relations defined by positive first-order fixpoints [Var82,Imm86], or by first-order inflationary fixpoints [GS86,Lei90a], or by alternating transitive-closure [Imm87]. The P-Time functions over $\mathbb{N}$ have, among others, characterizations in terms of a subrecursive schema [Cob65], provability in a weak system for arithmetic [Bus86], and typability in a bounded version of linear logic [GSS89].

These characterizations testify to the significance of P-Time, but they all seem to lack a principle directly pertinent to feasibility, one that would justify the identification of P-Time with feasible computing. Our aim here is to propose such a principle.

1.2 The ontology of numeric terms

Computational feasibility is closely related to the ontology of numeric terms. As soon as non-feasible functions are named, they take a life of their own, and ontologically problematic natural numbers become easily nameable, such as $3 \Uparrow 5 =_{df} 3^{3^{3^{3^{3}}}}$. In particular, once exponentiation is admitted, then very short terms exist whose numeric values exceeds not only human imagination, but also possible realization in the physical world: $3 \Uparrow 5$ could not be spelled out as a decimal numeral even by quark-size computers filling up the observable universe and working concurrently since the big bang at a speed that exceeds the limitations of quantum mechanics.

The abyss between the value and the notation-size of such terms has been addressed by a number of mathematicians and philosophers, including Bernays [Ber35], van Danzig [Dan56], Yessenin-Volpin [Yes70], Isles [Isl91], and Nelson [Nel86]. Gandy [Gan89] concludes that "very large numbers are abstract not concrete (not potentially concrete) objects: they are more akin to infinite sets than to concretely presented numbers."

1.3 Predicativity and potential infinities

Basic infinite sets, such as the set $\mathbb{N}$ of the natural numbers or the first inaccessible cardinal, are conceptualized as being generated by a process. To be admitted as legitimate, we must assume that some "universe" exists within which that process can be applied "indefinitely". Similarly, our belief that $3 \Uparrow 5$ denotes a natural numbers is based on the conviction that the calculation of that term will be completed eventually. Of course, we can support that conviction with a proof

*Author's current address: SCS, Carnegie Mellon University, Pittsburgh, PA 15213. Address effective Fall 1991: Computer Science Department, Indiana University, Bloomington, IN 47405.

CH3025-4/91/0000/0002$01.00 © 1991 IEEE

by induction, but, as we shall see, for that proof to make sense we must admit that *infinite sets exist as complete totalities.*

Less than a century ago, the legitimacy of infinite sets as completed totalities was not as universally taken for granted as it is today, under the influence of Cantorian set theory. Hilbert had hoped to shelter Mathematics from the potential dangers of actual infinities by reducing it to its finitistic fragment. An important aspect of Brouwer's intuitionistic foundations is the insistence that infinite totalities are only unbounded constructions: "The natural numbers, though treated, constitute only a potential totality in constructive mathematics" [Kre61].

Recall that a definition of a set X is *impredicative* if it refers to a collection of which X is an element. Uncontrolled impredicativity leads to contradictions, as in Russell's Paradox. However, impredicative definitions abound in Mathematical Analysis, where real numbers (i.e. subsets of $\mathbb{N}$ or functions over $\mathbb{N}$, depending on the representation) are defined in terms of quantification over all reals. We normally expect no contradiction to arise, because we implicitly assume that the power set of $\mathbb{N}$, $\mathcal{P}\mathbb{N}$, is given as a completed totality prior to the definition of any particular members thereof.[1] In predicative systems of analysis[2] one refrains from assuming the power set of $\mathbb{N}$ as given, albeit $\mathbb{N}$ is assumed as a completed totality. This implies that a subset of $\mathbb{N}$ cannot be defined in terms of quantification over $\mathcal{P}\mathbb{N}$, and circular definitions are thereby excluded.

An argument raised by Nelson [1986] is that the definitions of $\mathbb{N}$ are also circular: the generative (inductive) definition, as a set constructed by repeated application of the successor function, presupposes an understanding of $\mathbb{N}$ itself (specifically when Induction is proposed as the formal justification of the process). The definition of $\mathbb{N}$ as the intersection of all sets containing 0 and closed under successor presupposes that such a set exists, and moreover uses a blatantly impredicative quantification over sets.[3]

[1] Impredicative definitions of this form are captured by the Subset (Separation) axiom schema of Zermelo's Set Theory.

[2] Predicative Analysis goes back to Borel and the semi-intuitionists of the turn of the century, and has been revived by Kreisel, Feferman, Wang, Schüttee, and many others.

[3] Shoenfield and Wang (in conversation with Kreisel, reported in [Kre61, fn. 1]) have made the interesting dual observation that if the generative justification of $\mathbb{N}$ were to be taken as "predicative", then one should also accept as predicative the set $\mathbf{W}$ of all well-founded countably-branching trees, which is complete-Π^1_1 and not "predicative" in the sense of being hyperarithmetical.

Nelson point is, then, that the culprit in generating ontologically dubious terms is the impredicative justification of the set $\mathbb{N}$, and therefore the impredicativity of proof by Induction. Nelson observes that induction presupposes that $\mathbb{N}$ is given as a completed totality, and so using induction to justify that the values of certain terms are in $\mathbb{N}$ is an impredicative argument. He goes on to develop a system of Predicative Arithmetic, in which exponentiation is not provably correct. A problem with Nelson's development is that no clear cut rationale is given for admitting addition and multiplication, but not exponentiation, as primitives. Isles [1991] brings out the impredicative nature of the proof that the exponentiation function is well defined, but he too does not provide a foundational delination of feasibility.

1.4 Strictly Predicative Comprehension

Levels of impredicativity can be precisely calibrated by comprehension (set existence) principles, i.e. the admittance as legitimate of sets $\{x \in \mathbb{N} \mid \varphi\}$ for certain formulas φ. Much progress has been made in the last decade in calibrating the strength of formalisms for second-order arithmetic with weak forms of comprehension (notably by H. Friedman, Mints, Sieg, Simpson, and Smith). However, all formalisms considered are built on top of Primitive Recursive Arithmetic, so these studies are of no help in delineating the impredicativity involved in the primitive recursive (PR) functions, let alone in smaller classes.

A framework for calibrating the impredicativity of sub-PR functions was proposed in [Lei83, Lei90], with second-order *logic* used in place of second-order arithmetic. Contrary to weak systems for second-order arithmetic, the set of natural numbers is here *not* assumed as a completed totality. The method does not depend on any choice of basic numeric functions (such as addition and multiplication) or of axioms for them, and is therefore suitable for calibrating the logical nature of "small" functions. Moreover, it applies as easily to any term algebra as to $\mathbb{N}$.

Consider now the question of what instances of comprehension can be justified on strictly predicative grounds. Since the existence of infinite sets as completed totality cannot be so justified, we must stipulate that relational variables range over finite or potentially-infinite sets, i.e. sets that are "coming into being". Over a given structure we use comprehension to delineate new sets that are finite or potentially infinite, from the structure functions and relations, and from relational variables which denote already-defined finite or potentially infinite sets. Specifically, if R is a relational variable, and $\vec{t}$ are terms (where $arity(\vec{t}) = arity(R)$), we admit $\{x \mid R(\vec{t})\}$. We must also admit finite unions and intersections of admitted sets. However, we can *not* admit the complement of an admitted set S, since this is tantamount to accepting S as an actual infinity, for which non-members can be identified. Also, the use of quantifiers is suspect, because they refer to exhaustive inspection of the structure universe. [4] We are thus led to accept, on strictly predicative grounds, comprehension over exactly the positive quantifier-free formulas (i.e. without negation or implication).

The main result of this paper states that the computable functions justified on the basis of positive quantifier-free comprehension are precisely the functions computable in deterministic polynomial time. This shows that the class P-Time arises naturally from a foundational analysis of feasibility, and that terms using exponentiation can be justified as meaningful only under the admission of infinite sets as completed totalities. Specific terms, such as $3 \uparrow\uparrow 5$, have their own complete computation as direct justification, but since no such computation can ever be exhibited, such terms can be feasibly justified only via the general justification of exponentiation, i.e. via implicit reference to completed infinite sets.

2 Functional programs

2.1 Herbrand-Gödel programs

Our canonical computation model is functional programs, in the Herbrand-Gödel style (See [Kle52] or [Lei90] App. 1 for expositions). The original Herbrand-Gödel definition is for $\mathbb{N}$, the free term algebra generated from a constant 0 and a unary function s. We use such programs over arbitrary free algebras, in particular the term algebra generated from a constant ϵ and unary functions 0 and 1, i.e. simply the set $W = \{0,1\}^*$ (e.g. the word 011 is identified with the term $011\epsilon = 0(1(1(\epsilon)))$). We can assume, without loss of generality, that functional programs are coherent, i.e. that they define a partial function, and not a multiple-valued function.[5]

For example, the following program (over W) computes the function $\odot$, which on input v, w returns $w^n = w \cdots w$ (n factors in concatenation) where $n = length(v)$. We use c to range over $\{0,1\}$.

$$\epsilon \oplus w = w \qquad (cv) \oplus w = c(v \oplus w)$$
$$w \odot \epsilon = \epsilon \qquad w \odot (cv) = w \oplus (w \odot v)$$

2.2 Convergence

To formally state the convergence of a functional program for some or for all input one needs to refer to potentially non-terminating computations. An approach common in Proof Theory, and due to Kleene [Kle52, Kle69], is to explicitly describe operational convergence, in a formalism sufficiently rich to code (Gödelize) the operational machinery. In logics of programs one expresses convergence using modal operators (as in Dynamic Logic, see e.g. [Pra80]) or using potentially non-denoting terms (see e.g. [Gol82]).

We continue here the alternative approach of [Lei83, Lei90], where programs are considered not as definitions of partial functions over the term-algebra A in hand, but as definitions of total functions over any structure whose vocabulary contains the generators of A. The key connection between such structures and convergence of programs over the intended term-algebra is given by the following observation [Lei83, Lei90]. Fix a term algebra A. For a functional program P (over A) let $[P]$ be the conjunction of the universal closures of the equations in P.

Theorem 2.1 *Let P be a functional program with principal function identifier* $\mathbf{f}$. *The following conditions are equivalent: (1) P converges (over A) for input $\vec{\mathbf{t}} \in A$; (2) for every model $\mathcal{S}$ of $[P]$, there is some $\mathbf{r} \in A$ such that $\mathcal{S} \models \mathbf{f}(\vec{\mathbf{t}}) = \mathbf{r}$; (3) there is some $\mathbf{r} \in A$ such that for every model $\mathcal{S}$ of $[P]$ $\mathcal{S} \models \mathbf{f}(\vec{\mathbf{t}}) = \mathbf{r}$.*

The entailment relation $\models$ refers here to all structures of the appropriate vocabulary.

2.3 Second-order statement of convergence

We consider a second-order extension of first-order logic with new variables ranging over relations, and quantification over such variables. Let A be a free term algebra. Writing A also for the predicate "is $\in A$", we have

$$A(x) \quad \equiv_{\rm df} \forall Q \;\; Cl_A[Q] \;\to\; Q(x)$$

[4] We comment on this in the list of research directions below.

[5] Kleene [1952] showed this for numeric functions. A proof for the general case can be obtained either by generalizing Kleene's proof for a computation model with fixpoint, or by generalizing the simulation used in Lemma 3.2 below for Turing machine computbility. Details will be given elsewhere.

where $Cl_A[Q]$ is a formula stating that Q is closed under the generators of A. For instance,

$$Cl_W[Q] \equiv_{df} Q(\epsilon) \wedge \forall u\,(Q(u) \rightarrow (Q(0u) \wedge Q(1u)))$$

From Theorem 2.1 we then conclude:

Theorem 2.2 *Let P be a functional program with principal function identifier* **f**. *P converges (over A) for all input iff*

$$[P] \models A(\vec{x}) \rightarrow A(\mathbf{f}(\vec{x}))$$

Here $arity(\vec{x}) = arity(\mathbf{f})$, $A(x_1 \ldots x_k)$ abbreviates $A(x_1) \wedge \cdots \wedge A(x_k)$, and the relational quantifiers have their standard interpretation.

2.4 Provable convergence

By Theorem 2.2 there is a natural, axiom-independent, way of formulating in formalisms for second-order logic the provable convergence of functions.

Let L be a formalism for second (or higher) order logic. We say that a function f over A is **provable in** L iff it is computed by some functional program P (with principal function identifier **f**) such that

$$[P] \vdash_L A(\vec{x}) \rightarrow A(\mathbf{f}(\vec{x}))$$

Given a collection C of formulas, let $L_2(C)$ be a formalism for second-order logic with comprehension for formulas in C (for example, the natural deduction formalism of [Pra65]). The interpretation in [Pra65] of second-order arithmetic in second-order logic implies that the provable functions (over $\mathbb{N}$) of L_2(all second-order formulas) are precisely the provably-recursive functions of second-order arithmetic.[6] In particular, from $N(x)$ one gets induction with respect to x for all formulas.

To obtain from $N(x)$ induction for a first-order arithmetic formula φ we need comprehension for the interpretation φ' of φ, which in general is *not* first-order, because quantifiers in φ are interpreted in φ' as quantifiers relativized to N. In [Lei90b, Lei91] it is shown that the provably recursive functions of first-order arithmetic are precisely the provably recursive functions of L_2(strict-Π^1_1), and that the primitive-recursive functions are precisely the provably recursive functions of L_2(strict-Π^1_1without relational parameters).[7]

[6] A simple method for dealing with Peano's third and fourth axioms is given in [Lei90].

[7] A formula is strict-Π^1_1 if it is of the form $\forall\vec{Q}\exists\vec{x}\psi$, with ψ quantifier-free. In [Lei91] we gave an overview of the concept's significance.

2.5 S-provable convergence

We shall refer here to a notion of provable convergence formally weaker than the one defined above. Let $\mathcal{S}$ be a structure in the vocabulary $V_A = \{\mathbf{f}_0 \ldots \mathbf{f}_k\}$ of A, where $arity(\mathbf{f}_i) = r_i \geqslant 0$. We say that $\mathcal{S}$ is *surjective* if its universe $|\mathcal{S}|$ is covered by the range of the structure functions and constants, i.e. if

$$\mathcal{S} \models Surj_A$$

where

$$Surj_A \equiv_{df} \forall u \bigvee_{i=0\ldots k} \exists v_1 \ldots v_{r_i}\; u = \mathbf{f}_i(v_1 \ldots v_{r_i}).$$

For example

$$Surj_W \equiv \forall u\,(u=\epsilon \vee \exists v\,(u=0v) \vee \exists v\,(u=1v))$$

The surjective structures include not only the free algebra A itself, but also most natural examples of non-standard models for the theory of A. For example, the flat A domain is surjective because $\perp = \mathbf{f}(\perp, \ldots)$ for any non 0-ary $\mathbf{f} \in V_A$ (we assume that A is non-trivial).

Since every term algebra A is surjective, Theorem 2.2 holds trivially when validity is restricted to validity in surjective structures; i.e. P converges over A for all input iff

$$[P],\, Surj_A \models A(\vec{x}) \rightarrow A(\mathbf{f}(\vec{x})).$$

Given a formalism L as above, we say that a function f over A is **s-provable** in L iff it is computed by some functional program P (with principal function identifier **f**) such that

$$[P],\, Surj_A \vdash_L A(\vec{x}) \rightarrow A(\mathbf{f}(\vec{x})).$$

Every function provable in L is trivially s-provable in L. The next theorem states that the converse holds when L has enough comprehension. Let $\alpha \equiv \alpha[x]$ be a formula with some single free variable x. If φ is a second-order formula, its *relativization to* α, φ^α, is obtained by restricting first-order quantification to elements satisfying α, and restricting second-order quantification to subsets of the the extension of α. I.e., φ^α is defined by recurrence on φ as follows, where, for k-ary Q, $Q \subseteq \alpha$ abbreviates $\forall v_1 \ldots v_k\, Q(\vec{v}) \rightarrow \alpha[v_1] \wedge \cdots \wedge \alpha[v_k]$.

$$\varphi^\alpha \equiv_{df} \varphi \quad (\varphi \text{ quantifier free})$$
$$(\neg\varphi)^\alpha \equiv_{df} \neg(\varphi^\alpha)$$
$$(\varphi \star \psi)^\alpha \equiv_{df} \varphi^\alpha \star \psi^\alpha \quad (\star \text{ a binary connective})$$