ELEMENTARY RECURSIVE ALGORITHMS [*]

YIANNIS N. MOSCHOVAKIS

Contents

1. What is an algorithm? 1
2. Equational logic of partial terms with conditionals 5
3. Continuous, pure recursors 7
4. Canonical forms and intensions 11
5. Decidability of intensional program equivalence 20

They run our lives, if you believe the hype in the news, but there is no precise definition of *algorithms* which is generally accepted by the mathematicians, logicians and computer scientists who create and study them.[1] My main aims here are (first) to discuss briefly and point to the few publications that try to deal with this foundational question and (primarily) to outline in Sections 4 and 5 simple proofs of two basic mathematical results about the *elementary recursive algorithms from specified primitives* expressed by *recursive* (McCarthy) *programs.*

§1. What is an algorithm? With the (standard, self-evident) notation of Sections 1D and 1E of ARIC, we will focus on algorithms which compute partial functions and (decide partial) relations

$$f : A^n \rightharpoonup A_s \qquad (s \in \{\texttt{ind}, \texttt{boole}\}, A_{\texttt{ind}} = A, A_{\texttt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}) \tag{1-1}$$

from the finitely many primitives of a (partial, typically infinite) Φ-structure

$$\mathbf{A} = (A, \{\phi^{\mathbf{A}}\}_{\phi\in\Phi}) \quad (\phi \in \Phi, \phi^{\mathbf{A}} : A^{\mathrm{arity}(\phi)} \rightharpoonup A_{\mathrm{sort}(\phi)}). \tag{1-2}$$

The most substantial part of this restriction is that it leaves out algorithms with side effects and interaction, cf. the footnote on page 3 of ARIC and the relevant Section 3B in Moschovakis [1989a].

Equally important is the restriction to *algorithms from specified primitives*, especially as the usual formulations of the *Church-Turing Thesis* suggest that the primitives of a Turing machine are in some sense "absolutely computable" and

[*]A preliminary version of the results in this article was included in an early draft of Moschovakis [2019] (ARIC) as an additional Chapter in Part 1. It was replaced by a brief summary in Section 2H of the final, published version of ARIC, because it is not directly relevant to the main aim of that monograph, which is to develop methods for deriving and justifying robust lower complexity bounds for mathematical problems. There are many references in the sequel to ARIC and to older work by several people, but I have included enough of the basic definitions and facts so that the statements and proofs of the new results in Sections 4 and 5 stand on their own.

[1]Using imprecise formulations of the *Church-Turing Thesis* and vague references to Church [1935], [1936] and Turing [1936], it is sometimes claimed naively that *algorithms are Turing machines.* This does not accord with the original formulations of the Church-Turing Thesis, cf. the discussion in Section 1.1 of Moschovakis [2014] (which repeats points well-known and understood by those who have thought about this matter); and it is not a useful assumption for algebraic complexity theory, cf. page 2 of ARIC.

need not be explicitly assumed or justified. We have noted in the Introduction to ARIC (and in several other places) why this is not a useful approach; but in trying to understand *computability* and the meaning of the Church-Turing Thesis, it is natural to ask whether there are absolutely computable primitives and what those might be. See Sections 2 and 8 of Moschovakis [2014] for a discussion of the problem and references to relevant work, especially the eloquent analyses in Gandy [1980] and Kripke [2000].

There is also the restriction to *first-order primitives*, partial functions and relations. This is necessary for the development of a conventional theory of complexity, but recursion and computability from higher type primitives have been extensively studied: see Kleene [1959], Kechris and Moschovakis [1977] and Sacks [1990] for the higher-type recursion which extends directly the first-order notion we have adopted in ARIC, and Longley and Normann [2015] for a near-complete exposition of the many and different approaches to the topic.[2]

Once we focus on algorithms which compute partial functions and relations as in (1-1) from the primitives of a Φ-structure, the problem of modeling them by set-theoretic objects comes down basically to choosing between *iterative algorithms* specified by (classical) computation models as in Section 2C of ARIC and *elementary recursive algorithms* expressed directly by recursive (McCarthy) programs; at least this is my view, which I have explained and defended as best I can in Section 3 of Moschovakis [1998].

The first of these choices—that algorithms are iterative processes—is *the standard view*, explicitly or implicitly adopted (sometimes with additional restrictions) by most mathematicians and computer scientists, including Knuth in Section 1.1 of his classic Knuth [1973]. More recently (and substantially more abstractly, on arbitrary structures), this standard view has been developed, advocated and defended by Gurevich and his collaborators, cf. Gurevich [1995], [2000] and Dershowitz and Gurevich [2008]; see also Tucker and Zucker [2000] and Duží [2014].

I have made the second choice—that algorithms are directly expressed by systems of mutual recursive definitions—and I have developed and defended this view in several papers, including Moschovakis [1998]. I will not repeat these arguments here, except for the few remarks in the remainder of this Section about the role that iterative algorithms play in the theory of recursion and (especially) Proposition 4.6, which verifies that iterative algorithms are "faithfully coded" by the recursive algorithms they define, and so their theory is not "lost" when we take recursive algorithms to be the basic objects.

By the definitions in Section 2A of ARIC (reviewed in Section 2 below), a **recursive** (McCarthy) **Φ-program** is a syntactic expression

$$E \equiv E_0 \text{ where } \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1, \dots, \mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\}, \tag{1-3}$$

where $E_0, E_1, \dots, E_k$ are (pure, explicit) $\Phi \cup \{\mathsf{p}_1, \dots, \mathsf{p}_K\}$-terms and for every $i = 1, \dots, K$ all the individual variables which occur in E_i are included in the

[2]See also Moschovakis [1989a]—which is about recursion on structures with arbitrary monotone functionals for primitives—and the subsequent Moschovakis [1989b] where the relevant notion of *algorithm from higher-type primitives* is modeled rigorously.

list $\vec{\mathsf{x}}_i$ of distinct individual variables; and an **extended program** is a pair

$$(E, \vec{\mathsf{x}}) \equiv E(\vec{\mathsf{x}}) \equiv E_0(\vec{\mathsf{x}}) \text{ where } \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1, \dots, \mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\} \tag{1-4}$$

of a program E and a list of distinct individual variables $\vec{\mathsf{x}}$ which includes all the individual variables which occur in E_0.

Recursive algorithms. My understanding of *the algorithm defined by* $E(\vec{\mathsf{x}})$ *in a* Φ*-structure* $\mathbf{A}$ is that it calls for solving in $\mathbf{A}$ the system of recursive equations in the *body* of $E(\vec{\mathsf{x}})$ (within the braces $\{\cdots\}$) and then plugging the solutions in its *head* $E_0(\vec{\mathsf{x}})$ to compute $\mathrm{den}(\mathbf{A}, E(\vec{x}))$, the value of the partial function computed by $E(\vec{\mathsf{x}})$ on the input $\vec{x}$; *how* we find the canonical (least) solutions of this system is not specified in this view by the algorithm from the primitives of $\mathbf{A}$ defined by $E(\vec{\mathsf{x}})$.

This "lack of specificity" of elementary recursive algorithms is surely a weakness of the view I have adopted, especially as it leads to some difficult problems.

Implementations. To compute $\mathrm{den}(\mathbf{A}, E(\vec{x}))$ for $\vec{x} \in A^n$, we might use the method outlined in the proof of the Fixed Point Lemma 1B.1 or the recursive machine defined in Section 2C of ARIC, or any one of several well known and much studied *implementations of recursion.* These are iterative algorithms defined on structures which are (typically) richer than $\mathbf{A}$ and they must satisfy additional properties relating them to $E(\vec{\mathsf{x}})$—we would not call any iterative algorithm from any primitives which happens to compute the same partial function on A as $E(\vec{\mathsf{x}})$ an implementation of it; so to specify *the implementations of a recursive program* is an important (and difficult) part of this approach to the foundations of the theory of algorithms, cf. Moschovakis [1998] and (especially) Moschovakis and Paschalis [2008] which proposes a precise definition and establishes some basic results about it.

On the other hand, the standard view has some problems of its own:

Recursion vs. iteration. Iterators are defined rigorously in ARIC and Theorem 2C.2 gives a strong, precise version of the slogan

(1-5) *iteration can be reduced to recursion*;

on every structure $\mathbf{A}$*, if* $f : A^n \rightharpoonup A_s$ *is computed by an* $\mathbf{A}$*-explicit iterator, then* f *is also computed by an* $\mathbf{A}$*-recursive program.* The converse of (1-5) is not true however: there are structures where some $\mathbf{A}$-recursive relation is not *tail recursive*, i.e., it cannot be decided by an iterator which is explicit in $\mathbf{A}$—it is necessary to add primitives and/or to enlarge the universe of $\mathbf{A}$. Classical examples include Patterson and Hewitt [1970] (Theorem 2G.1 in ARIC), Lynch and Blum [1979] and (the most interesting) Tiuryn [1989].[3] The last Chapter 9 of ARIC discusses a (basic) problem from *algebraic complexity theory*, a very rich and active research area in which the recursive representation of algorithms is essential.

Functional vs. imperative programming. It is sometimes argued that the main difference between recursion and iteration is a matter of "programming style", at least for the structures which are mostly studied in complexity theory in

[3]See also Jones [1999], [2001] and Bhaskar [2017], [2018].

which every recursive partial function is *tail recursive*, i.e., computed by some **A**-explicit iterator. This is not quite true: consider, for example the classical *merge-sort* algorithm (Section 1C of ARIC) which sorts strings from the primitives of the Lisp structure

$$\mathbf{L}^* = (L^*, \mathrm{nil}, \mathrm{eq}_{\mathrm{nil}}, \mathrm{head}, \mathrm{tail}, \mathrm{cons})$$

over an ordered set L, (1D-5) in ARIC; it is certainly implemented by some $\mathbf{L}^*$-explicit iterator (as is every recursive algorithm of $\mathbf{L}^*$), but which one of these *is* the merge-sort? It seems that we can understand this classic algorithm and reason about it better by looking at Proposition 1C.1 of ARIC rather than focussing on choosing some specific implementation of it.[4]

Proofs of correctness. In Proposition 1C.2 of ARIC, we claimed the correctness of the merge-sort—that it sorts—by just saying that

The sort function satisfies the equation ...

whose proof was too trivial to deserve more than the single sentence

The validity of (1C-6) *is immediate, by induction on* $|u|$.

To prove the correctness of an iterator that "expresses the merge-sort", you must first design a specific one and then explain how you can extract from all the "housekeeping" details necessary for the specification of iterators a proof that what is actually being computed is the sorting function; most likely you will trust that a formal version of (1C-6) of ARIC is implemented correctly by some compiler or interpreter of whatever higher-level language you are using, as we did for the recursive machine.

Simply put, whether correct or not, the view that algorithms are faithfully expressed by systems of recursive equations typically separates proofs of *their correctness* and their basic *complexity properties* which involve only purely mathematical facts from the relevant subject and standard results of fixed-point-theory, from proofs of *implementation correctness* for programming languages which are ultimately necessary but have a very different flavor.

Defining mathematical objects in set theory. Finally, it may be useful to review here briefly what it means to *define algorithms* according to a general (and widely accepted, I think) naive view of what it means to *define mathematical objects.* This is discussed more fully in Section 3 of Moschovakis [1998].

One standard example is the "definition" (or "construction") of *real numbers* using (canonically) convergent sequences of rational numbers: we set first

$$\begin{aligned} x \in \mathrm{Cauchy}(\mathbb{Q}) \iff & \ x : \mathbb{N} \to \mathbb{Q} \\ & \ \& \ (\forall k)(\exists n)(\forall i,j)\Big[i,j \geq n \Longrightarrow |x(i) - x(j)| < \frac{1}{k+1}\Big], \end{aligned}$$

next we put for $x, y \in \mathrm{Cauchy}(\mathbb{Q})$

$$x \sim y \iff (\forall k)(\exists n)(\forall i > n)|x(i) - y(i)| < \frac{1}{k+1},$$

[4]See also Theorem 4G.1 of ARIC for a precise formulation and proof of the basic optimality property of the merge-sort.

and finally we declare that $x, y \in \mathrm{Cauchy}(\mathbb{Q})$ *represent* the same real number if $x \sim y$.

In general, a *representation in set theory* of a mathematical notion P is a pair of a set (or class) $\mathcal{C}_P$ of set-theoretic objects which represent the objects that fall under P and an equivalence condition $\sim_P$ on $\mathcal{C}_P$ which models the identity relation on them; and the representation is *faithful*—and can be viewed as a *definition of P in set theory*—to the extent that the relations on P-objects that we want to study are those which are equivalent to the $\sim_P$-invariant properties of the objects in $\mathcal{C}_P$.

Now, number theorists could not care less about such "definitions" of real numbers and they were happily investigating the existence and properties of rational, algebraic and transcedental numbers for more than two thousand years before they were given in the 1870s. Part of the reason for giving them was to argue for adopting set theory as a "foundation" (a "universal language") for all of mathematics and to apply set theoretic methods to algebraic number theory;[5] but this is not the main point—some people would use category theory today and argue for its superiority over set theory as both a foundation and a tool for applications. The main point of looking for such "definitions" is to identify *the fundamental, characteristic properties of a mathematical notion*, which, to repeat, should be the $\sim_P$-invariant properties of the set-theoretic objects that model the notion.

Another, fundamental (and much more sophisticated) example of this process of constructing faithful modelings of mathematical notions is the identification in Kolmogorov [1933] of real-valued *random variables* with measurable functions $X : \Omega \to \mathbb{R}$ on a probability space, under various equivalence relations.

§2. Equational logic of partial terms with conditionals.

With the definitions in Section 1A of ARIC, a *partial function*

$$f : X \rightharpoonup W$$

with *input set* X and *output set* W is an (ordinary) function $f : D_f \to W$ on some subset $D_f \subseteq X$, the *domain of convergence* of f. We write

$$f(x)\downarrow \iff_{\mathrm{df}} x \in D_f, \quad f(x)\uparrow \iff_{\mathrm{df}} x \notin D_f \qquad (x \in X)$$

and most significantly, for $f, g : X \rightharpoonup W$ and $x \in X$,

$$f(x) = g(x) \iff_{\mathrm{df}} \Big(f(x)\downarrow \ \& \ g(x)\downarrow \ \& \ f(x) = g(x)\Big) \text{ or } \Big(f(x)\uparrow \ \& \ g(x)\uparrow\Big).$$

This relation between partial functions $f, g : X \rightharpoonup W$ with the same input and output sets is sometimes called *Kleene's strong equality* between "partial terms", but there is nothing partial about it: for any two $f, g : X \rightharpoonup W$ and any $x \in X$, the proposition $f(x) = g(x)$ is either true or false.

A *signature* or *vocabulary* is a set Φ of *function symbols*, each with assigned $\mathrm{arity}(\phi) \in \mathbb{N}$ and $\mathrm{sort}(\phi) \in \{\mathtt{ind}, \mathtt{boole}\}$; and as in (1-2), a (two-sorted, partial)

[5] Most spectacular of these was Cantor's proof that *transcedental numbers exist* by a counting argument much simpler than Liouville's original proof—the first "killer application" of set theory.

Φ-*structure* is a pair

$$\mathbf{A} = (A, \{\phi^{\mathbf{A}}\}_{\phi\in\Phi}) \quad (\phi \in \Phi, \phi^{\mathbf{A}} : A^{\mathrm{arity}(\phi)} \rightharpoonup A_{\mathrm{sort}(\phi)}) \tag{2-1}$$

of a (typically infinite) *universe* A and an interpretation of the function symbols which matches their formal arities and sorts, i.e., for each $\phi \in \Phi$,

$$\phi^{\mathbf{A}} : A^n \rightharpoonup A_s \qquad (n = \mathrm{arity}(\phi), s = \mathrm{sort}(\phi), A_{\mathtt{ind}} = A, A_{\mathtt{boole}} = \{\mathrm{tt}, \mathrm{ff}\}).$$

An *expansion* of a Φ-structure $\mathbf{A}$ in (2-1) is any $\Phi \cup \Psi$-structure

$$(\mathbf{A}, \Psi^{\mathbf{A}}) = (A, \{\phi^{\mathbf{A}}\}_{\phi\in\Phi}, \{\psi^{\mathbf{A}}\}_{\psi\in\Psi}) \tag{2-2}$$

on the same universe A with $\Phi \cap \Psi = \emptyset$.

Syntax. The (pure, explicit) Φ^2-*terms* are defined by the *structural recursion*

$$F :\equiv \mathrm{tt} \mid \mathrm{ff} \mid \mathsf{v}_i \mid \mathsf{p}^{s,n}(F_1, \dots, F_n) \\ \mid \phi(F_1, \dots, F_{\mathrm{arity}(\phi)}) \mid \mathrm{if}\ F_0\ \mathrm{then}\ F_1\ \mathrm{else}\ F_2,$$

where $\mathsf{v}_0, \mathsf{v}_1, \dots$ is a fixed list of *individual variables*; for each $s \in \{\mathtt{ind}, \mathtt{boole}\}$ and each $n \in \mathbb{N}$, $\mathsf{p}_0^{s,n}, \mathsf{p}_1^{s,n}, \dots$ is a fixed list of (partial) function variables of sort s and arity n; each term is assigned a sort in the natural way; and a *Parsing Lemma* (like Problem x1E.1 of ARIC) justifies the standard method of defining functions on these terms by structural recursion.[6]

A **Φ-term** is a Φ^2-term which has no function variables and a $\Phi\cup\{\mathsf{p}_1, \dots, \mathsf{p}_K\}$-**term** is a Φ^2-term whose function variables are all in the list $\mathsf{p}_1, \dots, \mathsf{p}_K$; these terms are naturally interpreted in expansions $(\mathbf{A}, p_1, \dots, p_K)$ of Φ-structures which interpret each p_i by some p_i of the correct sort and arity.

An **extended** Φ^2-term is a pair

$$(F, \vec{\mathsf{x}}) \equiv F(\vec{\mathsf{x}}) \equiv F(\mathsf{x}_1, \dots, \mathsf{x}_n)$$

of a Φ^2-term F and a finite list of distinct individual variables which includes all the individual variables that occur in F. The notation provides a simple way to deal with substitutions,

$$F(E_1, \dots, E_n) :\equiv F\{\mathsf{x}_1 :\equiv E_1, \dots, \mathsf{x}_n :\equiv E_n\} \\ (F(\vec{\mathsf{x}})\ \text{an extended term},\ E_1, \dots, E_n\ \text{terms of sort}\ \mathtt{ind}).$$

Semantics. The *denotation* (or *value*) $\mathrm{den}((\mathbf{A}, \vec{p}), F(\vec{x}))$ in a Φ-structure $\mathbf{A}$ of each extended $\Phi \cup \{\vec{\mathsf{p}}\}$-term $F(\vec{\mathsf{x}})$ for given values $\vec{p}, \vec{x}$ of its variables is defined by the usual (compositional) structural recursion on the definition of terms: skipping the $(\mathbf{A}, \vec{p})$ part (which remains constant),

$$\mathrm{den}(\mathrm{tt}(\vec{x})) = \mathrm{tt},\ \mathrm{den}(\mathrm{ff}(\vec{x})) = \mathrm{ff},\ \mathrm{den}(\mathsf{x}_i(\vec{x})) = x_i,$$

$$\mathrm{den}(\mathsf{p}_i(F_1, \dots, F_n)(\vec{x})) = p_i(\mathrm{den}(F_1(\vec{x})), \dots, \mathrm{den}(F_n(\vec{x}))),$$

$$\mathrm{den}(\phi(F_1, \dots, F_{\mathrm{arity}(\phi)})(\vec{x})) = \phi^{\mathbf{A}}(\mathrm{den}(F_1(\vec{x})), \dots, \mathrm{den}(F_{\mathrm{arity}(\phi)}(\vec{x}))),$$

$$\mathrm{den}(\mathrm{if}\ F_0\ \mathrm{then}\ F_1\ \mathrm{else}\ F_2(\vec{x})) = \mathrm{if}\ \mathrm{den}(F_0(\vec{x}))\ \mathrm{then}\ \mathrm{den}(F_1(\vec{x}))\ \mathrm{else}\ \mathrm{den}(F_2(\vec{x}));$$

[6] See Problem x2.1 for a detailed statement and proof of this. We just write ϕ for $\phi(\ )$ when $\mathrm{arity}(\phi) = 0$, $\mathsf{p}^{s,0}$ for $\mathsf{p}^{s,0}(\ )$ and we do not allow variables over the set $\mathbb{B} = \{\mathrm{ff}, \mathrm{tt}\}$ of truth values, cf. footnote 9 on p. 37 of ARIC.

and we will also use standard model-theoretic notation,

$$\begin{aligned}(\mathbf{A},\vec{p}) \models F(\vec{x}) = G(\vec{x}) &\iff_{\text{df}} \text{den}((\mathbf{A},\vec{p}), F(\vec{x})) = \text{den}((\mathbf{A},\vec{p}), G(\vec{x})),\\ \mathbf{A} \models F(\vec{\mathsf{x}}) = G(\vec{\mathsf{x}}) &\iff_{\text{df}} (\forall \vec{p}, \vec{x})\Big((\mathbf{A},\vec{p}) \models F(\vec{x}) = G(\vec{x}\Big)\\ \models F(\vec{\mathsf{x}}) = G(\vec{\mathsf{x}}) &\iff_{\text{df}} (\forall \mathbf{A})\Big(\mathbf{A} \models F(\vec{\mathsf{x}}) = G(\vec{\mathsf{x}})\Big).\end{aligned}$$

Problems for Section 2.

Problem x2.1. A set $\mathcal{T}$ of pairs (F, s) is closed under *term formation* if

$$(\text{tt}, \texttt{boole}), (\text{ff}, \texttt{boole}) \in \mathcal{T}, \text{ for all } i, (\mathsf{v}_i, \texttt{ind}) \in \mathcal{T},$$

$$\text{for all } \phi \in \Phi, \Big((F_1, \texttt{ind}), \dots, (F_{\text{arity}(\phi)}, \texttt{ind}) \in \mathcal{T}$$
$$\Longrightarrow (\phi(F_1, \dots, F_{\text{arity}(\phi)}), \text{sort}(\phi)) \in \mathcal{T}\Big),$$

$$\text{for all } \mathsf{p}_i^{s,n}, \Big((F_1, \texttt{ind}), \dots, (F_n, \texttt{ind}) \in \mathcal{T}$$
$$\Longrightarrow (\mathsf{p}_i^{s,n}(F_1, \dots, F_n), s) \in \mathcal{T}\Big)$$

$$\text{and } \Big((F_1, \texttt{boole}), (F_2, s), (F_3, s) \in \mathcal{T} \Longrightarrow (\text{if } F_1 \text{ then } F_2 \text{ else } F_3, s) \in \mathcal{T}\Big),$$

where we view tt, ff and v_i as *strings of symbols* of length 1; and a string F is a *pure, explicit* Φ^2*-term of sort* s if the pair (F, s) belongs to every set $\mathcal{T}$ which is closed under term formation.

Formulate a *Parsing Lemma* for pure, explicit Φ^2-terms (as in Problem x1E.1 of ARIC for terms without function variables) and give a complete proof of it using this precise definition.

§3. Continuous, pure recursors.

For any two sets X, W, a (continuous, pure) *recursor* on X to W is a tuple

(3-1) $$\alpha = (\alpha_0, \dots, \alpha_K) : X \rightsquigarrow W,$$

such that for suitable sets $Y_1^\alpha, W_1^\alpha, \dots, Y_K^\alpha, W_K^\alpha$,[7]

$$\begin{aligned}\alpha_0 &: X \times (Y_1^\alpha \rightharpoonup W_1^\alpha) \times \cdots \times (Y_K^\alpha \rightharpoonup W_K^\alpha) \rightharpoonup W, \text{ and}\\ \alpha_i &: (Y_1^\alpha \rightharpoonup W_1^\alpha) \times \cdots \times (Y_K^\alpha \rightharpoonup W_K^\alpha) \rightharpoonup W_i^\alpha \quad (1 \le i \le K)\end{aligned}$$

are continuous functionals. We allow $X = \mathbf{I} = \{\emptyset\}$ (as on page 9 of ARIC), in which case, by our conventions, $\alpha_0 : (Y_1^\alpha \rightharpoonup W_1^\alpha) \times \cdots \times (Y_K^\alpha \rightharpoonup W_K^\alpha) \rightharpoonup W$.

The number K is the *dimension* of α, α_0 is its *output* or *head functional*, the poset $(Y_1 \rightharpoonup W_1) \times \cdots \times (Y_K \rightharpoonup W_K)$ is its *solution space*, and we allow $K = 0$–in which case there is no solution space and (α_0) is naturally identified with the partial function $\alpha_0 : X \rightharpoonup W$. With the notation of (1B-6) of ARIC, α *defines* (or *computes*) the partial function $\overline{\alpha} : X \rightharpoonup W$, where

(3-2) $$\overline{\alpha}(x) = \alpha_0(x, \vec{p}) \ \overline{\text{where}} \Big\{p_1(y_1) = \alpha_1(y_1, \vec{p}), \dots, p_K(y_K) = \alpha_K(y_K, \vec{p})\Big\}.$$

[7]For the definitions of *monotone* and *continuous* functionals see Section 1A of ARIC.

We can summarize this situation succinctly by writing

$$\text{(3-3)}\quad \begin{aligned}\alpha(x) &= (\alpha_0,\dots,\alpha_K)(x)\\ &= \alpha_0(x,\vec{p})\ \textit{where}\ \Big\{p_1(y_1)=\alpha_1(y_1,\vec{p}),\dots,p_K(y_K)=\alpha_K(y_K,\vec{p})\Big\},\end{aligned}$$

where "*where*" is now an operation which assigns to every tuple of continuous functionals $(\alpha_0,\dots,\alpha_K)$ (with suitable input and output sets) a recursor.

The recursor defined by an extended program. Every (deterministic) extended recursive Φ-program

$$E(\vec{\mathsf{x}}) \equiv E_0(\vec{\mathsf{x}})\ \mathsf{where}\ \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1)=E_1,\dots,\mathsf{p}_K(\vec{\mathsf{x}}_K)=E_K\Big\}$$

(as in (1-4)) with dimension K, arity n and sort s defines naturally on each Φ-structure $\mathbf{A}$ the continuous pure *recursor on* A of sort s and arity n

$$\text{(3-4)}\qquad \mathfrak{r}(\mathbf{A},E(\vec{\mathsf{x}})) = (\alpha_0,\alpha_1,\dots,\alpha_K) : A^n \rightsquigarrow A_s,$$

where

$$\text{(3-5)}\qquad \begin{aligned}\alpha_0(\vec{x},\vec{p}) &= \mathrm{den}((\mathbf{A},\vec{p}),E_0(\vec{x})),\\ \alpha_i(\vec{x}_i,\vec{p}) &= \mathrm{den}((\mathbf{A},\vec{p}),E_i(\vec{x}_i))\quad (1\le i\le K)\end{aligned}$$

or, with the notation in (3-3),

$$\text{(3-6)}\quad \begin{aligned}\mathfrak{r}(\mathbf{A},E(\vec{x})) = \mathrm{den}&((\mathbf{A},\vec{p}),E_0(\vec{x}))\\ &\textit{where}\ \Big\{p_1(\vec{x}_1)=\mathrm{den}((\mathbf{A},\vec{p}),E_1(\vec{x}_1)),\\ &\qquad\dots,p_K(\vec{x}_K)=\mathrm{den}((\mathbf{A},\vec{p}),E_K(\vec{x}_K))\Big\}.\end{aligned}$$

It is immediate from the semantics of recursive programs on page 51 of ARIC that the recursor of an extended program computes its denotation,

$$\text{(3-7)}\qquad \overline{\mathfrak{r}(\mathbf{A},E(\vec{x}))} = \mathrm{den}(\mathbf{A},E(\vec{x}))\quad (\vec{x}\in A^n).$$

However : $\mathfrak{r}(\mathbf{A},E(\vec{\mathsf{x}}))$ *does not typically model faithfully the algorithm expressed by* $E(\vec{\mathsf{x}})$ *on* $\mathbf{A}$, partly because it does not take into account the *explicit steps* which may be required to computes $\mathrm{den}(\mathbf{A},E(\vec{x}))$ and (more importantly) because *it does not record the dependence of that algorithm on the primitives of* $\mathbf{A}$. In the extreme case, if $E\equiv E_0$ where $\{\ \}$ is a program with empty body (an explicit Φ-term), then

$$\mathfrak{r}(\mathbf{A},E(\vec{\mathsf{x}})) = ((\lambda\vec{x})\,\mathrm{den}(\mathbf{A},E(\vec{x})))$$

is a trivial recursor of dimension 0, a partial function on A—and it is the same for all explicit terms which define this partial function from any primitives, which is certainly not right. In the next Section 4 we will define the *intension* $\mathrm{int}(\mathbf{A},E(\vec{\mathsf{x}}))$ of $E(\vec{\mathsf{x}})$ in $\mathbf{A}$ which (we will claim) models faithfully the algorithm from the primitives of $\mathbf{A}$ *expressed* by $E(\vec{\mathsf{x}})$. As it turns out, however,

$$\mathrm{int}(\mathbf{A},E(\vec{\mathsf{x}})) = \mathfrak{r}(\mathbf{A},\mathrm{cf}(E(\vec{\mathsf{x}})))$$

for some extended program $\mathrm{cf}(E(\vec{\mathsf{x}}))$ which is canonically associated with $E(\vec{\mathsf{x}})$, and so every recursive algorithm of a structure $\mathbf{A}$ is $\mathfrak{r}(\mathbf{A},F(\vec{\mathsf{x}})))$ for some $F(\vec{\mathsf{x}})$.

Strong recursor isomorphism. The solutions of the system of recursive equations in the body of a recursor α as in (3-3) do not depend on the order in which these equations are listed and the known methods for computing them also do not depend on that order in any important way; so it is natural to identify recursors which differ only in the order in which their bodies are enumerated, so that, for example

$$\begin{aligned}\alpha_0(x,p_1,p_2,p_3) \textit{ where } \Big\{ & p_1(y_1) = \alpha_1(y_1,p_1,p_2,p_3),\\ & p_2(y_2) = \alpha_2(y_2,p_1,p_2,p_3),\ p_3(y_3) = \alpha_3(y_3,p_1,p_2,p_3)\Big\}\\ = \alpha_0(x,p_1,p_2,p_3) \textit{ where } \Big\{ & p_3(y_3) = \alpha_3(y_3,p_1,p_2,p_3),\\ & p_1(y_1) = \alpha_1(y_1,p_1,p_2,p_3),\ p_2(y_2) = \alpha_2(y_2,p_1,p_2,p_3)\Big\}.\end{aligned}$$

The precise definition of this equivalence relation is a bit messy in the general case: two recursors $\alpha, \beta : X \rightsquigarrow W$ on the same input and output sets are *strongly isomorphic*—or just *equal*—if they have the same dimension K and there is a permutation $\pi : \{1,\dots,K\} \rightarrowtail\!\!\!\!\rightarrow \{1,\dots,K\}$ with inverse $\rho = \pi^{-1}$ such that

$$\begin{gathered} Y_i^\beta = Y_{\pi(i)}^\alpha, \quad W_i^\beta = W_{\pi(i)}^\alpha \quad (i = 1,\dots,K),\\ \beta_0(x,q_1,\dots,q_K) = \alpha_0(x,q_{\rho(1)},\dots,q_{\rho(K)}),\\ \beta_i(y_{\pi(i)},q_1,\dots,q_K) = \alpha_{\pi(i)}(y_{\pi(i)},q_{\rho(1)},\dots,q_{\rho(K)}),\ (1 \le i \le K).\end{gathered} \tag{3-8}$$

Directly from the definitions, strongly isomorphic recursors $\alpha, \beta : X \rightsquigarrow W$ compute the same partial function $\overline{\alpha} = \overline{\beta} : X \rightharpoonup W$; the idea, of course, is that *strongly isomorphic recursors model the same algorithm*, and so we will simply write $\alpha = \beta$ to indicate that α and β are strongly isomorphic. This view is discussed in several of the articles cited above and we will not go into it here.[8]

For the recursors defined by recursive programs, this definition takes the following form, where for any Φ^2-term F and sequences of distinct function and individual variables $\vec{\mathsf{q}}, \vec{\mathsf{p}}, \vec{\mathsf{y}}, \vec{\mathsf{x}}$ (satisfying the obvious sort and arity conditions),

$$\begin{aligned} \text{(3-9)}\quad & F\{\vec{\mathsf{q}} :\equiv \vec{\mathsf{p}}, \vec{\mathsf{y}} :\equiv \vec{\mathsf{x}}\}\\ & :\equiv \text{the result of replacing in } F \text{ every } \mathsf{q}_i \text{ by } \mathsf{p}_i \text{ and every } \mathsf{y}_j \text{ by } \mathsf{x}_j.\end{aligned}$$

Lemma 3.1. *Two extended Φ-programs*

$$\begin{cases} E(\vec{\mathsf{x}}) \equiv E_0(\vec{\mathsf{x}}) \text{ where } \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1,\dots,\mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\},\\ F(\vec{\mathsf{x}}) \equiv F_0(\vec{\mathsf{x}}) \text{ where } \Big\{\mathsf{q}_1(\vec{\mathsf{y}}_1) = F_1,\dots,\mathsf{q}_{K'}(\vec{\mathsf{y}}_{K'}) = E_{K'}\Big\}\end{cases} \tag{3-10}$$

of the same sort and arity define the same recursor on a Φ-structure $\mathbf{A}$ exactly when they have the same number of parts ($K' = K$) and there is a permutation $\pi : \{0,\dots,K\} \rightarrowtail\!\!\!\!\rightarrow \{0,\dots,K\}$ with inverse $\rho = \pi^{-1}$ such that $\pi(0) = 0$ and for

[8]This finest relation of recursor equivalence was introduced (for monotone recursors) in Moschovakis [1989b], and more general, weaker notions were considered in Moschovakis [1998], [2001] and in Moschovakis and Paschalis [2008].

all $\vec{p}$ and $\vec{x}$,

$$\text{(3-11)}\quad \begin{array}{l}(\mathbf{A},\vec{p}) \models E_0(\vec{x}) = F_0\{\vec{\mathsf{q}} :\equiv \rho(\vec{\mathsf{p}})\}(\vec{x}),\\ (\mathbf{A},\vec{p}) \models E_i(\vec{x}_i) = F_{\pi(i)}\{\vec{\mathsf{q}} :\equiv \rho(\vec{\mathsf{p}}), \vec{\mathsf{y}}_i :\equiv \vec{\mathsf{x}}_i\}(\vec{x}_i), \quad (i = 1,\dots,K),\end{array}$$

where $\rho(\mathsf{p}_1,\dots,\mathsf{p}_K) = (\mathsf{p}_{\rho(1)},\dots,\mathsf{p}_{\rho(K)})$.

THE PROOF is an exercise in managing messy notations and we leave it for Problem x3.1. ⊣

(Extended) ***program congruence.*** Two programs E and F are *congruent* if F can be constructed from E by an alphabetic change (renaming) of the bound individual and function variables in its parts (as in (3-9)) and a permutation of the equations in the body of E. Congruence is obviously an equivalence relation on programs, congruent programs have the same free variables and we write

$$E \equiv_c F \iff_{\text{df}} E \text{ and } F \text{ are congruent},$$
$$E(\vec{\mathsf{x}}) \equiv_c F(\vec{\mathsf{y}}) \iff_{\text{df}} \vec{\mathsf{x}} \equiv \vec{\mathsf{y}} \text{ and } E \equiv_c F.$$

By Lemma 3.1, congruent extended programs define equal recursors in every structure $\mathbf{A}$, i.e., for every extended program $E(\vec{\mathsf{x}})$ and every program F,

$$\text{(3-12)}\qquad E \equiv_c F \Longrightarrow (\forall \mathbf{A})[\mathfrak{r}(\mathbf{A}, E(\vec{\mathsf{x}})) = \mathfrak{r}(\mathbf{A}, F(\vec{\mathsf{x}}))],$$

but the converse fails, cf. Problem x3.2.

More general recursors. The definition of recursors we gave is basically the one in Moschovakis [1989b], except that we allowed there the parts α_i of α to be (monotone but) not continuous and to depend on the input set X, i.e.,

$$(\star)\qquad \alpha = (\alpha_0, \alpha_1, \dots, \alpha_K) : X \rightsquigarrow W,$$

such that for suitable sets $Y_1^\alpha, W_1^\alpha, \dots, Y_K^\alpha, W_K^\alpha$,

$$\alpha_0 : (Y_1^\alpha \rightharpoonup W_1^\alpha) \times \cdots \times (Y_K^\alpha \rightharpoonup W_K^\alpha) \times X \rightharpoonup W, \text{ and}$$
$$\alpha_i : Y_i^\alpha \times (Y_1^\alpha \rightharpoonup W_1^\alpha) \times \cdots \times (Y_K^\alpha \rightharpoonup W_K^\alpha) \times X \rightharpoonup W_i^\alpha \quad (1 \le i \le K).$$

Allowing the parts to be discontinuous is necessary for the theory of higher-type recursion which was the topic of Moschovakis [1989a], [1989b], but their dependence on the input set X is not: the simpler, present notion is easier to work with and it includes the more general objects if we identify α in $(\star)$ with

$$\alpha' = (\alpha'_0, \alpha'_1, \dots, \alpha'_K) : X \rightsquigarrow W$$

on the sets $Y'_i = X \times Y_i, W'_i = W_i$ $(1 \le i \le K)$ with

$$\alpha'_0(x, p_1, \dots, p_K) = \alpha_0(\lambda y_1 p(x, y_1), \dots, \lambda y_K p(x, y_K), x),$$
$$\alpha'_i(x, y_i, p_1, \dots, p_K) = \alpha_i(\lambda y_1 p(x, y_1), \dots, \lambda y_K p(x, y_K), x) \quad (1 \le i \le K).$$

Substantially more general notions of (monotone and continuous) recursors whose solution spaces are products of arbitrary complete posets were introduced in Moschovakis [1998], [2001] and (in greater detail) in Moschovakis and Paschalis [2008]; and it is routine to define *nondeterministic algorithms* along the same lines, using the fixpoint semantics of nondeterministic programs on page 84 of ARIC.

Problems for Section 3.

Problem x3.1. Prove Lemma 3.1. HINT: Check by a (routine) induction on terms, that *for any two lists of distinct function variables* $\vec{\mathsf{p}} \equiv \mathsf{p}_1, \dots, \mathsf{p}_K$ *and* $\vec{\mathsf{q}} \equiv \mathsf{q}_1, \dots, \mathsf{q}_K$ *and individual variables* $\vec{\mathsf{y}} \equiv \mathsf{y}_1, \dots, \mathsf{y}_m$*,* $\vec{\mathsf{x}} \equiv \mathsf{x}_1, \dots, \mathsf{x}_m$*, and for every* $\Phi \cup \{\vec{\mathsf{q}}\}$*-term* M *whose free variables are all in the list* $\vec{\mathsf{y}}$*,*

$$\text{if } \beta(\vec{y}, \vec{q}) = \text{den}((\mathbf{A}, \vec{q}), M(\vec{y})),$$
$$\text{then } \beta(\vec{x}, \vec{p}) = \text{den}((\mathbf{A}, \vec{p}), M\{\vec{\mathsf{q}} :\equiv \vec{\mathsf{p}}, \vec{\mathsf{y}} :\equiv \vec{\mathsf{x}}\}(\vec{x})),$$

assuming, of course, that the sorts and arities of the function variables are such that the claim makes sense.

Problem x3.2. Prove that for every explicit term E of sort `boole`,

$$\models \text{if } E \text{ then } E \text{ else } E = E$$

and use this to define two extended programs $E(\mathsf{x}), F(\mathsf{x})$ such that for all $\mathbf{A}$, $\mathfrak{r}(\mathbf{A}, E(\vec{\mathsf{x}})) = \mathfrak{r}(\mathbf{A}, F(\vec{\mathsf{x}}))$ but $E \not\equiv_c F$.

§4. Canonical forms and intensions.

In this section we will associate with each extended (recursive) Φ-program

$$E(\vec{\mathsf{x}}) \equiv E_0(\vec{\mathsf{x}}) \text{ where } \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1, \dots, \mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\} \tag{4-1}$$

a (unique up to congruence) *canonical form* $\text{cf}(E(\vec{\mathsf{x}}))$, so that the construction

$$(\mathbf{A}, E(\vec{\mathsf{x}})) \mapsto \mathfrak{r}(\mathbf{A}, \text{cf}(E(\vec{\mathsf{x}})))$$

captures the algorithm expressed by $E(\vec{\mathsf{x}})$ in any Φ-structure $\mathbf{A}$. This theory of canonical forms yields, in particular, a robust notion of *the elementary* (first-order) *algorithms* of a structure $\mathbf{A}$.

Some more syntax. For the syntactic work we will do in this Section we need to enrich and simplify the notation of ARIC summarized in Section 2.

We introduce two function symbols cond_s of arity 3 and sort s ($\equiv$ `ind` or `boole`) and the abbreviations

$$\mathsf{cond}_s(F_0, F_1, F_2) :\equiv \text{if } F_0 \text{ then } F_1 \text{ else } F_2;$$

this is semantically inappropriate because the conditionals do not define (strict) partial functions, but it simplifies considerably the definition of pure, explicit Φ^2-terms on page 6 which now takes the form

(Pure explicit terms) $\qquad F :\equiv \mathrm{tt} \mid \mathrm{ff} \mid \mathsf{v}_i \mid \mathsf{c}(F_1, \dots, F_n)$,

where c is any n-ary function symbol ϕ in the vocabulary Φ, or an n-ary function variable $\mathsf{p}_i^{s,n}$ or cond_s (and $n = 3$).

Set representations. With each extended Φ-program $E(\vec{\mathsf{x}})$ as in (4-1), we associate its *set representation*

$$\text{set}(E(\vec{\mathsf{x}})) = \Big\{E_0(\vec{\mathsf{x}}),\ \mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1, \dots,\ \mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\}. \tag{4-2}$$

Notice that $\text{set}(E(\vec{\mathsf{x}}))$ determines $E_0(\vec{\mathsf{x}})$, its only member which is not an equation, and by the definition of the congruence relation on page 10,

$$\text{set}(E(\vec{\mathsf{x}})) = \text{set}(F(\vec{\mathsf{x}})) \Longrightarrow E \equiv_c F;$$

the converse implication fails, but the algorithm which computes canonical forms naturally operates on these set representations of extended terms, so having a notation for them simplifies greatly its specification.

Immediate and irreducible terms. A term I is *immediate* if it is an individual variable or a direct call to a recursive variable applied to individual variables,

(Immediate terms) $\qquad I :\equiv \mathsf{v}_i \mid \mathsf{p}^{s,0} \mid \mathsf{p}_i^{n,s}(\mathsf{u}_1, \dots, \mathsf{u}_n);$

and a term F is *irreducible* if it is immediate, a Boolean constant, or of the form $\mathsf{c}(I_1, \dots, I_l)$ with immediate $I_1, \dots, I_n$,[9]

(Irreducible terms) $\qquad F :\equiv I \mid \mathrm{tt} \mid \mathrm{ff} \mid \mathsf{c}(I_1, \dots, I_n)$

For example, $\mathsf{u}, \mathsf{p}(\mathsf{u}_1, \mathsf{u}_2, \mathsf{u}_1)$ are immediate, $\phi(\mathsf{u})$ and $\mathsf{p}(\mathsf{u}, \mathsf{q}(\mathsf{v}), \mathsf{r}(\mathsf{u}))$ are irreducible but not immediate, and $\mathsf{p}(u, \phi(\mathsf{v}))$ is reducible.

Computationally, we think of immediate terms as "generalized variables" which can be accessed directly, like array entries $a[i], b[i, j, i]$ in some programming languages, and a term is irreducible if it can be computed by direct (not nested) calls to primitives or to recursive variables. Both of these properties of terms are (trivially) preserved by alphabetic change of variables (3-9),

(4-3) *if F is immediate (irreducible),*

then $F\{\vec{\mathsf{q}} :\equiv \vec{\mathsf{p}}, \vec{\mathsf{y}} :\equiv \vec{\mathsf{x}}\}$ is also immediate (irreducible).

Irreducible extended programs. An extended program

$$E(\vec{\mathsf{x}}) \equiv E_0(\vec{\mathsf{x}}) \text{ where } \Big\{\mathsf{p}_1(\vec{\mathsf{x}}_1) = E_1, \dots, \mathsf{p}_K(\vec{\mathsf{x}}_K) = E_K\Big\}$$

is *irreducible* if $E_0, E_1, \dots, E_K$ are all irreducible terms. These are the extended programs which cannot be reduced by the basic process of *reduction*, to which we turn next.

Arrow reduction. We first define the simplest reduction relation

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}), \quad \vec{\mathsf{x}} \equiv (\mathsf{x}_1, \dots, \mathsf{x}_n)$$

on extended programs with the same list of n free variable, where p, q are function variables and j is a number. The definition splits in two cases on the triple (E, p, j) and it helps to call a function variable r *fresh* if it is none of the function variables p_i in the body of E.

(1) *The body case: one of the equations in the body of E is*

$$\mathsf{p}(\vec{\mathsf{x}}_\mathsf{p}) = \mathsf{c}(G_1, \dots, G_{j-1}, G_j, G_{j+1}, \dots, G_l)$$

and the term G_j is not immediate. Put

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \iff_{\mathrm{df}} \mathsf{q} \text{ is fresh, } \mathrm{arity}(\mathsf{q}) = \mathrm{arity}(\mathsf{p}), \ \mathrm{sort}(\mathsf{q}) = \mathrm{sort}(\mathsf{p}), \text{ and}$$
$$\mathrm{set}(F(\vec{\mathsf{x}})) = \Big(\mathrm{set}(E(\vec{\mathsf{x}})) \setminus \Big\{\mathsf{p}(\vec{\mathsf{x}}_\mathsf{p}) = \mathsf{c}(G_1, \dots, G_{j-1}, G_j, G_{j+1}, \dots, G_l)\Big\}\Big)$$
$$\cup \Big\{\mathsf{p}(\vec{\mathsf{x}}_\mathsf{p}) = \mathsf{c}(G_1, \dots, G_{j-1}, \mathsf{q}(\vec{\mathsf{x}}_\mathsf{p}), G_{j+1}, \dots, G_l), \quad \mathsf{q}(\vec{\mathsf{x}}_\mathsf{p}) = G_j\Big\}.$$

[9]There are natural sort restrictions in these definitions and assignments of sorts to immediate and irreducible terms that we will suppress in the notation, cf. Problem x4.1.

$$\begin{array}{rl}
E(\vec{\mathsf{x}}) : & \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = \mathsf{c}(G_1, G_2, G_3) \\
& \xrightarrow{(\mathsf{p},1,\mathsf{q})} \\
F(\vec{\mathsf{x}}) : & \begin{array}{l} \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = \mathsf{c}(\mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}), G_2, G_3) \\ \mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}) = G_1 \end{array} \\
& \xrightarrow{(\mathsf{p},3,\mathsf{q}')} \\
H(\vec{\mathsf{x}}) : & \begin{array}{l} \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = \mathsf{c}(\mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}), G_2, \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}})) \\ \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}}) = G_3 \\ \mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}) = G_1 \end{array}
\end{array}
\qquad
\begin{array}{rl}
E(\vec{\mathsf{x}}) : & \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = \mathsf{c}(G_1, G_2, G_3) \\
& \xrightarrow{(\mathsf{p},3,\mathsf{q}')} \\
F'(\vec{\mathsf{x}}) : & \begin{array}{l} \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = \mathsf{c}(G_1, G_2, \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}})) \\ \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}}) = G_3 \end{array} \\
& \xrightarrow{(\mathsf{p},1,\mathsf{q})} \\
H(\vec{\mathsf{x}}) : & \begin{array}{l} \mathsf{p}(\vec{\mathsf{x}}_{\mathsf{p}}) = c(\mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}), G_2, \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}})) \\ \mathsf{q}(\vec{\mathsf{x}}_{\mathsf{p}}) = G_1 \\ \mathsf{q}'(\vec{\mathsf{x}}_{\mathsf{p}}) = G_3 \end{array}
\end{array}$$

Diagram 1. Amalgamation example, the body case.

(2) *The head case*: p *and* q *are both fresh,* $\text{arity}(\mathsf{q}) = n$*,* $\text{sort}(\mathsf{q}) = \text{sort}(E_0)$,

$$E_0 \equiv \mathsf{c}(G_1, \dots, G_{j-1}, G_j, G_{j+1}, \dots, G_l)$$

and the term G_j *is not immediate.* Put

$$\begin{aligned}
&E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \\
&\quad \Longleftrightarrow_{\text{df}} \text{set}(F(\vec{\mathsf{x}})) = \Big(\text{set}(E(\vec{\mathsf{x}})) \setminus \Big\{\mathsf{c}(G_1, \dots, G_{j-1}, G_j, G_{j+1}, \dots, G_l)\Big\}\Big) \\
&\qquad\qquad \cup \Big\{\mathsf{c}(G_1, \dots, G_{j-1}, \mathsf{q}(\vec{\mathsf{x}}), G_{j+1}, \dots, G_l), \quad \mathsf{q}(\vec{\mathsf{x}}) = G_j\Big\}.
\end{aligned}$$

Notice that in the head case, the specific function variable p does not occur on the right-hand-side of the definition as it does in the body case, it is used only as a "marker" that this is the head case: we put

$$\begin{aligned}
\text{(4-4)} \quad \mathsf{p} \sim_E \mathsf{p}' \Longleftrightarrow_{\text{df}} &\text{ either } \mathsf{p} \equiv \mathsf{p}' \in \{\mathsf{p}_1, \dots, \mathsf{p}_K\} \quad \text{(the body case)} \\
&\text{ or } \{\mathsf{p}, \mathsf{p}'\} \cap \{\mathsf{p}_1, \dots, \mathsf{p}_K\} = \emptyset \quad \text{(the head case)}.
\end{aligned}$$

Lemma 4.1. (1) *If* $E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}})$ *and* $E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F'(\vec{\mathsf{x}})$*, then* $F \equiv_c F'$.

(2) *If* q' *is fresh,* $\text{arity}(\mathsf{q}') = \text{arity}(\mathsf{q})$ *and* $\text{sort}(\mathsf{q}') = \text{sort}(\mathsf{q})$*, then*

$$\text{(4-5)} \qquad E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \Longrightarrow E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q}')} F\{\mathsf{q} :\equiv \mathsf{q}'\}(\vec{\mathsf{x}}).$$

(3) *If* r' *is fresh and* $\mathsf{r}' \not\equiv \mathsf{q}$*, then*

$$\text{(4-6)} \qquad E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \Longrightarrow E\{\mathsf{r} :\equiv \mathsf{r}'\}(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F\{\mathsf{r} :\equiv \mathsf{r}'\}(\vec{\mathsf{x}}).$$

(4) *An extended program* $E(\vec{\mathsf{x}})$ *is irreducible if and only if there are no* $\mathsf{p}, j, \mathsf{q}, F$ *such that* $E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}})$.

(5) *For every extended program* $E(\vec{\mathsf{x}})$*, every program* F*, every triple* $(\mathsf{p}, j, \mathsf{q})$ *and every structure* $\mathbf{A}$*,*

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \Longrightarrow \text{den}(\mathbf{A}, E(\vec{x})) = \text{den}(\mathbf{A}, F(\vec{x})) \quad (\vec{x} \in A^n).$$

Proof. (1) – (3) are trivial, once we get through the notation and (4) follows immediately by the definition. (5) is also quite easy, either by a direct, fixpoint argument or by applying the *Head* and *Bekič-Scott* rules of Theorem 1B.2 of ARIC. ⊣

$$
\begin{array}{rl@{\qquad\qquad}rl}
E(\vec{\mathsf{x}}) : & \mathsf{c}(G_1, G_2, G_3) & E(\vec{\mathsf{x}}) : & \mathsf{c}(G_1, G_2, G_3) \\
& \xrightarrow{(\mathsf{p},1,\mathsf{q})} & & \xrightarrow{(\mathsf{p},3,\mathsf{q}')} \\
F(\vec{\mathsf{x}}) : & \begin{array}{r} \mathsf{c}(\mathsf{q}(\vec{\mathsf{x}}), G_2, G_3) \\ \mathsf{q}(\vec{\mathsf{x}}) = G_1 \end{array} & F'(\vec{\mathsf{x}}) : & \begin{array}{r} \mathsf{c}(G_1, G_2, \mathsf{q}'(\vec{\mathsf{x}})) \\ \mathsf{q}'(\vec{\mathsf{x}}) = G_3 \end{array} \\
& \xrightarrow{(\mathsf{p},3,\mathsf{q}')} & & \xrightarrow{(\mathsf{p},1,\mathsf{q})} \\
H(\vec{\mathsf{x}}) : & \begin{array}{r} \mathsf{c}(\mathsf{q}(\vec{\mathsf{x}}), G_2, \mathsf{q}'(\vec{\mathsf{x}})) \\ \mathsf{q}'(\vec{\mathsf{x}}) = G_3 \\ \mathsf{q}(\vec{\mathsf{x}}) = G_1 \end{array} & H(\vec{\mathsf{x}}) : & \begin{array}{r} c(\mathsf{q}(\vec{\mathsf{x}}), G_2, \mathsf{q}'(\vec{\mathsf{x}}) \\ \mathsf{q}(\vec{\mathsf{x}}) = G_1 \\ \mathsf{q}'(\vec{\mathsf{x}}) = G_3 \end{array}
\end{array}
$$

Diagram 2. Amalgamation example, the head case.

The key property of the reduction calculus is the following, simple

Lemma 4.2 (Amalgamation). *If* ($\mathsf{p} \not\sim_E \mathsf{p}'$ *or* $j \neq j'$), $\mathsf{q} \not\equiv \mathsf{q}'$,

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \text{ and } E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q}')} F'(\vec{\mathsf{x}}), \tag{4-7}$$

then there is a program H *such that*

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q}')} H(\vec{\mathsf{x}}) \text{ and } E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q}')} F'(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{p})} H(\vec{\mathsf{x}}). \tag{4-8}$$

Proof. This is obvious if $\mathsf{p} \not\sim_E \mathsf{p}'$, so that the two assumed reductions operate independently on different parts of E and they give the same result if they are executed in either order. For the proof in the more interesting case when $\mathsf{p} \sim_E \mathsf{p}'$, we just put down as examples the relevant sequences of needed replacements in Diagram 1 for the body case when p is ternary, $j = 1$ and $j' = 3$, and in Diagram 2 for the head case when $j = 1$ and $j' = 3$. The general case is only notationally more complex. ⊣

Extended program reduction. Using now arrow reduction, we define the **one-step** and **full** reduction relations on extended programs by

$$E(\vec{\mathsf{x}}) \Rightarrow_1 F(\vec{\mathsf{x}}) \iff_{\text{df}} \text{ for some } \mathsf{p}, j, \mathsf{q}, E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F(\vec{\mathsf{x}}), \tag{4-9}$$

$$\begin{aligned} E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}}) \iff_{\text{df}} & E \equiv_c F \\ & \text{or } (\exists F^1, \dots, F^k)[E(\vec{\mathsf{x}}) \Rightarrow_1 F^1(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^k(\vec{\mathsf{x}}) \text{ and } F^k \equiv_c F]. \end{aligned} \tag{4-10}$$

It is immediate from Lemma 4.1 that for every structure $\mathbf{A}$, every extended program $E(\vec{\mathsf{x}})$ and every F,

$$E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}}) \Longrightarrow \mathrm{den}(\mathbf{A}, E(\vec{x})) = \mathrm{den}(\mathbf{A}, F(\vec{x})) \quad (\vec{x} \in A^n), \tag{4-11}$$

but this is true even if we removed the all-important, non-immediacy restrictions in the definition of reduction, by Theorem 1B.2 of ARIC. The claim is that

if $E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}})$,

then $E(\vec{\mathsf{x}})$ *and* $F(\vec{\mathsf{x}})$ *express the same algorithm in every* Φ*-structure*,

but we will not get here into defending this naive view beyond what we said in Sections 1 and 3.

Caution. Reduction is a syntactic operation on extended recursive programs which models (very abstractly) *partial compilation*, bringing the mutual recursion expressed by the program to a useful form before the recursion is implemented *without committing to any particular method of implementation.* No real computation is done by it, and it does not embody any "optimization" of the algorithm expressed by an extended program, cf. Problem x4.3.

Size. Let size(E) be *the number of occurrences of non-immediate proper subterms*[10] of some part of a program E. For example, if

$$E \equiv \mathsf{p}(\mathsf{x}, \boxed{\phi_0(\mathsf{x})}) \text{ where } \Big\{\mathsf{p}(\mathsf{x},\mathsf{y}) = \text{if } \boxed{\text{test}(\boxed{\phi_0(\mathsf{x})}, \mathsf{y})} \text{ then } \mathsf{y} \text{ else } \boxed{\mathsf{p}(\mathsf{x}, \boxed{\sigma(\mathsf{x},\mathsf{y})})}\Big\}, \tag{4-12}$$

with $\Phi = \{\phi_0, \text{test}, \sigma\}$, then size($E$) = 5, because the proper subterms we count are the boxed $\phi_0(\mathsf{x})$ (twice), test$(\phi_0(\mathsf{x}), \mathsf{y})$, $\sigma(\mathsf{x},\mathsf{y})$, and $\mathsf{p}(\phi_0(\mathsf{x}), \sigma(\mathsf{x},\mathsf{y}))$.

From the definition of reduction, easily, an extended program $E(\vec{\mathsf{x}})$ is irreducible exactly when size$(E) = 0$ and each one-step reduction lowers size by 1, so, trivially:

Lemma 4.3. *If* $E(\vec{\mathsf{x}}) \Rightarrow_1 F^1(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^k(\vec{\mathsf{x}})$ *is a sequence of one-step reductions starting with* $E(\vec{\mathsf{x}})$, *then* $k \le \text{size}(E)$; *and* $F^k(\vec{\mathsf{x}})$ *is irreducible if and only if* $k = \text{size}(E)$.

We illustrate the reduction process by constructing in Figure 3 a maximal reduction sequence starting with $E(\mathsf{x})$ with E the program in (4-12).

Lemma 4.4. *For any extended program* $E(\vec{\mathsf{x}})$ *and irreducible* $F^1(\vec{\mathsf{x}}), F^2(\vec{\mathsf{x}})$,

$$\Big(E(\vec{\mathsf{x}}) \Rightarrow F^1(\vec{\mathsf{x}}) \ \&\ E(\vec{\mathsf{x}}) \Rightarrow F^2(\vec{\mathsf{x}})\Big) \Longrightarrow F^1 \equiv_c F^2.$$

PROOF is by induction on size(E).

Basis, size$(E) \le 1$. If size$(E) = 0$ so that $E(\vec{\mathsf{x}})$ is irreducible, then the hypothesis gives immediately $E \equiv_c F^1$ and $E \equiv_c F^2$, so $F^1 \equiv_c F^2$.

If size$(E) = 1$, then there is exactly one part E_i of E which can activate a one-step reduction. In the head case, the definition gives

$$\begin{aligned}\text{set}(F^1(\vec{\mathsf{x}})) = \Big(&\text{set}(E(\vec{\mathsf{x}})) \setminus \{\mathsf{c}(G_1, \dots, G_{j-1}, G_j, G_{j+1}, \dots, G_l)\}\Big)\\ &\cup \{\mathsf{c}(G_1, \dots, G_{j-1}, \mathsf{q}(\vec{\mathsf{x}}), G_{j+1}, \dots, G_l),\ \mathsf{q}(\vec{\mathsf{x}}) = G_j\}\end{aligned}$$

with a fresh function variable q and the same equation for set$(F^2(\vec{\mathsf{x}}))$ with a (possibly) different fresh function variable q'; but then $F^1 \equiv_c F^2$ by (1) of Lemma 4.1. The same argument works in the body case.

Induction Step, $k = \text{size}(E) \ge 2$. The hypothesis and Lemma 4.3 supply reduction sequences

$$\begin{aligned}(*)\quad & E(\vec{\mathsf{x}}) \Rightarrow_1 F^{1,1}(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^{1,k}(\vec{\mathsf{x}}) \equiv F^1(\vec{\mathsf{x}}),\\ & E(\vec{\mathsf{x}}) \Rightarrow_1 F^{2,1}(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^{2,k}(\vec{\mathsf{x}}) \equiv F^2(\vec{\mathsf{x}}),\end{aligned}$$

[10]See Problem x4.2 for a rigorous definition and proofs of the properties of size(E).

$$E(\mathsf{x}) : \begin{array}{l} \mathsf{p}(\mathsf{x},\phi_0(\mathsf{x}))\ \mathsf{where}\ \{ \\ \qquad \mathsf{p}(\mathsf{x},\mathsf{y}) = \text{if } \underline{\text{test}(\phi_0(\mathsf{x}),\mathsf{y})} \text{ then } \mathsf{y} \text{ else } \mathsf{p}(\mathsf{x},\sigma(\mathsf{x},\mathsf{y}))\} \end{array}$$

$$\xrightarrow{(\mathsf{p},1,\mathsf{q}_1)}$$

$$F^1(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\phi_0(\mathsf{x}))\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{p}(\mathsf{x},\underline{\sigma(\mathsf{x},\mathsf{y})}), \\ \mathsf{q}_1(\mathsf{x},\mathsf{y}) &= \text{test}(\phi_0(\mathsf{x}),\mathsf{y})\} \end{array}$$

$$\xrightarrow{(\mathsf{p},3,\mathsf{q}_2)}$$

$$F^2(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\phi_0(\mathsf{x}))\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{q}_2(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_2(\mathsf{x},\mathsf{y}) &= \mathsf{p}(\underline{\phi_0(\mathsf{x})},\sigma(\mathsf{x},\mathsf{y})), \mathsf{q}_1(\mathsf{x},\mathsf{y}) = \text{test}(\phi_0(\mathsf{x}),\mathsf{y})\} \end{array}$$

$$\xrightarrow{(\mathsf{q}_2,1,q_3)}$$

$$F^3(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\phi_0(\mathsf{x}))\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{q}_2(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_2(\mathsf{x},\mathsf{y}) &= \mathsf{p}(\mathsf{q}_3(\mathsf{x},\mathsf{y}),\sigma(\mathsf{x},\mathsf{y})), \mathsf{q}_3(\mathsf{x},\mathsf{y}) = \phi_0(\mathsf{x}), \\ \mathsf{q}_1(\mathsf{x},\mathsf{y}) &= \text{test}(\underline{\phi_0(\mathsf{x})},\mathsf{y})\} \end{array}$$

$$\xrightarrow{(\mathsf{q}_1,1,\mathsf{q}_4)}$$

$$F^4(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\phi_0(\mathsf{x}))\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{q}_2(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_2(\mathsf{x},\mathsf{y}) &= \mathsf{p}(\mathsf{q}_3(\mathsf{x},\mathsf{y}),\underline{\sigma(\mathsf{x},\mathsf{y})}), \\ \mathsf{q}_3(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x}), \\ \mathsf{q}_1(\mathsf{x},\mathsf{y}) &= \text{test}(\mathsf{q}_4(\mathsf{x},\mathsf{y}),\mathsf{y}), \\ \mathsf{q}_4(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x})\} \end{array}$$

$$\xrightarrow{(\mathsf{q}_2,2,\mathsf{q}_5)}$$

$$F^5(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\underline{\phi_0(\mathsf{x})})\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{q}_2(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_2(\mathsf{x},\mathsf{y}) &= \mathsf{p}(\mathsf{q}_3(\mathsf{x},\mathsf{y}),\mathsf{q}_5(\mathsf{x},\mathsf{y})), \\ \mathsf{q}_5(\mathsf{x},\mathsf{y}) &= \sigma(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_3(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x}),\ \mathsf{q}_1(\mathsf{x},\mathsf{y}) = \text{test}(\mathsf{q}_4(\mathsf{x},\mathsf{y}),\mathsf{y}), \\ \mathsf{q}_4(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x})\} \end{array}$$

$$\xrightarrow{(\mathsf{r},2,\mathsf{q}_6)}$$

$$F^6(\mathsf{x}) : \begin{array}{rl} \mathsf{p}(\mathsf{x},\mathsf{q}_6(x))\ \mathsf{where}\ \{ & \\ \mathsf{p}(\mathsf{x},\mathsf{y}) &= \text{if } \mathsf{q}_1(\mathsf{x},\mathsf{y}) \text{ then } \mathsf{y} \text{ else } \mathsf{q}_2(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_2(\mathsf{x},\mathsf{y}) &= \mathsf{p}(\mathsf{q}_3(\mathsf{x},\mathsf{y}),\mathsf{q}_5(\mathsf{x},\mathsf{y})), \\ \mathsf{q}_5(\mathsf{x},\mathsf{y}) &= \sigma(\mathsf{x},\mathsf{y}), \\ \mathsf{q}_3(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x}),\ \mathsf{q}_1(\mathsf{x},\mathsf{y}) = \text{test}(\mathsf{q}_4(\mathsf{x},\mathsf{y}),\mathsf{y},) \\ \mathsf{q}_4(\mathsf{x},\mathsf{y}) &= \phi_0(\mathsf{x}), \mathsf{q}_6(x) = \phi_0(\mathsf{x})\} \end{array}$$

DIAGRAM 3. A maximal reduction sequence for $E(\mathsf{x})$ in (4-12).

so focussing on the first one-step reductions in the two hypotheses, there are triples $(\mathsf{p}, j, \mathsf{q})$ and $(\mathsf{p}', j', \mathsf{q}')$ such that

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F^{1,1}(\vec{\mathsf{x}}) \text{ and } E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q}')} F^{2,1}(\vec{\mathsf{x}}). \tag{4-13}$$

We consider three cases on how this may arise:

Case 1: $(\mathsf{p} \not\sim_E \mathsf{p}'$ or $j \neq j')$ and $\mathsf{q} \not\equiv \mathsf{q}'$. The Amalgamation Lemma 4.2 applies in this case and supplies a program H such that

$$(**) \qquad F^{1,1}(\vec{\mathsf{x}}) \Rightarrow_1 H(\vec{\mathsf{x}}) \text{ and } F^{2,1}(\vec{\mathsf{x}}) \Rightarrow_1 H(\vec{\mathsf{x}}).$$

We fix a (maximal) reduction sequence

$$(***) \qquad H(\vec{\mathsf{x}}) \Rightarrow_1 H^3(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 H^k(\vec{\mathsf{x}}) \text{ (with an irreducible } H^k(\vec{\mathsf{x}}))$$

and notice that from the reductions in the three starred displays,

$$F^{1,1}(\vec{\mathsf{x}}) \Rightarrow F^1(\vec{\mathsf{x}}),\ F^{1,1}(\vec{\mathsf{x}}) \Rightarrow H^k(\vec{\mathsf{x}}), \qquad F^{2,1}(\vec{\mathsf{x}}) \Rightarrow F^2(\vec{\mathsf{x}}),\ F^{2,1}(\vec{\mathsf{x}}) \Rightarrow H^k(\vec{\mathsf{x}});$$

but $\mathrm{size}(F^{1,1}) = \mathrm{size}(F^{2,1}) = k-1$, so the Induction Hypothesis applies and yields the required $F^1 \equiv_c H^k \equiv_c F^2$.

Case 2, $(\mathsf{p} \not\sim_E \mathsf{p}'$ or $j \neq j')$ but $\mathsf{q} \equiv \mathsf{q}'$, so the first one-step reductions supplied by the Hypothesis are

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F^{1,1}(\vec{\mathsf{x}}) \text{ and } E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q})} F^{2,1}(\vec{\mathsf{x}}).$$

If q' is fresh but with the same arity and sort as q, then (2) of Lemma 4.1 gives

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p}',j',\mathsf{q}')} F^{2,1}\{\mathsf{q} :\equiv \mathsf{q}'\}(\vec{\mathsf{x}});$$

now (3) of Lemma 4.1 gives

$$F^{2,1}\{\mathsf{q} :\equiv \mathsf{q}'\}(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^{2,k}\{\mathsf{q} :\equiv \mathsf{q}'\}(\vec{\mathsf{x}})$$

so that Case 1 applies and gives

$$F^1 \equiv_c F^2\{\mathsf{q} :\equiv \mathsf{q}'\};$$

but $F^2\{\mathsf{q} :\equiv \mathsf{q}'\} \equiv_c F^2$, which completes the argument in this case.

Case 3, $\mathsf{p} \sim \mathsf{p}', j = j'$ and $\mathsf{q} \not\equiv \mathsf{q}'$. The first one-step reductions supplied by the Hypothesis now are

$$E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q})} F^{1,1}(\vec{\mathsf{x}}) \text{ and } E(\vec{\mathsf{x}}) \xrightarrow{(\mathsf{p},j,\mathsf{q}')} F^{2,1}(\vec{\mathsf{x}})$$

where q and q' are distinct but have the same sort and arity; so

$$F^{1,1}\{\mathsf{q} :\equiv \mathsf{r}\} \equiv_c F^{2,1}\{\mathsf{q}' :\equiv \mathsf{r}\}$$

with any fresh r; and then by the Induction Hypothesis and (3) of Lemma 4.1 as above,

$$F^{1,k} \equiv_c F^{1,k}\{\mathsf{q} :\equiv \mathsf{r}\} \equiv_c F^{2,k}\{\mathsf{q}' :\equiv \mathsf{r}\} \equiv_c F^2,$$

which is what we needed to prove. $\dashv$

Theorem 4.5 (Canonical forms). *Every extended program $E(\vec{\mathsf{x}})$ is reducible to a unique up to congruence irreducible extended program $\mathrm{cf}(E(\vec{\mathsf{x}}))$, its canonical form.*

In detail: with every extended program $E(\vec{\mathsf{x}})$, we can associate an extended program $\mathrm{cf}(E(\vec{\mathsf{x}}))$ with the following properties:

(1) $\mathrm{cf}(E(\vec{\mathsf{x}}))$ *is irreducible.*
(2) $E(\vec{\mathsf{x}}) \Rightarrow \mathrm{cf}(E(\vec{\mathsf{x}}))$.
(3) *If $F(\vec{\mathsf{x}})$ is irreducible and $E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}})$, then $F(\vec{\mathsf{x}}) \equiv_c \mathrm{cf}(E(\vec{\mathsf{x}}))$.*

It follows that for all $E(\vec{\mathsf{x}}), F(\vec{\mathsf{x}})$,

$$E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}}) \Longrightarrow \mathrm{cf}(E(\vec{\mathsf{x}})) \equiv_c \mathrm{cf}(F(\vec{\mathsf{x}})).$$

Proof. If $E(\vec{\mathsf{x}})$ is irreducible, take $\mathrm{cf}(E(\vec{\mathsf{x}})) \equiv E(\vec{\mathsf{x}})$, and if $\mathrm{size}(E) = k > 0$, fix a maximal sequence of one-step reductions

$$E(\vec{\mathsf{x}}) \Rightarrow_1 F^1(\vec{\mathsf{x}}) \Rightarrow_1 \cdots \Rightarrow_1 F^k(\vec{\mathsf{x}})$$

and set $\mathrm{cf}(E(\vec{\mathsf{x}})) :\equiv F^k(\vec{\mathsf{x}})$; now (1) and (2) are immediate and (3) follows from Lemma 4.4. The last claim holds because $F(\vec{\mathsf{x}}) \Rightarrow \mathrm{cf}(F(\vec{\mathsf{x}}))$, so

$$\text{if } E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}}) \text{ then } E(\vec{\mathsf{x}}) \Rightarrow \mathrm{cf}(F(\vec{\mathsf{x}}))$$

and hence $\mathrm{cf}(E(\vec{\mathsf{x}})) \equiv_c \mathrm{cf}(F(\vec{\mathsf{x}}))$ by (3). ⊣

It is clearly possible to assign to each extended program $E(\vec{\mathsf{x}})$ a specific canonical form, e.g., by choosing $\mathrm{cf}(E(\vec{\mathsf{x}}))$ to be the "lexicographically least" irreducible $F(\vec{\mathsf{x}})$ such that $E(\vec{\mathsf{x}}) \Rightarrow F(\vec{\mathsf{x}})$. This, however, is neither convenient nor useful, and it is best to think of $\mathrm{cf}(E(\vec{\mathsf{x}}))$ as denoting ambiguously *any irreducible extended program* such that $E(\vec{\mathsf{x}}) \Rightarrow \mathrm{cf}(E(\vec{\mathsf{x}}))$; the theorem insures that any two such *canonical forms* of $E(\vec{\mathsf{x}})$ are congruent, and the construction in the proof gives a simple way to compute one of them.

Canonical forms for richer languages. A substantially more general version of the Canonical Form Theorem for *functional structures* was established in Moschovakis [1989a], and there are natural versions of it for many richer languages, including a suitable formulation of the typed λ-calculus with recursion (PCF). The proof we gave here for McCarthy programs is considerably easier than these, more general results.

Intensions and elementary algorithms. The (referential) *intension* of an extended McCarthy program $E(\vec{\mathsf{x}})$ in a structure $\mathbf{A}$ is the recursor of its canonical form,

$$\mathrm{int}(\mathbf{A}, E(\vec{\mathsf{x}})) =_{\mathrm{df}} \mathfrak{r}(\mathbf{A}, \mathrm{cf}(E)(\vec{\mathsf{x}})); \tag{4-14}$$

it does not depend on the choice of $\mathrm{cf}(E)$ by (3-12), and it models the *elementary algorithm* expressed by $E(\vec{\mathsf{x}})$ in $\mathbf{A}$.

The recursor representation of an iterative algorithm. Using the precise definition of the classical notion of an *iterator* (sequential machine)

$$\mathfrak{i} = (\mathrm{input}, S, \sigma, T, \mathrm{output}) : X \rightsquigarrow W$$

in Section 2C of ARIC, let

$$\mathbf{A}_\mathfrak{i} = (A_\mathfrak{i}, X, W, S, \mathrm{input}, \sigma, T, \mathrm{output})$$

be the structure associated with $\mathfrak{i}$, where

$$A_\mathfrak{i} = X \uplus W \uplus S$$

is the disjoint sum of the sets X, W and S, and let

$$E_\mathfrak{i}(\mathsf{x}) :\equiv \mathsf{q}(\mathrm{input}(\mathsf{x})) \text{ where } \{\mathsf{q}(\mathsf{s}) = \text{if } T(\mathsf{s}) \text{ then output}(\mathsf{s}) \text{ else } \mathsf{q}(\sigma(\mathsf{s}))\}.$$

Proposition 4.6. *An iterator* i *is determined by its intension*

$$\text{int}(\mathsf{i}) = \text{int}(\mathbf{A}_\mathsf{i}, E_\mathsf{i}(\mathsf{x})) = \mathfrak{r}(\mathbf{A}_\mathsf{i}, \text{cf}(E_\mathsf{i})(\mathsf{x})), \tag{4-15}$$

the recursive algorithm on $\mathbf{A}_\mathsf{i}$ *expressed by the extended tail recursive program* $E_\mathsf{i}(\vec{\mathsf{x}})$ *associated with* i.

PROOF is easy, if a bit technical, and we leave it for Problem x4.4*. ⊣

Theorem 4.2 in Moschovakis and Paschalis [2008] is a much stronger result along these lines, but it requires several definitions for its formulation and this simple Proposition expresses quite clearly the basic message: every property of an iterator i can be expressed as a property of its associated recursor int(i), and so choosing recursive rather than iterative algorithms as the basic objects does not force us to miss any important facts about the simpler objects.

Problems for Section 4.

Problem x4.1. A term I is *immediate of sort* s if $I \equiv \mathsf{v}_i$ and $s = \texttt{ind}$; or $I \equiv \mathsf{p}^{s,0}$; or $I \equiv \mathsf{p}^{s,n}(\mathsf{u}_1, \dots, \mathsf{u}_n)$ where each u_i is an individual variable. Write out a similar, full (recursive) definition of *irreducible terms of sort* s.

Problem x4.2 (Size). (1) Prove than there is exactly one function size(E) which assigns a number to every explicit term so that

if $E(\vec{\mathsf{x}})$ *is irreducible, then* size(E) = 0, *and*

$$\text{size}(\mathsf{c}(G_1, \dots, G_l)) = \sum \big\{\text{size}(G_i) + 1 \mid G_i \text{ is not immediate}\big\}.$$

(2) For a program E as in (4-1), set

$$\text{size}(E) = \sum \big\{\text{size}(E)_i \mid E_i \text{ is a part of } E\big\}$$

and check that

(1) $E(\vec{\mathsf{x}})$ is irreducible if and only if size(E) = 0, and
(2) if $E(\vec{\mathsf{x}}) \Rightarrow_1 F(\vec{\mathsf{x}})$, then size($F$) = size($E$) − 1.

Problem x4.3. Notice that the irreducible extended program $F_5(\vec{\mathsf{x}})$ in Diagram 3 to which $E(\vec{\mathsf{x}})$ reduces calls for computing the value $\phi_1(x)$ twice in each "loop". Define an extended program $F(\vec{\mathsf{x}})$ which computes the same partial function as $E(\vec{\mathsf{x}})$ but calls for computing $\phi_1(x)$ only once in each loop, and construct a complete reduction sequence for $E^*(\vec{\mathsf{x}})$.

Problem x4.4*. Prove Proposition 4.6. HINT: Use the reduction calculus to compute a canonical form of the program $E_\mathsf{i}(\vec{\mathsf{x}})$,

$$\begin{aligned}\text{cf}(E_\mathsf{i}(\vec{\mathsf{x}})) \equiv_c \mathsf{q}(\mathsf{p}_1(\mathsf{x}))\ \textit{where}\ \{&\mathsf{p}_1(\mathsf{x}) = \text{input}(\mathsf{x}),\\ &\mathsf{p}_2(\mathsf{s}) = T(\mathsf{s}),\ \mathsf{p}_3(\mathsf{s}) = \text{output}(\mathsf{s}),\ \mathsf{p}_4(\mathsf{s}) = \sigma(\mathsf{s}),\ \mathsf{p}_5(\mathsf{s}) = \mathsf{q}(\mathsf{p}_4(\mathsf{s}))\\ &\mathsf{q}(\mathsf{s}) = \text{if } \mathsf{p}_2(\mathsf{s}) \text{ then } \mathsf{p}_3(\mathsf{s}) \text{ else } \mathsf{p}_5(\mathsf{s})\},\end{aligned}$$

so that $\text{int}(\mathsf{i}(\mathsf{x})) = \text{int}(\mathbf{A}_\mathsf{i}, E_\mathsf{i}(\mathsf{x})) = \mathfrak{r}(\mathbf{A}_\mathsf{i}, \text{cf}(E_\mathsf{i}(\mathsf{x}))) = (\alpha_0, \alpha_1, \dots, \alpha_6)$, where

$$\alpha_i : A_\mathsf{i} \times (A_\mathsf{i} \rightharpoonup A_\mathsf{i}) \times (A_\mathsf{i} \rightharpoonup \mathbb{B}) \times (A_\mathsf{i} \rightharpoonup A_\mathsf{i})^4 \rightharpoonup A_\mathsf{i} \quad (i = 1, \dots, 6).$$

The result is practically trivial from this, except for the fact that we only know the canonical form up to congruence, so it is not immediate how to "pick out" from the functionals α_i the input, output, etc. of $\mathfrak{i}$—and this is complicated by the fact that the sets X, W and S may overlap, in fact they may all be the same set. We need to use the details of the coding of many-sorted structures by single-sorted ones specified in Section 1D of ARIC.

§5. Decidability of intensional program equivalence. Two extended Φ-programs are *intensionally equivalent on a Φ-structure* $\mathbf{A}$ if they have equal intensions,

$$E(\vec{\mathsf{x}}) =^{\mathbf{A}}_{\text{int}} F(\vec{\mathsf{x}}) \iff_{\text{df}} \text{int}(\mathbf{A}, E(\vec{\mathsf{x}})) = \text{int}(\mathbf{A}, F(\vec{\mathsf{x}})); \tag{5-1}$$

they are (globally) *intensionally equivalent* if they are intensionally equivalent on every infinite $\mathbf{A}$,

$$E(\vec{\mathsf{x}}) =_{\text{int}} F(\vec{\mathsf{x}}) \iff_{\text{df}} (\forall \text{ infinite } \mathbf{A})[\text{int}(\mathbf{A}, E(\vec{\mathsf{x}})) = \text{int}(\mathbf{A}, F(\vec{\mathsf{x}}))]. \tag{5-2}$$

The main result of this section is

Theorem 5.1. *For every infinite Φ-structure $\mathbf{A}$, the relation of intensional equivalence on $\mathbf{A}$ between extended Φ-programs is decidable.*[11]

We will also prove that global intensional equivalence between Φ-programs is essentially equivalent with congruence, and so it is decidable, Theorem 5.14.

Plan for the proof. To decide if $E(\vec{\mathsf{x}})$ and $F(\vec{\mathsf{x}})$ are intensionally equivalent on $\mathbf{A}$, we compute their canonical forms, check that these have the same number $K+1$ of irreducible parts E_i, F_j and then (to apply Lemma 3.1) check if there is a permutation π of $\{0, \dots, K\}$ with inverse ρ such that $\pi(0) = 0$ and

$$\begin{aligned}
\mathbf{A} &\models E_0(\vec{\mathsf{x}}) &&= F_0\{\vec{\mathsf{q}} :\equiv \rho(\vec{\mathsf{p}})\}(\vec{\mathsf{x}}),\\
\mathbf{A} &\models E_i(\vec{\mathsf{x}}_i) &&= F_{\pi(i)}\{\vec{\mathsf{q}} :\equiv \rho(\vec{\mathsf{p}}), \vec{\mathsf{y}}_{i\pi(i)} :\equiv \vec{\mathsf{x}}_i\}, \quad (i = 1, \dots, K).
\end{aligned}$$

These identities are all irreducible by (4-3), so the problem comes down to deciding the validity of *irreducible term identities* in $\mathbf{A}$. The proof involves associating with each infinite Φ-structure $\mathbf{A}$ a finite list of "conditions" about its (finitely many) primitives—its *dictionary*—which codifies all the information needed to decide whether an arbitrary irreducible identity holds in $\mathbf{A}$; it involves some fussy details, but fundamentally it is an elementary exercise in the *equational logic of partial terms with conditionals.* In classical terminology, it is (basically) a *Finite Basis Theorem* for the theory of irreducible identities which hold in an arbitrary, infinite (partial) Φ-structure.[12]

[11]Except for a reference to a published result in Problem x5.15*, I will not discuss in this paper the question of *complexity* of intensional equivalence on recursive programs, primarily because I do not know anything non-trivial about it.

[12]The decidability of intensional equivalence is considerably simpler for total structures $\mathbf{A}$, cf. Lemma 5.9 and substantially more difficult for the FLR-structures of Moschovakis [1989a] which allow *monotone, discontinuous functionals* among their primitives; see Moschovakis [1994]—and a correction which fills a gap in the proof of this result in Moschovakis [1994] which is posted (along with the paper) on ynm's homepage. It is an open—and, I think, interesting—question for the *acyclic recursive algorithms* of Moschovakis [2006], where it models *synonymy* (or *faithful translation*) for fragments of natural language.

The satisfaction relation. We will also need in this Section the classical characterization of valid identities in terms of the *satisfaction relation.*

An *assignment* σ in a Φ-structure $\mathbf{A}$ associates with each variable an object of the proper kind, i.e.,

$$\sigma(\mathsf{v}_i) \in A, \quad \sigma(\mathsf{p}_i^{s,n}) : A^n \rightharpoonup A_s,$$

so that, for example, the *boolean function variables* $\mathsf{p}_i^{\mathtt{boole},0}$ are assigned nullary partial functions $p : \mathbf{I} \rightharpoonup \mathbb{B}$, essentially $\uparrow$, tt or ff. For each Φ^2-term E, we define

$$\sigma(\mathbf{A}, E) =_{\mathrm{df}} \text{ the value (perhaps } \uparrow) \text{ of } E \text{ in } \mathbf{A} \text{ under } \sigma$$

by the usual compositional recursion and we set

$$\mathbf{A}, \sigma \models E = F \iff_{\mathrm{df}} \sigma(\mathbf{A}, E) = \sigma(\mathbf{A}, F).$$

If the variables of E are among $\mathsf{p}_1, \dots, \mathsf{p}_K, \mathsf{x}_1, \dots, \mathsf{x}_n$, then

$$\sigma(\mathbf{A}, E) = \mathrm{den}((\mathbf{A}, \sigma(\mathsf{p}_1), \dots, \sigma(\mathsf{p}_K)), \sigma(\mathsf{x}_1), \dots, \sigma(\mathsf{x}_n)),$$

and so by the notation on p. 7, for any two extended Φ^2-terms,

$$\mathbf{A} \models E(\vec{\mathsf{x}}) = F(\vec{\mathsf{x}}) \iff (\text{for all } \sigma)[\mathbf{A}, \sigma \models E = F].$$

Preliminary constructions and notation. We fix for this Section an infinite Φ-structure $\mathbf{A} = (A, \Phi)$ (with finite Φ, as always) and we assume without loss of generality that among the symbols in Φ are $\psi_{\mathrm{tt}}, \psi_{\mathrm{ff}}, \mathrm{id}$ such that

$$\psi_{\mathrm{tt}}^{\mathbf{A}} = \mathrm{tt}, \quad \psi_{\mathrm{ff}}^{\mathbf{A}} = \mathrm{ff}, \quad \mathrm{id}^{\mathbf{A}}(t) = t.$$

We will often define assignments partially, only on the variables which occur in a term E and specifying only finitely many values $\sigma(\mathsf{p}_i^{s,n})(\vec{x})$ of the function variables—those needed to compute $\sigma(\mathbf{A}, E)$—perhaps to come back later and extend σ when we view E as a subterm of some F.

Using the assumption that A is infinite, we fix pairwise disjoint, infinite sets

$$A^{\mathtt{ind}}, \ A_i^{\mathtt{ind},n} \subset A$$

whose union is co-infinite in A and we fix an assignment $\ddot{\sigma}$ on the individual and function variables of sort `ind` using injections

$$\ddot{\sigma} : \{\mathsf{v}_0, \mathsf{v}_1, \dots\} \rightarrowtail A^{\mathtt{ind}}, \quad \ddot{\sigma}(\mathsf{p}_i^{\mathtt{ind},n}) : A^n \rightarrowtail A_i^{\mathtt{ind},n}.$$

We do not define at this point any values $\ddot{\sigma}(\mathsf{p}_i^{\mathtt{boole},n})(x_1, \dots, x_n)$ for function variables of sort `boole`.

A term E is **pure, algebraic** if it is of sort `ind` and none of tt, ff, the conditional or any symbol from Φ occurs in it, i.e.,

(Pure, algebraic terms) $\qquad E :\equiv \mathsf{v}_i \mid \mathsf{p}_i^{\mathtt{ind},0} \mid \mathsf{p}_i^{\mathtt{ind},n}(E_1, \dots, E_n).$

Lemma 5.2. *If E, F are pure, algebraic terms, then*

$$\mathbf{A}, \ddot{\sigma} \models E = F \iff E \equiv F.$$

Proof of this (classical) result is by an easy induction on length(E), using the Parsing Lemma in Problem x2.1. ⊣

Forms of irreducible identities. To organize the proof of Theorem 5.1, we will appeal to the (trivial) fact that *every irreducible term is in exactly one of the following forms*:

(1) tt, ff, or an individual variable v.
(2) Function variable application, $\mathsf{p}(z_1, \dots, z_n)$, where the terms $z_1, \dots, z_n$ are immediate of sort `ind` and $n \geq 0$; p can be of either sort, `ind` or `boole`.
(3) Conditional, if z_1 then z_2 else z_3, with immediate z_1 of sort `boole` and immediate z_2, z_3 of the same sort, `ind` or `boole`.
(4) Primitive application, $\phi(z_1, \dots, z_k)$, $k \geq 0$, with immediate $z_1, \dots, z_k$, all of sort `ind` and ϕ of either sort.

It follows that every irreducible identity falls in one of the ten, pairwise exclusive forms [i-j] with $1 \leq i \leq j \leq 4$ depending on the forms of its two sides. In dealing with these cases in the following Lemmas, we will use repeatedly the (trivial) fact that

> *if u, v are immediate of sort* `boole` *and $u \not\equiv v$, then we can extend $\ddot{\sigma}$ to the function variables which occur in u and v so that $\ddot{\sigma}(u)$ and $\ddot{\sigma}(v)$ take any pre-assigned truth values* (or diverge).

Lemma 5.3 ([2-2]). *For any two irreducible terms in form* (2),

$$\mathbf{A} \models \mathsf{p}(z_1, \dots, z_n) = \mathsf{q}(w_1, \dots, w_m) \iff \mathsf{p}(z_1, \dots, z_n) \equiv \mathsf{q}(w_1, \dots, w_m).$$

Proof. Let $E \equiv \mathsf{p}(z_1, \dots, z_n)$ and $F \equiv \mathsf{q}(w_1, \dots, w_m)$, assume $\mathbf{A} \models E = F$ and consider two cases:

Case 1, sort(p) = sort(q) = `ind`; now E and F are pure algebraic and so by Lemma 5.2:

$$\mathbf{A} \models E = F \Longrightarrow \mathbf{A}, \ddot{\sigma} \models E = F \Longrightarrow E \equiv F.$$

Case 2, sort(p) = sort(q) = `boole`. If $\mathsf{p} \not\equiv \mathsf{q}$, extend $\ddot{\sigma}$ by setting

$$\ddot{\sigma}(\mathsf{p})(x_1, \dots, x_n) = \mathrm{tt}, \quad \ddot{\sigma}(\mathsf{q})(y_1, \dots, y_m) = \mathrm{ff}$$

and check that $\mathbf{A}, \ddot{\sigma} \not\models E = F$; so $\mathsf{p} \equiv \mathsf{q}$, hence $n = m$, and it is enough to prove that $z_1 \equiv w_1, \dots, z_n \equiv w_n$. If not, then for some i, $\ddot{\sigma}(z_i) \neq \ddot{\sigma}(w_i)$ by Lemma 5.2; and then we can choose some $a \in A$ and extend $\ddot{\sigma}$ by setting

$$\ddot{\sigma}(\mathsf{p})(x_1, \dots, x_n) = \text{if } (x_i = a) \text{ then tt else ff},$$

so that $\ddot{\sigma}(\mathsf{p}(z_1, \dots, z_n)) \neq \ddot{\sigma}(\mathsf{p}(w_1, \dots, w_n))$, which contradicts the hypothesis.$\dashv$

Lemma 5.4 ([3-3]). *If $z_1, \dots, w_3$ are all immediate, then*

$$\begin{aligned} \mathbf{A} \models \text{if } z_1 \text{ then } z_2 \text{ else } z_3 = \text{if } w_1 \text{ then } w_2 \text{ else } w_3, \\ \iff z_1 \equiv w_1, z_2 \equiv w_2, z_3 \equiv w_3. \end{aligned}$$

Proof. Assume $\mathbf{A} \models$ if z_1 then z_2 else $z_3 =$ if w_1 then w_2 else w_3, note that z_1, w_1 must be of sort `boole` and consider cases:

Case (a): z_2, z_3, w_2, w_3 are of sort `ind`. If $z_1 \not\equiv w_1$, extend $\ddot{\sigma}$ so that

$$\ddot{\sigma}(z_1) = \mathrm{tt}, \quad \ddot{\sigma}(w_1)\uparrow$$

which gives $\ddot{\sigma}(\text{if } z_1 \text{ then } z_2 \text{ else } z_3) = \ddot{\sigma}(z_2)$ while $\ddot{\sigma}(\text{if } w_1 \text{ then } w_2 \text{ else } w_3)\uparrow$, hence $z_1 \equiv w_1$. Setting $\ddot{\sigma}(z_1) = \ddot{\sigma}(w_1) = \mathrm{tt}$ gives $z_2 \equiv w_2$ and setting $\ddot{\sigma}(z_1) = \mathrm{ff}$ gives $z_3 \equiv w_3$.

Case (b): all $z_1, z_2, z_3, w_1, w_2, w_3$ are of sort `boole`. Assume first, towards a contradiction that $z_1 \not\equiv w_1$, and set

$$\ddot{\sigma}(z_1) = \mathrm{tt}, \quad \ddot{\sigma}(w_1)\uparrow .$$

If $z_2 \not\equiv w_1$, we can further set $\ddot{\sigma}(z_2) = \mathrm{tt}$ (even if $z_2 \equiv z_1$) and get a contradiction, so $z_2 \equiv w_1$. The same argument, with $\ddot{\sigma}(z_1) = \mathrm{ff}$ this time gives $z_3 \equiv w_1$; and the symmetric argument gives $w_2 \equiv w_3 \equiv z_1$, so what we have now is

$$\mathbf{A} \models \text{if } z_1 \text{ then } w_1 \text{ else } w_1 = \text{if } w_1 \text{ then } z_1 \text{ else } z_1$$

with the added hypothesis that $z_1 \not\equiv w_1$; but then we can set

$$\ddot{\sigma}(z_1) = \mathrm{tt}, \quad \ddot{\sigma}(w_1) = \mathrm{ff},$$

which gives

$$\ddot{\sigma}(\text{if } z_1 \text{ then } w_1 \text{ else } w_1) = \ddot{\sigma}(w_1) = \mathrm{ff}, \quad \ddot{\sigma}(\text{if } w_1 \text{ then } z_1 \text{ else } z_1) = \ddot{\sigma}(z_1) = \mathrm{tt},$$

which violates the hypothesis; so $z_1 \equiv w_1$, and the hypothesis in Case (b) takes the form

$$\mathbf{A} \models \text{if } z_1 \text{ then } z_2 \text{ else } z_3 = \text{if } z_1 \text{ then } w_2 \text{ else } w_3.$$

Now assume towards a contradiction that $z_2 \not\equiv w_2$; so one of them is different from z_1, suppose it is w_2; so we now get a contradiction by setting $\ddot{\sigma}(z_1) = \ddot{\sigma}(z_2) = \mathrm{tt}$ and $\ddot{\sigma}(w_2) = \mathrm{ff}$.

At this point we know that $z_1 \equiv w_1$ and $z_2 \equiv w_2$, and the last bit that $z_3 \equiv w_3$ is simpler. ⊣

Lemma 5.5 ([2-3])**.** *If $z_1, \dots, z_m, w_1, w_2, w_3$ are all immediate, then*

$$\mathbf{A} \models \mathsf{p}(z_1, \dots, z_m) = \text{if } w_1 \text{ then } w_2 \text{ else } w_3 \iff \mathsf{p}(z_1, \dots, z_m) \equiv w_1 \equiv w_2 \equiv w_3.$$

Proof. Notice first that if p is of sort `ind`, then

$$\mathbf{A}, \sigma \not\models \mathsf{p}(z_1, \dots, z_m) = \text{if } w_1 \text{ then } w_2 \text{ else } w_3$$

for any σ which converges on $\mathsf{p}(z_1, \dots, z_m)$ but such that $\sigma(w_1)\uparrow$.

If p is of sort `boole` and $\mathsf{p}(z_1, \dots, z_m) \not\equiv w_1$, then we can get a σ such that $\sigma(\mathsf{p}(z_1, \dots, z_m))\downarrow$ but $\sigma(w_1)\uparrow$ so the hypothesis fails again. It follows that $\mathsf{p}(z_1, \dots, z_m) \equiv w_1$ and the hypothesis takes the form

$$\mathbf{A} \models w_1 = \text{if } w_1 \text{ then } w_2 \text{ else } w_3.$$

If $w_2 \not\equiv w_1$, we get a contradiction by setting $\sigma(w_1) = \mathrm{tt}, \sigma(w_2)\uparrow$, and similarly if $w_1 \not\equiv w_3$. ⊣

Lemma 5.6 ([3-4])**.** *If $z_1, \dots, z_m, w_1, w_2, w_3$ are all immediate, then*

$$\mathbf{A} \not\models \phi(z_1, \dots, z_m) = \text{if } w_1 \text{ then } w_2 \text{ else } w_3.$$

Proof. Notice that $w_1 \not\equiv z_i$, since $\mathrm{sort}(w_1) = \texttt{boole}$ while $\mathrm{sort}(z_i) = \texttt{ind}$ and consider three cases:

If $\phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_m))\downarrow$, extend $\ddot{\sigma}$ by setting $\ddot{\sigma}(w_1)\uparrow$ so

$$(*) \qquad \mathbf{A}, \ddot{\sigma} \not\models \phi(z_1, \dots, z_m) = \text{if } w_1 \text{ then } w_2 \text{ else } w_3.$$

If $\phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_m))\uparrow$ and ϕ is of sort $\texttt{ind}$, then setting now $\ddot{\sigma}(w_1) = \mathrm{tt}$ gives $(*)$ again, since $\ddot{\sigma}(w_2)\downarrow$.

If $\phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_m))\uparrow$ and ϕ is of sort $\texttt{boole}$, then we can set $\ddot{\sigma}(w_1) = \ddot{\sigma}(w_2) = \mathrm{tt}$ which again gives $(*)$. ⊣

Lemma 5.7. *If $z_1, \dots, z_k, w_1, \dots, w_m$ are all immediate, then*

$$\textit{if } \mathbf{A} \models \phi(z_1, \dots, z_k) = \mathsf{p}(w_1, \dots, w_m),$$
$$\textit{then } \mathsf{p}(w_1, \dots, w_m) \equiv z_j \textit{ for some } j$$
$$\textit{and so } \mathbf{A} \models \phi(z_1, \dots, z_k) = z_j = \mathrm{id}(z_j).$$

In particular, with $k > 0$ and $m = 0$,

$$\mathbf{A} \models \phi(z_1, \dots, z_k) = \mathsf{p} \Longrightarrow \mathsf{p} \equiv z_j \textit{ for some } j,$$

and with $k = 0$, $\mathbf{A} \not\models \phi = \mathsf{p}(z_1, \dots, z_m)$.

Proof. Assume the hypothesis $\mathbf{A} \models \phi(z_1, \dots, z_k) = \mathsf{p}(w_1, \dots, w_m)$ and also that $\mathsf{p}(w_1, \dots, w_m) \not\equiv z_j$ for any j.

(a) $\mathrm{sort}(\phi) = \texttt{ind}$; because if $\mathrm{sort}(\phi) = \texttt{boole}$, then

$$\text{either } \phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_k)))\downarrow \text{ and we can set } \ddot{\sigma}(\mathsf{p})(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_k))\uparrow$$
$$\text{or } \phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_k)))\uparrow \text{ and we can set } \ddot{\sigma}(\mathsf{p})(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_k))\downarrow .$$

$(*)$ Call E *relevant* if $E \equiv z_i$ or $E \equiv w_i$ for some i, set $\sigma(\mathsf{x}) = \ddot{\sigma}(\mathsf{x})$ for every individual variable which occurs in a relevant term, and for a function variable $\mathsf{q}^{\texttt{ind},n}$ of sort $\texttt{ind}$ and arity n, set

$$\sigma(\mathsf{q}^{\texttt{ind},n})(x_1, \dots, x_n) = v \iff_{\mathrm{df}} \text{there is a relevant } F \equiv \mathsf{q}^{\texttt{ind},n}(\mathsf{x}_1, \dots, \mathsf{x}_n)$$
$$\text{such that } x_1 = \ddot{\sigma}(\mathsf{x}_1), \dots, x_n = \ddot{\sigma}(\mathsf{x}_n), \ \& \ \ddot{\sigma}(\mathsf{q}^{\texttt{ind},n})(x_1, \dots, x_n) = v.$$

(b) *For each pure, algebraic term E, $\sigma(E)$ is defined exactly when E is relevant and then $\sigma(E) = \ddot{\sigma}(E)$.* This is because by the definition, if $\sigma(E)$ is defined, then $\sigma(E) = \ddot{\sigma}(E) = \ddot{\sigma}(F)$ for a relevant F and hence $E \equiv F$ by Lemma 5.2.

It follows that $\sigma(\mathsf{p}(w_1, \dots, w_m))$ is not defined by the stipulation $(*)$, since $\mathsf{p}(w_1, \dots, w_m) \not\equiv z_i$ by the hypothesis and each w_j is a proper subterm of $\mathsf{p}(w_1, \dots, w_m)$, so it cannot be equal to it; and then we reach a contradiction by setting $\sigma(\mathsf{p})$ so that

$$\sigma(\mathsf{p})(\ddot{\sigma}(w_1), \dots, \ddot{\sigma}(w_m)) \neq \phi^{\mathbf{A}}(\ddot{\sigma}(z_1), \dots, \ddot{\sigma}(z_k)). \qquad ⊣$$

Lemma 5.8. *If there is an algorithm which decides the validity in $\mathbf{A}$ of identities in form* [4-4].

$$\mathbf{A} \models \phi(z_1, \dots, z_n) = \psi(z_{n+1}, \dots, z_l) \quad (\phi, \psi \in \Phi, z_1, \dots, z_l \text{ immediate}),$$

then the relation of intensional equivalence on $\mathbf{A}$ between extended Φ-programs is decidable.

PROOF. Forms [1-1], [1-2] and [1-3] are trivial, cf. Problem x5.1.

Form [2-2] is decided by Lemma 5.3 and forms [2-3], [3-3] and [3-4] are decided by Lemmas 5.5, 5.4 and 5.6.

Identities in forms [4-1] (equivalent to [1-4]) and [4-2] (equivalent to [2-4]) are either

$$\phi(z_1,\dots,z_n) = \mathrm{tt} = \psi_{\mathrm{tt}} \text{ and } \phi(z_1,\dots,z_n) = \mathrm{ff} = \psi_{\mathrm{ff}}$$

which are in form [4-4] with our assumption that $\psi_{\mathrm{tt}}, \psi_{\mathrm{ff}} \in \Phi$, or by Lemma 5.7 and our assumption that $\mathrm{id} \in \Phi$, they are equivalent in $\mathbf{A}$ to identities

$$\phi(z_1,\dots,z_n) = z_j = \mathrm{id}(z_j)$$

for some j (that we can compute), cf. Problem x5.10. ⊣

At this point there is a fork in the road: it is very easy to finish the proof of Theorem 5.1 for total structures, while the general case of arbitrary partial structures involves some technical difficulties. We do the simple thing first.

Lemma 5.9. *Suppose* $\mathbf{A}$ *is a total, infinite,* Φ*-structure,* $\phi, \psi \in \Phi$ *and* $z_1,\dots,z_l$ *are immediate terms.*

(1) *If* $\mathbf{A} \models \phi(z_1,\dots,z_n) = \psi(z_{n+1},\dots,z_l)$*, then for all* $i = 1,\dots,n$*,*

if z_i *is not an individual variable, then there is a* j *such that* $z_i \equiv z_{n+j}$*.*

(2) *There is a sequence* $\mathsf{x}_1,\dots,\mathsf{x}_l$ *of* (not necessarily distinct) *individual variables such that*

$$(\forall i,j)\Big(\mathsf{x}_i \equiv \mathsf{x}_j \iff z_i \equiv z_j\Big),$$

and for any such sequence

$$\text{(5-3)}\quad \mathbf{A} \models \phi(z_1,\dots,z_n) = \psi(z_{n+1},\dots,z_l) \iff \mathbf{A} \models \phi(\mathsf{x}_1,\dots,\mathsf{x}_n) = \psi(\mathsf{x}_{n+1},\dots,\mathsf{x}_l).$$

PROOF. (1) Suppose towards a contradiction that

$$\mathbf{A} \models \phi(z_1,\dots,z_n) = \psi(z_{n+1},\dots,z_l)$$

but $z_i \equiv \mathsf{p}(\mathsf{x}_1,\dots,\mathsf{x}_k)$ is not identical with any z_{n+j} and define the assignment σ which agrees with $\ddot\sigma$ on all individual and function variables except that

$$\sigma(\mathsf{p})(\ddot\sigma(\mathsf{x}_1),\dots,\ddot\sigma(\mathsf{x}_k))\uparrow;$$

this gives $\phi^{\mathbf{A}}(\sigma(z_1),\dots,\sigma(z_n))\uparrow$ and $\phi^{\mathbf{A}}(\sigma_{n+1},\dots\sigma(z_l))\downarrow$, which contradicts the hypothesis.

(2) To construct a sequence $\mathsf{x}_1,\dots,\mathsf{x}_l$ of individual variables with the required property by induction on i: pick a fresh x_i if $z_i \not\equiv z_j$ for every $j < i$ and otherwise let $\mathsf{x}_i :\equiv \mathsf{x}_j$ for the least (and hence every) $j < i$ such that $z_i \equiv z_j$.

Suppose $\mathsf{x}_1,\dots,\mathsf{x}_l$ are such that $\mathsf{x}_i \equiv \mathsf{x}_j \iff z_i \equiv z_j$, assume that

$$\mathbf{A} \models \phi(z_1,\dots,z_n) = \psi(z_{n+1},\dots,z_l)$$

and for any assignment σ define τ so that

$$(\forall i = 1,\dots,k)[\tau(z_i) = \sigma(\mathsf{x}_i)],$$

which is possible by the assumption relating x_i and z_i. The hypothesis gives

$$\phi^{\mathbf{A}}(\tau(z_1),\dots,\tau(z_n)) = \psi^{\mathbf{A}}(\tau(z_{n+1}),\dots,\tau(z_l)),$$

which then implies

$$\phi^{\mathbf{A}}(\sigma(\mathsf{x}_1),\dots,\sigma(\mathsf{x}_n)) = \psi^{\mathbf{A}}(\sigma(\mathsf{x}_{n+1}),\dots,\sigma(\mathsf{x}_l)),$$

and since σ was arbitrary we get the required

$$\mathbf{A} \models \phi(\mathsf{x}_1,\dots,\mathsf{x}_n) = \psi(\mathsf{x}_{n+1},\dots,\mathsf{x}_l).$$

The converse is proved similarly. $\dashv$

The point of the Lemma is that, for given ϕ,ψ, there are infinitely many identities which may or may not satisfy the left-hand-side of (5-3), because there are infinitely many immediate terms, e.g., z_1 could be any one of

$$\mathsf{p}^1_1(\mathsf{v}_1), \mathsf{p}^2_1(\mathsf{v}_1,\mathsf{v}_1), \mathsf{p}^3_1(\mathsf{v}_1,\mathsf{v}_1,\mathsf{v}_1),\dots;$$

while the right-hand-side of (5-3) involves only finitely many identities (up to alphabetic change), which we can exploit as follows:

An **individual bare identity** in Φ is any identity in form [4-4]

$$(*) \qquad \theta : \phi(\mathsf{x}_1,\dots,\mathsf{x}_k) = \psi(\mathsf{x}_{k+1},\dots,\mathsf{x}_l),$$

where the (not necessarily distinct) individual variables $\mathsf{x}_1,\dots,\mathsf{x}_l$ are chosen from the first l entries in a fixed list $\mathsf{v}_0,\mathsf{v}_1,\dots$ of all individual variables; and the **individual dictionary** of a total structure $\mathbf{A}$ is the set

$$\text{(Ind-Dictionary)} \quad D(\mathbf{A}) = \{\theta \mid \theta \text{ is an individual bare identity and } \mathbf{A} \models \theta\}.$$

This is a finite set (because Φ is finite) and by (2) of Lemma 5.9, for any identity in form [4-4] we can construct an individual bare identity such that

$$\text{(5-4)} \quad \mathbf{A} \models \phi(z_1,\dots,z_k) = \psi(z_{k+1},\dots,z_l) \\ \iff \phi(\mathsf{x}_1,\dots,\mathsf{x}_k) = \psi(\mathsf{x}_{k+1},\dots,\mathsf{x}_l) \in D(\mathbf{A});$$

and then Lemma 5.8 then gives immediately

Theorem 5.10. *For every total, infinite Φ-structure $\mathbf{A}$, the relation of intensional equivalence on $\mathbf{A}$ between extended Φ-programs is decidable.*

The general case. We now turn to the proof of Theorem 5.1 for an arbitrary infinite partial Φ-structure $\mathbf{A}$ which is similar in structure but requires some additional arguments. It will be useful to keep in mind the following example of an irreducible identity which illustrates the changes that we will need to make to the argument:

$$\text{(5-5)} \qquad \phi(\mathsf{p}(\mathsf{u}),\mathsf{u},\mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}),\mathsf{u},\mathsf{v}).$$

An individual variable u is **placed** in an irreducible identity

$$\text{(5-6)} \qquad \phi(z_1,\dots,z_k) = \psi(z_{k+1},\dots,z_l)$$

if $\mathsf{u} \equiv z_i$ for some i; and an assignment σ is **injective** if it assigns distinct values $\sigma(\mathsf{u}) \neq \sigma(\mathsf{v})$ in A to distinct, placed individual variables $\mathsf{u} \not\equiv \mathsf{v}$. We write

$$\text{(5-7)} \quad \mathbf{A} \models_{\text{inj}} \phi(z_1,\dots,z_k) = \psi(z_{k+1},\dots,z_l) \\ \iff_{\text{df}} \text{ for every injective } \sigma,\ \mathbf{A},\sigma \models \phi(z_1,\dots,z_k) = \psi(z_{k+1},\dots,z_l).$$

Lemma 5.11. *With every irreducible identity* (5-6)*, we can associate a sequence* $\mathsf{w}_1, \dots, \mathsf{w}_l$ *of* (not necessarily distinct) *individual and nullary function variables of sort* `ind` *so that for every* (infinite) Φ*-structure* $\mathbf{A}$*,*

$$\text{(5-8)}\quad \mathbf{A} \models_{\text{inj}} \phi(z_1, \dots, z_k) = \psi(z_{k+1}, \dots, z_l) \\ \iff \mathbf{A} \models_{\text{inj}} \phi(\mathsf{w}_1, \dots, \mathsf{w}_k) = \psi(\mathsf{w}_{k+1}, \dots, \mathsf{w}_l).$$

Proof. Call i *new* (in (5-6)) if there is no $j < i$ such that $z_i \equiv z_j$, and set (by induction on i):

(a1) $\mathsf{w}_i :\equiv z_i$, if i is new and z_i is an individual or nullary function variable;
(a2) $\mathsf{w}_i :\equiv$ some fresh nullary function variable (distinct from every z_j and from every w_j with $j < i$), if i is new and $z_i \equiv \mathsf{p}(\mathsf{u}_1, \dots, \mathsf{u}_n)$ with $n > 0$;
(a3) $\mathsf{w}_i :\equiv \mathsf{w}_j$ for the least (and hence every) $j < i$ such that $z_i \equiv z_j$, if i is not new.

Notice that directly from the definition,

$$\text{(5-9)}\qquad \mathsf{w}_i \equiv \mathsf{w}_j \iff z_i \equiv z_j \quad (1 \le i, j \le l),$$

cf. Problem x5.3.

To prove first the direction ($\Longrightarrow$) in (5-8), assume that

$$\mathbf{A} \models_{\text{inj}} \phi(z_1, \dots, z_k) = \psi(z_{k+1}, \dots, z_l),$$

let τ be any injective assignment, and define σ by setting first

$$\sigma(z_i) := \tau(\mathsf{w}_i), \text{ if } z_i \text{ is an individual or nullary function variable.}$$

Since τ is injective, this already insures that, however we extend it, σ will be an injective assignment.

Next, if u is an individual variable which occurs in some $z_i \equiv \mathsf{p}(\mathsf{u}_1, \dots, \mathsf{u}_n)$ and is not placed, set $\sigma(\mathsf{u}) = \bar{\mathsf{u}}$ to a fresh value in A, so that

$$\mathsf{u} \not\equiv \mathsf{v} \Longrightarrow \bar{\mathsf{u}} \neq \bar{\mathsf{v}}.$$

At this point we have defined σ on all the individual and nullary function variables which occur in (5-8) and it assigns distinct values to distinct individual variables. To define it on n-ary function variables with $n > 0$ which occur in (5-8), set

$$\sigma(\mathsf{p})(\bar{\mathsf{u}}_1, \dots, \bar{\mathsf{u}}_n) = \tau(\mathsf{w}_i), \text{ if } \mathsf{p}(\mathsf{u}_1, \dots, \mathsf{u}_n) \equiv z_i;$$

which is a good definition, because if it happens that $\bar{\mathsf{u}}_1 = \bar{\mathsf{v}}_1, \dots, \bar{\mathsf{u}}_n = \bar{\mathsf{v}}_n$ for some variables $\mathsf{v}_1, \dots, \mathsf{v}_n$ such that $\mathsf{p}(\mathsf{v}_1, \dots, \mathsf{v}_n) \equiv z_j$ for some $j \neq i$, then $\mathsf{u}_1 \equiv \mathsf{v}_1, \dots, \mathsf{u}_n \equiv \mathsf{v}_n$, hence $z_i \equiv z_j$ and $\tau(\mathsf{w}_i) = \tau(\mathsf{w}_j)$ by (5-9).

The hypothesis now gives us that $\mathbf{A}, \sigma \models \phi(z_1, \dots, z_k) = \psi(z_{k+1}, \dots, z_l)$, and we verify $\mathbf{A}, \tau \models \phi(\mathsf{w}_1, \dots, \mathsf{w}_k) = \psi(\mathsf{w}_{k+1}, \dots, \mathsf{w}_l)$ by a direct computation:

$$\begin{aligned} \tau(\mathbf{A}, \phi(\mathsf{w}_1, \dots, \mathsf{w}_k)) &= \phi^{\mathbf{A}}(\tau(\mathsf{w}_1), \dots, \tau(\mathsf{w}_k)) \\ &= \phi^{\mathbf{A}}(\sigma(z_1), \dots, \sigma(z_k)) \quad \text{(by the construction)} \\ &= \psi^{\mathbf{A}}(\sigma(z_{k+1}), \dots, \sigma(z_l)) \quad \text{(by the hypothesis)} \\ n &= \psi^{\mathbf{A}}(\tau(\mathsf{w}_{k+1}), \dots, \tau(\mathsf{w}_l)) \quad \text{(by the construction)} \\ &= \tau(\mathbf{A}, \psi(\mathsf{w}_{k+1}, \dots, \mathsf{w}_l)). \end{aligned}$$

The converse direction ($\Longleftarrow$) of (5-8) is proved similarly and we leave it for Problem x5.7. $\dashv$

The lemma reduces the problem of the validity in $\mathbf{A}$ of any identity in (5-6) to the validity in $\mathbf{A}$ of a single identity from a finite list and that would give us the decision method we want by the same argument we used for total structures by appealing to Lemma 5.9—except for the annoying subscript $_{\text{inj}}$, which we must deal with next.

Consider the example in (5-5), for which the construction in the Lemma verified that

$$\mathbf{A} \models_{\text{inj}} \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v}) \iff \mathbf{A} \models_{\text{inj}} \phi(\mathsf{q}_1, \mathsf{u}, \mathsf{v}) = \psi(\mathsf{q}_2, \mathsf{u}, \mathsf{v}),$$

an equivalence which may fail without the subscript $_{\text{inj}}$ by Problem x5.6. Now

$$\mathbf{A} \models \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v}) \Longrightarrow \mathbf{A} \models \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{u}) = \psi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{u});$$

the construction in the Lemma associates the identity $\phi(\mathsf{q}, \mathsf{u}, \mathsf{u}) = \psi(\mathsf{q}, \mathsf{u}, \mathsf{u})$ with $\phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{u}) = \psi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{u})$; and it is quite easy to check that

$$\text{(5-10)} \quad \mathbf{A} \models \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v})$$
$$\iff \mathbf{A} \models_{\text{inj}} \phi(\mathsf{q}_1, \mathsf{u}, \mathsf{v}) = \psi(\mathsf{q}_2, \mathsf{u}, \mathsf{v}) \;\&\; \mathbf{A} \models_{\text{inj}} \phi(\mathsf{q}, \mathsf{u}, \mathsf{u}) = \psi(\mathsf{q}, \mathsf{u}, \mathsf{u}),$$

cf. Problem x5.8. This reduces the validity of $\phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v})$ in $\mathbf{A}$ to the injective validity in $\mathbf{A}$ of two identities which are effectively constructed from $\phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v})$; and the natural extension of this reduction holds for arbitrary irreducible identities which involve the primitives in Φ:

Lemma 5.12. *Let $\theta \equiv \phi(z_1 \dots, z_k) = \psi(z_{k+1}, \dots, z_l)$ be any irreducible identity with $\phi, \psi \in \Phi$, let $(\mathsf{u}_1, \dots, \mathsf{u}_m)$ be an enumeration of the individual variables which are placed in θ, let $\mathcal{E}$ be the set of equivalence relations on the set $\{\mathsf{u}_1, \dots, \mathsf{u}_m\}$ and for any $\sim \in \mathcal{E}$, let*

$$\widetilde{\mathsf{u}}_i :\equiv \mathsf{u}_j \text{ for the least } j \text{ such that } \mathsf{u}_i \sim \mathsf{u}_j \quad (i = 1, \dots, m).$$

Then, for any infinite Φ-structure $\mathbf{A}$,

$$\text{(5-11)} \qquad \mathbf{A} \models \theta \iff \textstyle\bigwedge\!\!\!\bigwedge_{\sim \in \mathcal{E}} \mathbf{A} \models_{\text{inj}} \theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\},$$

where $\theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}$ is the result of replacing in θ each u_i by its representative $\widetilde{\mathsf{u}}_i$ in the equivalence relation $\sim$.

Proof. The direction ($\Longrightarrow$) of (5-11) is trivial: because

$$\mathbf{A} \models \theta \Longrightarrow (\forall \sim \in \mathcal{E})[\mathbf{A} \models \theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}]$$
$$\Longrightarrow (\forall \sim \in \mathcal{E})[\mathbf{A} \models_{\text{inj}} \theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}].$$

For the converse, assume the right-hand-side of (5-11), fix an assignment σ and set

$$\mathsf{u}_i \sim \mathsf{u}_j \iff \sigma(\mathsf{u}_i) = \sigma(\mathsf{u}_j) \qquad \text{(notice that } \sim \text{ depends on } \sigma);$$

σ is injective for the identity $\theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}$ whose placed individual variables are exactly the representatives $\{\widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}$ of the placed individual variables of θ, and so the hypothesis gives

$$\mathbf{A}, \sigma \models \theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}.$$

To infer that $\mathbf{A}, \sigma \models \theta$, check first that for every immediate term z,

$$\sigma(z) = \sigma(z\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}) = \sigma(\widetilde{z});$$

this holds by the definition of $\sim$ if $z \equiv \mathsf{u}_i$ for some i and then trivially in every other case. Finally, compute:

$$\begin{aligned}
&\mathbf{A}, \sigma \models \theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\} \\
&\qquad \Longrightarrow \phi^{\mathbf{A}}(\sigma(\widetilde{z}_1), \dots, \sigma(\widetilde{z}_k)) = \psi^{\mathbf{A}}(\sigma(\widetilde{z}_{k+1}), \dots, \sigma(\widetilde{z}_l)) \\
&\qquad \Longrightarrow \phi^{\mathbf{A}}(\sigma(z_1), \dots, \sigma(z_k)) = \psi^{\mathbf{A}}(\sigma(z_{k+1}), \dots, \sigma(z_l)) \\
&\qquad \Longrightarrow \sigma(\mathbf{A}, \phi(z_1, \dots, z_k)) = \sigma(\mathbf{A}, \psi(z_{k+1}, \dots, z_l)) \Longrightarrow \mathbf{A}, \sigma \models \theta. \quad \dashv
\end{aligned}$$

A **bare Φ-identity** is an identity of the form

$$\phi(\mathsf{w}_1, \dots, \mathsf{w}_k) = \psi(\mathsf{w}_{k+1}, \dots, \mathsf{w}_l), \tag{5-12}$$

where each w_i is an individual or nullary function variable of sort `ind` chosen from the first $2l$ entries in a fixed list

$$\mathsf{v}_0, \mathsf{p}_0^{\mathtt{ind},0}, \mathsf{v}_1, \mathsf{p}_1^{\mathtt{ind},0}, \dots$$

of all such variables; and the **dictionary** of $\mathbf{A}$ is the set

$$D(\mathbf{A}) = \{\theta \mid \theta \text{ is a bare identity and } \mathbf{A} \models_{\text{inj}} \theta\}. \tag{Dictionary}$$

The bare Φ-identities are alphabetic variants of the identities constructed in Lemma 5.11 and there are only finitely many of them, cf. Problem x5.9.

Corollary 5.13. *With every irreducible identity of the form*

$$\theta \equiv \phi(z_1, \dots, z_k) = \psi(z_{k+1}, \dots, z_l),$$

we can associate a finite sequence $\theta_1, \dots, \theta_n$ of bare identities so that for every Φ-structure $\mathbf{A}$*,*

$$\mathbf{A} \models \theta \iff \mathbf{A} \models_{\text{inj}} \theta_1 \;\&\; \cdots \;\&\; \mathbf{A} \models_{\text{inj}} \theta_n \iff \theta_1, \dots, \theta_n \in D(\mathbf{A}). \tag{5-13}$$

Proof. Let $\sim_1, \dots, \sim_n$ be an enumeration of the equivalence relations on the set $\{\mathsf{u}_1, \dots, \mathsf{u}_m\}$ of the placed individual variables, let, for each i, θ_i' be the identity associated with $\theta\{\mathsf{u}_i :\equiv \widetilde{\mathsf{u}}_i \mid i = 1, \dots, m\}$ when $\sim = \sim_i$ by the construction in Lemma 5.11, and let θ_i be the alphabetic variant of θ_i' in which only the allowed variables occur. $\dashv$

For example, to be ridiculously formal, the bare identities associated with Example (5-5) ny (5-10) are

$$\phi(\mathsf{p}_0^{\mathtt{ind},0}, \mathsf{v}_0, \mathsf{v}_1) = \psi(\mathsf{p}_1^{\mathtt{ind},0}, \mathsf{v}_0, \mathsf{v}_1) \text{ and } \phi(\mathsf{p}_0^{\mathtt{ind},0}, \mathsf{v}_0, \mathsf{v}_0) = \psi(\mathsf{p}_0^{\mathtt{ind},0}, \mathsf{v}_0, \mathsf{v}_0).$$

Proof of Theorem 5.1 is now immediate by appealing to Lemma 5.8. $\dashv$

Global intensional equivalence. We might guess that irreducible programs are globally intensionally equivalent only if they are congruent, but this is spoiled by the trivial,

$$\text{(5-14)} \qquad \models E = \text{if } E \text{ then } E \text{ else } E \quad (\text{with } E \text{ explicit of boolean sort})$$

already noticed in Problem x3.2. So we need to adjust:

An irreducible program E is **proper** if none of its parts is an immediate term of the form p or $\mathsf{p}(\mathsf{u}_1, \dots, \mathsf{u}_n)$ and boolean sort.

Theorem 5.14. (1) *Every extended irreducible Φ-program is globally intensionally equivalent to a proper one.*

(2) *Two extended, proper, irreducible Φ-programs are globally intensionally equivalent if and only if they are congruent.*

(3) *The relation of global intensional equivalence between extended Φ-programs is decidable.*

Proof. (1) Replace every part $E_i \equiv \underline{\mathsf{p}^{\mathsf{boole},n}(\mathsf{w}_1, \dots, \mathsf{w}_n)}$ of the program which is immediate and of Boolean sort by $\underline{\text{if } E_i \text{ then } E_i \text{ else } E_i}$.

(2) We need to show that if $E(\vec{\mathsf{x}})$ and $F(\vec{\mathsf{x}})$ are proper, extended irreducible Φ-programs, then

$$\Big(E(\vec{\mathsf{x}}) =^{\mathbf{A}}_{\text{int}} F(\vec{\mathsf{x}}) \text{ for every infinite } \Phi\text{-structure } \mathbf{A}\Big) \Longrightarrow E \equiv_c F;$$

and the idea is to use the hypothesis on a single, suitably "free" total structure $\mathbf{A}$ and then apply Lemma 5.9.

We fix an infinite (countable) set A.

Step 1. For each $\phi \in \Phi$ of sort `ind` and arity $n \geq 0$ (if there are any such), fix a set $R_\phi \subset A$ and an injection $\phi^{\mathbf{A}} : A^n \rightarrowtail R_\phi$, so that

$$\phi \not\equiv \psi \Longrightarrow R_\phi \cap R_\psi = \emptyset \text{ and } \bigcup\{R_\phi \mid \phi \in \Phi\} \text{ is co-infinite.}$$

If arity$(\phi) = 0$, then $R_\phi = \{\overline{\phi}\}$ is a singleton.

It is easy—if a bit tedious—to check that for all immediate terms $z_1, \dots, z_k$, $w_1, \dots, w_n$ and however we complete the definition of $\mathbf{A}$, if ϕ and ψ are of sort `ind`, then

$$\text{(5-15)} \quad \mathbf{A} \models \phi(z_1, \dots, z_k) = \psi(w_1, \dots, w_n) \Longrightarrow \phi(z_1, \dots, z_k) \equiv \psi(w_1, \dots, w_n),$$

cf. Problem x5.11.

Step 2. For any two $\phi, \psi \in \Phi$ of sort `boole` and arities $k > 0, n > 0$ and for every equivalence relation $\sim$ on the set $\{1, \dots, k, k+1, \dots, k+n\}$, we choose a fresh tuple $a_1, \dots, a_k, a_{k+1}, \dots, a_{k+n}$ of elements in A such that

$$a_i = a_j \iff i \sim j$$

and set $\phi^{\mathbf{A}}(a_1, \dots, a_k) = \mathrm{tt}, \quad \psi^{\mathbf{A}}(a_{k+1}, \dots, a_{k+n}) = \mathrm{ff}$. We do this by enumerating all triples $(\phi, \psi, \sim)$ where ϕ, ψ are of boolean sort and arities $k > 0, n > 0$ and $\sim$ is an equivalence relation on $\{1, \dots, k+n\}$, and then successively fixing a fresh tuple $a_1, \dots, a_{k+n}$ for each triple with the required property—which we can do since, at any stage, we have used up only finitely many members of A.

This insures (5-15) for ϕ, ψ of sort `boole` and non-zero arities no matter how we complete the definitions of $\phi^{\mathbf{A}}$.

Step 3. If $\gamma \in \Phi$ is of sort `boole` and arity 0 (a boolean constant), we set $\gamma^{\mathbf{A}} \uparrow$.

At this point we have completed the definition of a structure **A** and the hypothesis gives us that

$$\mathbf{A} \models E_i = F_i \quad (i = 0, \dots K).$$

Lemma 1. For $i = 0, \dots, K$, $E_i \equiv F_i$, unless both E_i and F_i are distinct boolean constants in Φ.

Proof is easy (if tedious) by appealing to Lemmas 5.2 and 5.5 and we leave it for Problem x5.12. $\dashv$ (Lemma 1)

Lemma 2. For each boolean constant $\gamma \in \Phi$,

$$\Big|\{i \leq K \mid E_i \equiv \gamma\}\Big| = \Big|\{j \leq K \mid F_j \equiv \gamma\}\Big|.$$

Proof. Fix γ and define the structure $\mathbf{B} = \mathbf{B}^\gamma$ exactly as we defined **A** in Steps 1 and 2 above but replacing Step 3 by the following:

Step 3^γ. Set $\gamma^{\mathbf{B}} = \mathrm{tt}$ and for every other boolean constant $\delta \in \Phi$, set $\delta^{\mathbf{B}} \uparrow$.

The hypothesis gives us a permutation $\pi : \{0, \dots, K\} \to \{0, \dots, K\}$ such that $\pi(0) = 0$ and

$$\mathbf{B} \models E_i = F_{\pi(i)}, \quad i \leq K.$$

By a slight modification of the detailed case analysis in the proof of Lemma 1 (in Cases [2-5], [3-5] and [4-5]),

if E_i is not a propositional constant in Φ, or tt or ff,

then $F_{\pi(i)}$ is not a propositional constant in Φ, or tt or ff,

and directly from the definition of **B**,

if E_i is a propositional constant in Φ other than γ,

then $F_{\pi(i)}$ is also a propositional constant in Φ other than γ.

So π pairs the parts of E which are tt or γ with the parts of F which are tt or γ, hence

$$\Big|\{i \leq K \mid E_i \equiv \gamma \text{ or } E_i \equiv \mathrm{tt}\}\Big| = \Big|\{j \leq K \mid F_j \equiv \gamma \text{ or } F_j \equiv \mathrm{tt}\}\Big|;$$

but $\Big|\{i \leq K \mid E_i \equiv \mathrm{tt}\}\Big| = \Big|\{j \leq K \mid F_j \equiv \mathrm{tt}\}\Big|$ by Lemma 1, and so the last displayed identity implies the claim in the Lemma. $\dashv$ (Lemma 2)

Lemma 3. $E_0 \equiv F_0$.

Proof. This is true by Lemma 1, if E_0 is not a boolean constant in Φ.

If $E_0 \equiv \gamma$, then the construction in Lemma 2 gives $F_0 \equiv \mathrm{tt} \vee F_0 \equiv \gamma$; and the same construction starting with $\gamma^{\mathbf{B}} = \mathrm{ff}$ gives $F_0 \equiv \mathrm{ff} \vee F_0 \equiv \gamma$, so that

$$(F_0 \equiv \mathrm{tt} \vee F_0 \equiv \gamma) \ \& \ (F_0 \equiv \mathrm{ff} \vee F_0 \equiv \gamma),$$

which implies that $F_0 \equiv \gamma$. $\dashv$ (Lemma 3)

To complete the proof that $E \equiv_c F$, we need to construct a permutation $\rho : \{0, \dots, K\} \rightarrowtail\!\!\!\!\twoheadrightarrow \{0, \dots, K\}$ such that $\rho(0) = 0$ and

$$E_i \equiv F_{\rho(i)} \quad (i = 1, \dots, K). \tag{5-16}$$

We start by setting $\rho(i) = i$ if E_i is not a boolean constant in Φ or $i = 0$, which insures (5-16) for such i by Lemmas 1 and 3. Next, for each boolean constant $\gamma \in \Phi$, we appeal to Lemma 2 to extend ρ in any way so that

$$\rho : \{i \le K \mid E_i \equiv \gamma\} \rightarrowtail\!\!\!\!\twoheadrightarrow \{j \le K \mid F_j \equiv \gamma\}$$

and establishes that $E \equiv_c F$.

Part (2) follows almost immediately from (1) and we leave it for Problem x5.14. $\dashv$

Together with the reduction calculus in Section 4, Theorem 5.14 suggests an obvious way to axiomatize the relation of *global intensional equivalence* between programs, but we will not go into it here.

Problems for Section 5.

Problem x5.1. Give a decision procedure for $\mathbf{A} \models E = F$ in forms [1-1], [1-2], [1-3] and arbitrary $\mathbf{A}$.

Problem x5.2. True or false: for any infinite A and any immediate terms $z_1, \dots, z_m, w_1, w_2, w_3$ with $\mathrm{sort}(w_2) = \mathrm{sort}(w_3) = s \in \{\mathtt{ind}, \mathtt{boole}\}$, we can define a partial function $\phi : A^m \rightharpoonup A_s$ such that

$$(A, \phi) \models \phi(z_1, \dots, z_m) = \text{if } \ w_1 \text{ then } w_2 \text{ else } w_3.$$

Problem x5.3. Prove (5-9), that with the definitions in the proof of Lemma 5.11,

$$\mathsf{w}_i \equiv \mathsf{w}_j \iff z_i \equiv z_j \quad (1 \le i, j \le l). \tag{$*$}$$

Hint: Use induction on i.

Problem x5.4. Prove that if E and F are distinct immediate terms or tt or ff, then $\mathbf{A} \not\models E = F$, for any infinite $\mathbf{A}$.

Problem x5.5. Suppose $\mathbf{A}$ is a total Φ-structure and $\phi, \psi \in \Phi$. Find a sequence of (not necessarily distinct) individual variables $\mathsf{w}_1, \dots, \mathsf{w}_6$ such that

$$\mathbf{A} \models \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v}) \iff \mathbf{A} \models \phi(\mathsf{w}_1, \mathsf{w}_2, \mathsf{w}_3) = \psi(\mathsf{w}_4, \mathsf{w}_5, \mathsf{w}_6).$$

Problem x5.6. Give an example where

$$\mathbf{A} \models \phi(\mathsf{p}(\mathsf{u}), \mathsf{u}, \mathsf{v}) = \psi(\mathsf{p}(\mathsf{v}), \mathsf{u}, \mathsf{v}) \text{ but } \mathbf{A} \not\models \phi(\mathsf{q}_1, \mathsf{u}, \mathsf{v}) = \psi(\mathsf{q}_2, \mathsf{u}, \mathsf{v}).$$

Hint: Set $\phi^{\mathbf{A}}(x, s, t) = \psi^{\mathbf{A}}(y, s, t) = f(s, t)$, with a partial function $f(s, t)$ which converges only when $s = t$.

Problem x5.7. Prove the direction ($\Longleftarrow$) of (5-8), i.e.,

$$\begin{aligned} (*) \quad \mathbf{A} \models_{\text{inj}} \phi(\mathsf{w}_1, \dots, \mathsf{w}_k) = \psi(\mathsf{w}_{k+1}, \dots, \mathsf{w}_l) \\ \Longrightarrow \mathbf{A} \models_{\text{inj}} \phi(z_1, \dots, z_k) = \psi(z_{k+1}, \dots, z_l) \end{aligned}$$

with the choice of variables $\mathsf{w}_1, \dots, \mathsf{w}_l$ made in the proof of Lemma 5.11.

Problem x5.8. Prove (5-10).

Problem x5.9. Prove that the number of bare Φ-identities as in (5-12) is bounded above by $2^{2\text{arity}(\Phi)}|\Phi|^2$, where arity$(\Phi)$ is the largest arity of any $\phi \in \Phi$ and $|\Phi|$ is its cardinality.

Problem x5.10. Prove that if $\phi(z_1, \dots, z_k)$ is irreducible, w is an individual variable and $\mathbf{A} \models \mathsf{w} = \phi(z_1, \dots, z_k)$ for some structure $\mathbf{A}$, then every z_i is an individual variable and w is one of them.

Problem x5.11. Prove (5-15)

$$\mathbf{A} \models \phi(z_1, \dots, z_k) = \psi(w_1, \dots, w_n) \Longrightarrow \phi(z_1, \dots, z_k) \equiv \psi(w_1, \dots, w_n),$$

in Step 1 of the proof of Theorem 5.14.

Problem x5.12. Prove Lemma 1 in the proof of Theorem 5.14.

Problem x5.13. Prove Case (2-2) of Theorem 5.14, that for immediate terms $z_1, \dots, z_l$,

$$\models p(z_1, \dots, z_k) = q(z_{k+1}, \dots, z_l) \Longrightarrow p(z_1, \dots, z_k) \equiv q(z_{k+1}, \dots, z_l).$$

Problem x5.14. Derive (2) of Theorem 5.14 from (1).

Propositional programs. An explicit term E is *propositional* if it has no individual variables, no symbols from Φ and only boolean, nullary function variables,

(Prop terms) $\qquad P :\equiv \mathrm{tt} \mid \mathrm{ff} \mid \mathsf{p}_i^{\mathtt{boole},0} \mid \text{if } P_1 \text{ then } P_2 \text{ else } P_3;$

and a program is *propositional* if all its parts are, e.g.,

$$\text{Liar} :\equiv \mathsf{p} \text{ where } \{\mathsf{p} = \text{if } \mathsf{p} \text{ then ff else tt}\},$$
$$\text{Truthteller} :\equiv \mathsf{p} \text{ where } \{\mathsf{p} = \text{if } \mathsf{p} \text{ then tt else ff}\}.$$

These are significant for the project of using referential intensions to model *meanings* initiated in Moschovakis [1994], [2006] which is far from our topic, but they also bear on the complexity problem for intensional equivalence.

For our purposes here, a *finite graph with $n > 0$ nodes* is a binary relation E on the set $\{0, \dots, n-1\}$, the *edge relation* on the set of nodes of the graph $(\{0, \dots, n-1\}, E)$ in more standard terminology.[13]

Problem x5.15* (Moschovakis [1994], for a related language)**.** Prove that the problem of intensional equivalence between propositional programs is at least as hard as the problem of isomorphism between graphs. HINT: You need to associate with each graph E of size n an irreducible propositional program $\widetilde{E}$ which codes E, so that

$$E \text{ is isomorphic with } F \iff \widetilde{E} =_{\mathsf{int}} \widetilde{F} \iff \widetilde{E} \equiv_c \widetilde{F};$$

and the trick is to use propositional variables p_i, one for each $i < n$ and p_{ij}, one for each pair (i, j) with $i, j < n$.

[13]To the best of my knowledge (and that of Wikipedia), it is still open as I write this whether the graph isomorphism problem is co-NP.

REFERENCES

Siddharth Bhaskar

[2017] *A difference in complexity between recursion and tail recursion*, ***Theory of Computing Systems***, vol. 60, pp. 299–313.

[2018] *Recursion versus tail recursion over* $\overline{\mathbb{F}p}$, ***Journal of Logical and Algebraic Methods in Programming***, pp. 68–90.

Alonzo Church

[1935] *An unsolvable problem in elementary number theory*, ***Bulletin of the American Mathematical Society***, vol. 41, pp. 332–333, This is an abstract of Church [1936].

[1936] *An unsolvable problem in elementary number theory*, ***American Journal of Mathematics***, pp. 345–363, An abstract of this paper was published in Church [1935].

Nachum Dershowitz and Yuri Gurevich

[2008] *A natural axiomatization of computability and proof of Church's Thesis*, ***The Bulletin of Symbolic Logic***, vol. 14, pp. 299–350.

M. Duží

[2014] *A procedural interpretation of the Church-Turing thesis*, ***Church's Thesis: Logic, Mind and Nature*** (Adam Olszewski, Bartosz Brozek, and Piotr Urbanczyk, editors), Copernicus Center Press, Krakow 2013.

Robin Gandy

[1980] *Church's Thesis and principles for mechanisms*, ***The Kleene Symposium*** (J. Barwise, H. J. Keisler, and K. Kunen, editors), North Holland Publishing Co, pp. 123–148.

Yuri Gurevich

[1995] *Evolving algebras 1993: Lipari guide*, ***Specification and validation methods*** (E. Börger, editor), Oxford University Press, pp. 9–36.

[2000] *Sequential abstract state machines capture sequential algorithms*, ***ACM Transactions on Computational Logic***, vol. 1, pp. 77–111.

Neil D. Jones

[1999] *LOGSPACE and PTIME characterized by programming languages*, ***Theoretical Computer Science***, pp. 151–174.

[2001] *The expressive power of higher-order types or, life without CONS*, ***Journal of Functional Programming***, vol. 11, pp. 55–94.

A. S. Kechris and Y. N. Moschovakis

[1977] *Recursion in higher types*, ***Handbook of mathematical logic*** (J. Barwise, editor), Studies in Logic, No. 90, North Holland, Amsterdam, pp. 681–737.

Stephen C. Kleene

[1959] *Recursive functionals and quantifiers of finite types I*, ***Transactions of the American Mathematical Society***, vol. 91, pp. 1–52.

D. E. Knuth

[1973] ***The Art of Computer Programming, Volume 1. Fundamental Algorithms***, second ed., Addison-Wesley.

A. N. Kolmogorov

[1933] ***Foundations of the theory of probability***, Chelsea Publishing Co., New York, English 1956 translation of the 1933 German monograph, edited by Nathan Morrison and with an added bibliography by A. T. Bharucha-Reid.

SAUL A. KRIPKE

[2000] *From the Church-Turing Thesis to the First-Order Algorithm Theorem*, ***Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*** (Washington, DC, USA), LICS '00, IEEE Computer Society, The reference is to an abstract. A video of a talk by Saul Kripke at *The 21st International Workshop on the History and Philosophy of Science* with the same title is posted at `http://www.youtube.com/watch?v=D9SP5wj882w`, and this is my only knowledge of this article.

JOHN LONGLEY AND DAG NORMANN

[2015] ***Higher-order computability***, Springer.

NANCY A. LYNCH AND EDWARD K. BLUM

[1979] *A difference in expressive power between flowcharts and recursion schemes*, ***Mathematical Systems Theory***, pp. 205–211.

YIANNIS N. MOSCHOVAKIS

[1989a] *The formal language of recursion*, ***The Journal of Symbolic Logic***, vol. 54, pp. 1216–1252, posted in ynm's homepage.

[1989b] *A mathematical modeling of pure, recursive algorithms*, ***Logic at Botik '89*** (A. R. Meyer and M. A. Taitslin, editors), vol. 363, Springer-Verlag, Berlin, Lecture Notes in Computer Science, posted in ynm's homepage, pp. 208–229.

[1994] *Sense and denotation as algorithm and value*, ***Logic colloquium '90*** (J. Väänänen and J. Oikkonen, editors), Lecture Notes in Logic, vol. 2, Association for Symbolic Logic, posted in ynm's homepage, together with a corrected and expanded proof of the main Theorem 4.1 of this paper, pp. 210–249.

[1998] *On founding the theory of algorithms*, ***Truth in mathematics*** (H. G. Dales and G. Oliveri, editors), Clarendon Press, Oxford, posted in ynm's homepage, pp. 71–104.

[2001] *What is an algorithm?*, ***Mathematics unlimited – 2001 and beyond*** (B. Engquist and W. Schmid, editors), Springer, posted in ynm's homepage, pp. 929–936.

[2006] *A logical calculus of meaning and synonymy*, ***Linguistics and Philosophy***, vol. 29, pp. 27–89, posted in ynm's homepage.

[2014] *On the Church-Turing Thesis and relative recursion*, ***Logic and Science Facing the New Technologies*** (Peter Schroeder-Heister, Gerhard Heinzmann, Wilfrid Hodges, and Pierre Edouard Bour, editors), College Publications, Logic, Methodology and Philosophy of Science, Proceedings of the 14th International Congress (Nancy), posted in ynm's homepage, pp. 179–200.

[2019] ***Abstract recursion and intrinsic complexity***, ASL Lecture Notes in Logic, vol. 48, Cambridge University Press, posted in ynm's homepage.

YIANNIS N. MOSCHOVAKIS AND VASILIS PASCHALIS

[2008] *Elementary algorithms and their implementations*, ***New computational paradigms*** (S. B. Cooper, Benedikt Lowe, and Andrea Sorbi, editors), Springer, posted in ynm's homepage, pp. 81—118.

MICHAEL S. PATTERSON AND CARL E. HEWITT

[1970] *Comparative schematology*, MIT AI Lab publication posted at http://hdl.handle.net/1721.1/6291 with the date 1978.

GERALD E. SACKS

[1990] ***Higher recursion theory***, Perspectives in Mathematical Logic, Springer.

JERZY TIURYN

[1989] *A simplified proof of DDL < DL*, ***Information and Computation***, vol. 82, pp. 1–12.

J.V. Tucker and J.I. Zucker

[2000] *Computable functions and semicomputable sets on many-sorted algebras*, ***Handbook of Logic in Computer Science*** (S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors), vol. 5, Oxford University Press, pp. 317–523.

Alan M. Turing

[1936] *On computable numbers with an application to the Entscheidungsproblem*, ***Proceedings of the London Mathematical Society***, vol. 42, pp. 230–265, *A correction*, ibid. volume 43 (1937), pp. 544–546.

DEPARTMENT OF MATHEMATICS
UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA, USA
and
DEPARTMENT OF MATHEMATICS, UNIVERSITY OF ATHENS, GREECE
E-mail: ynm@math.ucla.edu
URL: http://www.math.ucla.edu/∼ynm