YIANNIS N. MOSCHOVAKIS

THE LOGIC OF FUNCTIONAL RECURSION[1]

1. INTRODUCTION

Recursive definitions of functions abound in mathematics and logic: in the classical theory of recursive functions on the set N of natural numbers—of course; in (so-called) abstract or generalized recursion theory; and, most significantly, in programming languages, where (especially since the work of Scott [9]) it has become clear that the main "programming construct" is *definition by recursion.* And so it is important to understand how properties of recursively defined functions—follow *logically* from their definitions. The paper could be called **the logic of programs**, but the title above is more specific, more general (in some ways), and also more honest, since the austere, mathematical formulation in which the problem is best understood is quite removed from actual programming practice. The formal proof systems and completeness results of this paper are related to "program verification" very much like predicate logic and its completeness are related to axiomatic set theory; they are certainly relevant, but not of much help in establishing specific, concrete results.

In its most general form, a recursive definition of a function is expressed by a *fixpoint equation* of the form

$$p(u) = f(u,p) \qquad (p : M \rightarrow W, f : M \times (M \rightarrow W) \rightarrow W), \tag{1}$$

in which the functional f provides a method for computing (or attempting to compute) each value $p(u)$ of $p : M \rightarrow W$, perhaps using ("calling") other values of p in the process. In applications, we deal mostly with *mutual recursive definitions* of the form

$$A = f_0(\vec{p}) \text{ where } \{p_1(\vec{u}_1) = f_1(\vec{u}_1,\vec{p}),\ldots,p_n(\vec{u}_n) = f_1(\vec{u}_n,\vec{p})\}; \tag{2}$$

the idea here is that first we "solve" the system

$$p_1(\vec{u}_1) = f_1(\vec{u}_1,\vec{p}),\ldots,p_n(\vec{u}_n) = f_n(\vec{u}_n,\vec{p}),$$

and then we apply the "head" functional f_0 to the "solutions" to get the required value. It is not hard to reduce such mutual recursions to the simpler (1), but their logic becomes more rather than less complex in the process and, in the end, it is best to take (2) as basic. So the language of functional recursion is, in effect, the language of the "where" construct, and it is formalized in the *Formal* (or *Functional*) *Language of Recursion* FLR introduced

[1]During the preparation of this paper the author was partially supported by an NSF Grant.

in [5]. It is possible (and useful) to think of FLR as a very abstract programming language, rich enough so that the more familiar, friendlier programming languages can be "faithfully" interpreted in it.

The basic *Monotone Fixpoint Theorem* insures the existence of a *least solution* for (1), when the range W of the "unknown" function p is a *complete poset*, and the functional f is *monotone*.[2] This (and its corollaries) suffice for the development of denotational semantics for deterministic programming languages. The situation is very different for languages with non-deterministic and "concurrent" constructs, whose interpretation requires the assignment of "canonical solutions" to (1), in situations where W is not a poset and so least solutions do not exist; it then becomes a serious problem how to justify the choice of one proposed method over another. Part of the motivation for this work is the hope that a better understanding of the laws of logic which govern deterministic recursion will help us understand, evaluate and compare the various approaches to "concurrency modeling."

The main contributions of this paper are: **first**, the introduction of a new class of least-fixed-point interpretations for FLR (and programming languages in general), a refinement of denotational semantics which is closer to our intuitive understanding of deterministic programs and mathematically simpler; **second**, the completeness and decidability of an equational proof system $\mathcal{E}_0$ for (absolute) identities between FLR expressions (program equivalence), with respect to the new interpretations; and, **third**, the completeness of a stronger system $\mathcal{E}$ for FLR-*sentences* which are valid consequences of a *first order theory*, suitably defined. The axiom systems $\mathcal{E}_0$ and $\mathcal{E}$ of Sections 7 and 10 are not complete (and could not be complete) for the general *consequence relation*, but they are strong enough so that many natural proofs of program correctness can (in principle) be formalized in them. They are also *sound* for the models of concurrent systems introduced in [6, 7, 8], and so the completeness results (especially) support the proposition that *the logic of non-deterministic and concurrent (recursive) programs is the same as that of deterministic programs.*

Acknowledgment. This paper generalizes and extends some of the basic results of the joint work [1], which deals with FLR_0, the *propositional* (or *equational*) part of FLR. I have omitted specific references to [1] (partly for reasons of space and also because the final form of [1] is not quite ready yet), and I have tried to make this paper self-contained, but, ideally, the two papers should be read together. I am grateful to my students Monica McArthur, Larry Moss and Glen Whitney and to Tonny Hurkens, for their many ideas and invaluable contributions to this project.

[2]A partially ordered set (poset) W is (directed) complete (a dcpo) if every directed $X \subseteq W$ has a least upper bound $\sup X$; $f : U \to V$ is monotone if $x \leq y \Longrightarrow f(u) \leq f(y)$; and (for complete U, V), $f : U \to V$ is *Scott-continuous* if for every non-empty, directed $X \subseteq U$, $f(\sup X) = \sup\{f(x) \mid x \in X\}$. There is no room in this paper to motivate or explain the basic ideas of poset theory and the *fixpoint theory of programs*, and so I will assume that the reader is familiar with them.

Term:	$A :\equiv$	$\mathsf{ff} \mid \mathsf{tt}$
		$\mid p(Z_1, \dots, Z_n)$
		$\mid \mathsf{f}(Z_1, \dots, Z_n, \pi_1, \dots, \pi_m)$
		$\mid \mathsf{if}\ A_0\ \mathsf{then}\ A_1\ \mathsf{else}\ A_2\ \mathsf{fi}$
		$\mid A_0\ \mathsf{where}\ \{p_1(\vec{u}_1) = A_1, \dots, p_n(\vec{u}_n) = A_n\}$
	$Z :\equiv$	$u \mid A$
ℓ-term:	$\pi :\equiv$	$p \mid \ell(u_1, \dots, u_n)A$
Expression:	$E :\equiv$	$u \mid A \mid \pi$
Equation:	$\varepsilon :\equiv$	$u_1 = u_2 \mid u = A \mid A = u \mid A_1 = A_2 \mid \pi_1 = \pi_2$

TABLE 1. The syntax of FLR.

2. THE SYNTAX OF FLR

We fix formal, basic (individual) variables $\mathsf{u}_0, \mathsf{u}_1, \dots$ and for each $n \in \mathbb{N}$, formal function variables $\mathsf{p}_0^n, \mathsf{p}_1^n, \dots$ of arity n. An FLR **signature** is a pair $\boldsymbol{\tau} = (\{\mathsf{f}_i \mid i \in I\}, \text{kind})$ of a set of **functional symbols** $\{\mathsf{f}_i \mid i \in I\}$ and a mapping which assigns to each f_i a tuple $\text{kind}(\mathsf{f}_i) = \langle n, k_1, \dots, k_m \rangle$. The general idea is that in the intended interpretations of FLR($\boldsymbol{\tau}$), each f_i will be interpreted by a functional $f_i(u_1, \dots, u_n, p_1, \dots, p_m)$, where each u_i ranges over the interpretation of the basic variables, and each p_j ranges over the interpretation of function variables of arity k_j. Functional symbols of kind $\langle n \rangle$ (with $m = 0$) will be interpreted by functions, including *nullary* functions when the kind is $\langle 0 \rangle$.[3]

The **expressions** of FLR($\boldsymbol{\tau}$) comprise the **variables**, the **terms**, and the **ℓ-terms**, which will be interpreted by functions and so have arities associated with them. They—and the (well formed) **equations**—are defined simultaneously by the inductive clauses in Table 1 and the following remarks, in which we also specify which variable occurrences are *bound*, the remaining ones being *free*. We use "metavariables" u, v for formal basic variables and p, q, r for function variables, sometime with subscripts and in vectors, $\vec{u} = u_1, \dots, u_n$.

Notice that *basic variables are not terms, but function variables are ℓ-terms*, a mild eccentricity of the language which, however, fits in with the intended semantics.

(1) In $p(Z_1, \dots, Z_n)$ and $\mathsf{f}(Z_1, \dots, Z_n, \pi_1, \dots, \pi_m)$, each Z_i is a basic variable or a term; p is a function variable of arity n; and f is a functional symbol of

[3]There are three inessential differences between this and the original presentation of FLR in [5]. First, only one sort of basic variables is allowed here, as is common in proof theory, and so the truth values cannot be assigned to basic variables, although they are allowed to be values of terms; second, the conditional is "typed" differently, i.e., its first argument must be a term (not a basic variable); and third, I write $\ell(u)$ rather than $\lambda(u)$, to avoid appealing inadvertently to classical properties of the λ-calculus which are not valid in all interpretations of FLR, some of which are strongly intensional. The two versions are equivalent in expressive power for the class of models considered in [5].

kind $\langle n, k_1, \dots, k_m\rangle$ with $k_j = \text{arity}(\pi_j)$. If f is a nullary symbol—a *constant*—we will write f rather than $\mathsf{f}()$.

(2) In the conditional expression, A_0, A_1 and A_2 are terms. We view this expression as an abbreviation of the more formal

$$\mathsf{cond}(\ell()A_0, \ell()A_1, \ell()A_2),$$

which makes it clear that the conditional will be interpreted as if it were a given functional symbol, of kind $\langle 0,0,0,0\rangle$.

(3) In the recursive term A_0 where $\{p_1(\vec{u}_1) = A_1, \dots, p_n(\vec{u}_n) = A_n\}$, each A_i is a term, $p_1, \dots, p_n$ are distinct function variables, and each $\vec{u}_i$ is a (possibly empty) list of distinct basic variables of length arity(p_i). An occurrence of a variable x in the **head** A_0 or some **recursion part** A_i of this term is free, if it is a free occurrence of x in A_i, x is not some p_j, and also x is not one of the basic variables in the list $\vec{u}_i$. For an example, consider

$$\bot \equiv p() \text{ where } \{p() = p()\} \tag{3}$$

which plays a special role in the theory.

(4) In $\ell(\vec{u})A$, A is a term and $u_1, \dots, u_n$ are distinct, basic variables. This ℓ-term has arity n, and an occurrence of a variable u is free in it if it is free in A and not one of the u_is. We also count function variables among the ℓ-terms, basically identifying p with $\ell(\vec{u})p(\vec{u})$. We set $\ell(\vec{u})\ell(\vec{v})A \equiv_{\text{df}} \ell(\vec{u},\vec{v})A$ when the lists $\vec{u}$ and $\vec{v}$ have no common variables, so we can meaningfully prefix ℓ-terms by the ℓ operator.

(5) In a well-formed ℓ-equation $\pi_1 = \pi_2$, the two sides must have the same arity.

For examples of expressions, see the list of axioms in Table 2.

2.1. SUBSTITUTIONS (REPLACEMENTS). By $E\{\vec{u} :\equiv \vec{Z}\}$ we denote the result of replacing in E every free occurrence of each u_i by the corresponding basic variable or term Z_i. If $E(\vec{u})$ stands for an expression in which the variables $\vec{u}$ may occur free, we will also write, as usual, $E(\vec{Z})$ for $E(\vec{u})\{\vec{u} :\equiv \vec{Z}\}$. The substitution $E\{p :\equiv \ell(\vec{u})A\}$ of ℓ-terms or function variables for function variables is defined by induction, using basic variable substitution in the one non-trivial clause, e.g.,

$$p(Z_1, Z_2)\{p :\equiv \ell(u_1,u_2)A\} \equiv A\{u_1 :\equiv Z_1', u_2 :\equiv Z_2'\},$$

where $Z_1' \equiv Z_1\{p :\equiv \ell(u_1,u_2)A\}$, and similarly with Z_2'. And with the abbreviated notation again: $E(\pi) \equiv E(p)\{p :\equiv \pi\}$, for an ℓ-term π.

3. STRICT SEMANTICS

3.1. STRICT FUNCTIONALS. With each (non-empty) set M, we associate the *flat poset*

$$M^\dagger = M \cup \{\mathrm{ff}, \mathrm{tt}, \bot\}, \tag{4}$$

where the truth values ff, tt and $\bot$ (*bottom*, the "undefined") are fixed objects outside M; $\bot \le x$ for every $x \in M^\dagger$; and all points other than $\bot$ are pairwise incomparable. An (extended) **partial function** on a set M is any function $p : M^n \to M^\dagger$, and a (partial, extended, monotone) **strict functional** on M of kind $\langle n, k_1, \dots, k_m \rangle$ is any monotone mapping

$$f : M^n \times (M^{k_1} \to M^\dagger) \times \cdots \times (M^{k_m} \to M^\dagger) \to M^\dagger,$$

so that every n-ary partial function is a functional of kind $\langle n \rangle$. Perhaps the simplest functionals are the *calls* and the *strict conditional*,

$$\begin{aligned} \mathrm{call}_n(u_1, \dots, u_n, p) &= p(u_1, \dots, u_n), \\ \mathrm{cond}(p, q, r) &= \begin{cases} q() & \text{if } p() = \mathrm{tt}, \\ r() & \text{if } p() = \mathrm{ff}, \\ \bot & \text{otherwise}, \end{cases} \end{aligned} \tag{5}$$

but equally natural are the functionals which represent the operations of first order logic,

$$\mathrm{or}(p, q) = \begin{cases} \mathrm{tt} & \text{if } p() = \mathrm{tt} \text{ or } q() = \mathrm{tt}, \\ \mathrm{ff} & \text{if } p() = q() = \mathrm{ff}, \\ \bot & \text{otherwise}, \end{cases}$$

$$E_M(p) = \begin{cases} \mathrm{tt} & \text{if } (\exists u \in M)[p(u) = \mathrm{tt}], \\ \mathrm{ff} & \text{if } (\forall u \in M)[p(u) = \mathrm{ff}], \\ \bot & \text{otherwise.} \end{cases}$$

All but the last of these are *continuous*, but E_M is *discontinuous* when M is infinite.

3.2. FUNCTIONAL STRUCTURES. A (strict) *functional structure* of signature $\tau = (\{\mathsf{f}_i \mid i \in I\}, \mathrm{kind})$ is a tuple

$$\mathfrak{M} = (M, \{f_i \mid i \in I\}),$$

where each f_i is a functional on M with $\mathrm{kind}(f_i) = \mathrm{kind}(\mathsf{f}_i)$. By identifying each relation R_i on M with its characteristic function

$$\chi_{R_i}(\vec{u}) = \text{if } R(\vec{u}) \text{ then tt else ff},$$

we find among these the usual first-order structures of model theory

$$\mathfrak{M} = (M, R_1, \dots, R_l, f_1, \dots, f_m),$$

and, more specifically, the standard structure of arithmetic

$$\mathfrak{N} = (\mathrm{N}, 0, S, P, Z) \tag{6}$$

where $S(x) = x+1$, $P(x) = \text{if } (x = 0) \text{ then } 0 \text{ else } x-1$, and $Z(x) \iff x = 0$. Structures with non-trivial functionals come up naturally as expansions of first order structures by functionals such as those above, e.g.,

$$(\mathfrak{M}, \mathrm{or}, E_M) = (M, \{f_i \mid i \in I\}, \mathrm{or}, E_M).$$

3.3. STRICT DENOTATIONS. A **valuation** (variable assignment) ω in a functional structure $\mathfrak{M}$ assigns some $\omega(\mathsf{u}_i) \in M$ to each basic variable, and some n-ary partial function $\omega(\mathsf{p}_i^n) : M^n \to M^\dagger$ to every function variable of arity n. We now associate with each expression E and each valuation ω a *denotation* $[\![E]\!]_s(\omega)$, so that $[\![E]\!]_s(\omega) = \omega(E)$ if E is a variable; $[\![A]\!]_s(\omega) \in M^\dagger$, if A is a term; and $[\![\pi]\!]_s(\omega)$ is an n-ary partial function, if π is an ℓ-term of arity n. We skip the construction, which is by induction on the definition of expressions and follows familiar lines. The salient points are:

(D1) ff and tt are interpreted by ff and tt.

(D2) Application is interpreted strictly,[4] i.e.,

$$[\![p(Z_1,\dots,Z_n)]\!]_s(\omega) = \begin{cases} \omega(p)([\![Z_1]\!]_s(\omega),\dots,[\![Z_n]\!]_s(\omega)) \\ \qquad\qquad\qquad \text{if } [\![Z_1]\!]_s(\omega),\dots,[\![Z_n]\!]_s \in M, \\ \bot \qquad\qquad \text{otherwise,} \end{cases}$$

and similarly with the substitution of terms or variables in the given functionals.

(D3) Conditional terms are interpreted by the strict conditional (5).

(D4) Recursive terms are interpreted by the taking of mutual, least-fixed-points, and to justify this, we must interleave with the definition a proof that for each E, $[\![E]\!]_s(\omega)$ is *monotone in* ω. Notice that $[\![\bot]\!]_s = \bot$ for the closed term $\bot$ defined in (3).

(D5) $\ell(\vec{u})$ is interpreted by λ-abstraction.

3.4. Recursive definability. The partial functions definable on $\mathfrak{M}$ by closed ℓ-terms are the *recursive partial functions* of $\mathfrak{M}$, and in the case of $\mathfrak{N}$ they are precisely the classical, recursive partial functions on N; on $(\mathfrak{N}, E_{\mathrm{N}})$ we get the *hyperartithmetical* functions, and, more generally, for each first-order $\mathfrak{M}$, the recursive partial functions of $(\mathfrak{M}, E_M)$ are precisely the $\mathfrak{M}$-*inductive* partial functions, with domains of definition the $\mathfrak{M}$-inductive relations on $\mathfrak{M}$ studied in [4]; and for every functional f on N, the recursive, partial functions on $(\mathfrak{N}, f)$ are precisely those which are recursive in the sense of Kleene's type-2 recursion on N. Kleene's higher-type recursion can also be represented (easily and naturally) as strict, FLR-definability on the appropriate structure.[5]

3.5. Strict consequence in FLR. Assuming that all equations listed are well-formed, we write

$$E_1 = F_1, \dots, E_m = F_m \models_s E = F$$

if for every functional structure $\mathfrak{M}$ and valuation ω in $\mathfrak{M}$,

$$[\![E_i]\!]_s(\omega) = [\![F_i]\!]_s(\omega) \text{ for } i = 1,\dots m \Longrightarrow [\![E]\!]_s(\omega) = [\![F]\!]_s(\omega);$$

this is the fundamental notion of **logical consequence** in the category of strict, monotone structures. We will not make a list of strict-valid consequences here since plenty will be put down later on, but notice that the β-**reduction** rule fails,

$$\ell(u)p(u,v) = \ell(u)q(v) \not\models_s p(A,v) = q(v), \tag{7}$$

by the trivial counterexample $p(u,v) = q(v) = \mathrm{tt}$, $A \equiv \bot$.

[4]We might set instead $[\![p(Z)]\!]_s(\omega) = \mathrm{ff}$ (or record an "error" in some other way) when $[\![Z]\!]_s(\omega)$ is ff or tt, but it is easier to handle such finer modelings of computation in the context of the more general $\jmath$-structures of Section 6.

[5]This is essentially the approach to recursion in higher types taken in [2].

4. CONTINUOUS SEMANTICS

4.1. CONTINUOUS POSET STRUCTURES. If W is a complete poset, then each function space

$$\mathrm{Cont}(W^n \to W) = \{p : W^n \to W \mid p \text{ is Scott-continuous}\}$$

is also a complete poset under the pointwise partial ordering, and so it makes sense to require that a functional

$$f : W^n \times \mathrm{Cont}(W^{k_1} \to W) \times \cdots \times \mathrm{Cont}(W^{k_m} \to W) \to W$$

is *continuous*. A *continuous poset structure* of signature $\boldsymbol{\tau} = (\{\mathsf{f}_i \mid i \in I\}, \mathrm{kind})$ is a tuple

$$\mathfrak{W} = (W, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \{f_i \mid i \in I\}), \tag{8}$$

where W is a complete poset; ff and tt are distinct, *discrete* points in W (maximal and comparable only with $\perp$); each f_i is a continuous functional on W with $\mathrm{kind}(f_i) = \mathrm{kind}(\mathsf{f}_i)$; and cond is a continuous functional of kind $\langle 0, 0, 0, 0\rangle$, such that

$$\left.\begin{aligned} \mathrm{cond}(\lambda()\mathrm{tt}, \lambda()\alpha, \lambda()\beta) &= \alpha, \\ \mathrm{cond}(\lambda()\mathrm{ff}, \lambda()\alpha, \lambda()\beta) &= \beta, \\ \mathrm{cond}(\lambda()\perp, \lambda()\alpha, \lambda()\beta) &= \perp. \end{aligned}\right\} \tag{9}$$

Perhaps the simplest example is the continuous structure

$$\mathfrak{M}^\dagger = (M^\dagger, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, R_1^\dagger, \dots, R_l^\dagger, f_1^\dagger, \dots, f_m^\dagger)$$

associated with each first order $\mathfrak{M}$, where cond is the strict conditional and each $f_i^\dagger$ is the strict extension of f_i to $M^\dagger$ (set $= \perp$ when one of its basic arguments is not in M).

4.2. STRICT VS. CONTINUOUS SEMANTICS. The continuous denotation function $[\![E]\!]_c(\omega)$ is defined exactly like the strict one, except that application is honest,

$$[\![p(Z_1, \dots, Z_n)]\!]_c(\omega) = \omega(p)([\![Z_1]\!]_c(\omega), \dots, [\![Z_n]\!]_c(\omega)), \tag{10}$$

and we must verify in each case of the inductive definition that for each E, $[\![E]\!]_c(\omega)$ is *continuous in* ω. Because of (10), and with the natural notion of **continuous consequence**, the β-reduction rule holds,

$$\ell(\vec{u})A = \ell(\vec{u})B \models_c A\{\vec{u} :\equiv \vec{C}\} = B\{\vec{u} :\equiv \vec{C}\}, \tag{11}$$

assuming, of course, that the substitutions are free. This is one, important difference between strict and continuous FLR-consequence. For a simpler one which involves recursion rather than application and no hypotheses, notice that

$$\models_s p(q()) \text{ where } \{p(u) = \mathrm{tt}, q() = q()\} = \perp \tag{12}$$

but

$$\models_c p(q()) \text{ where } \{p(u) = \mathrm{tt}, q() = q()\} = \mathrm{tt}. \tag{13}$$

From this we infer that neither of the equations in (12) and (13) should be provable in the logic of recursion, since we would want the theorems to be valid under both the strict and the continuous interpretations.

4.3. DOMAINS. Suppose $\mathfrak{W}$ is a continuous structure as in (8) and there exists a homeomorphism

$$\lambda : \mathrm{Cont}(W \to W) \longrightarrow W \tag{14}$$

of W with its (continuous) function space, and consider the expansion

$$(\mathfrak{W}, \mathrm{id}, \mathrm{ap}, \lambda) = (W, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \{f_i \mid i \in I\}, \mathrm{id}, \mathrm{ap}, \lambda),$$

where $\mathrm{id}(x) = x$ and $\mathrm{ap}(x, y) = \lambda^{-1}(x)(y)$. Structures of this type were first constructed by Scott [9] to model the (untyped, extensional) λ-**calculus** and its "applied" extensions, and we can see better the connection with these systems if we introduce the abbreviations

$$A \cdot B \equiv \mathrm{ap}(A, B), \;\; \lambda(u)A \equiv \lambda(\ell(u)A)$$

on the terms of FLR($\boldsymbol{\tau}$) for the relevant $\boldsymbol{\tau}$. The more complex posets of "domain theory" with "reflection" properties like (14) can also be represented in this way as continuous, FLR structures, with the various imbeddings and homeomorphisms among function spaces coded by functionals among the "givens" of the structure.

5. STREAMS

If a is a constant in the signature, then

$$\models_{s,c} p(\mathsf{a}) \text{ where } \{p(u) = p(\mathsf{a})\} = \perp \tag{15}$$

i.e., this equation is valid in both strict and continuous semantics, simply because the least-fixed-point of $p(u) = p(a)$ is the totally undefined function $\lambda(u)\perp$. On the other hand, it is quite easy to "program" the term $p(\mathsf{a})$ where $\{p(u) = p(\mathsf{a})\}$ in Pascal or C, and if a stands for some act, like "*ring the bell*", and we run the resulting program, we will then get an unending sequence of "rings" instead of a dull "hang"—at least until the stack overflows. We construct here some very simple models of FLR which account for the expected interpretation of (15), and also motivate the definition of the more general structures in the next section.

5.1. STREAM STRUCTURES. Fix a set A (of "acts") and a set M (of "values"), disjoint from A, with specified, distinct points $0, 1 \in M$—think of $M = \mathbb{N}$. A *convergent stream* is any finite (possibly empty) sequence $\alpha = \langle a_0, \ldots, a_{n-1}, m\rangle$ with $a_0, \ldots, a_{n-1} \in A$, and $m = [\alpha] \in M$, the *value returned* by α. A *divergent stream* is any finite or infinite sequence $\alpha = \langle a_0, a_1, \ldots\rangle$ of acts, and it returns no value, $[\alpha] = \perp$. The set $\mathrm{Str} = \mathrm{Str}_{A,M}$ of all convergent and divergent streams is a complete poset under the "initial part" relation $\sqsubseteq$, and it carries the (continuous) operation of **sequential execution**,

$$\alpha\,; \beta = \begin{cases} \langle a_0, \ldots, a_{n-1}, b_0, b_1, \ldots\rangle & \text{if } \alpha = \langle a_0, \ldots, a_{n-1}, m\rangle \\ & \qquad\qquad \text{is convergent,} \\ \alpha & \text{if } \alpha \text{ is a divergent.} \end{cases}$$

Suppose now $\{f_i \mid i \in I\}$ are continuous functionals on Str, let ff and tt be the (no-act) streams $\langle 0\rangle$ and $\langle 1\rangle$ respectively, and let

$$\mathfrak{S} = (M, \mathrm{Str}, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \,;\,, \{\langle a, 1\rangle \mid a \in A\}, \{f_i \mid i \in I\}),$$

where each $\langle a, 1\rangle$ is is a constant (nullary) function and the (strict) conditional is defined by (5). For the semantics, we let the basic variables vary over M and the function variables over $(M \rightharpoonup \mathrm{Str})$, and we define the denotation function $[\![E]\!]_{str}(\omega)$ by simply copying all the clauses in the definition of the strict denotation $[\![E]\!]_s(\omega)$, except when $E \equiv p(\vec{Z})$ or $E \equiv f(\vec{Z}, \vec{\pi})$, when some of the Z_is are terms. For these cases, we first "execute" the acts of the stream arguments (from left to right) and then "pass" the returned values (if any) to the function, for example,

$$[\![\mathsf{p}(\mathsf{u}_1, A, \mathsf{u}_2, \mathsf{u}_3, B)]\!]_{str}(\omega) = \alpha\,;\beta\,;p(u_1, [\alpha], u_2, u_3, , [\beta]), \tag{16}$$

where $\alpha = [\![A]\!]_{str}(\omega)$, $\beta = [\![B]\!]_{str}(\omega)$, $p = \omega(\mathsf{p})$ and $u_i = \omega(\mathsf{u}_i)$. The definition is a bit messy in the general case, but very natural, and it assigns the correct, expected value to the term in (15), if a is the formal constant naming $\langle a, 1\rangle$.

5.2. Input dependent and interactive streams. The stream model is too simple to account for much of the behavior of programming languages with side-effects, e.g., it makes no room for state, input or interaction. One step up from it are the **input dependent streams**, which are functions $\alpha : S \rightharpoonup \mathrm{Str}_{A,M}$ from a given set of *states* S and so can account (partially) for input. We also assume that each act $a \in A$ induces a function $s \mapsto sa$ on the states, and then define sequential composition by:

$$(\alpha\,;\beta)(s) = \begin{cases} \alpha(s) & \text{if } \alpha(s) \text{ is divergent,} \\ \alpha(s)\,;\beta(sa_0a_1\cdots a_{n-1}) & \text{if } \alpha(s) = \langle a_0, \ldots, a_{n-1}, m\rangle. \end{cases}$$

Input dependent stream structures and the denotation function $[\![E]\!]_{ids}(\omega)$ are defined by a routine extension of the construction above.

Still richer in structure are the **interactive streams**, which are (suitably restricted) partial functions $\alpha : S^* \rightharpoonup (A \cup M \cup \{\perp\})$ from finite sequences of states to acts or values, and model the behavior of *reactive agents* in an interactive environment. (The agent α is reacting at each stage to a new state, produced by the execution of acts by other agents since the last time α acted.) We will not give any details of this construction here, since we do not need it, but it is natural and very simple.[6]

6. $\jmath$-SEMANTICS

In defining stream semantics, we emphasized the important principle "execute the acts first and then pass the returned value" but passed over without com-

[6]Input dependent streams are (essentially) the *procedures* and interactive streams the *behaviors* of [6], where these constructions were employed to build models of concurrency. A simpler language (with no value passing) was used in [6], but it is routine to extend the constructions given there to the present context.

ment the choice to evaluate the stream arguments *from left-to-right* in (16); but there is nothing sacrosanct about left-to-right evaluation, and it is clear that we would get a different denotation function if we chose to evaluate the streams from right-to-left, or (indeed) if we "interweaved" the evaluation of the stream arguments in some fixed way.[7] If the terms are interpreted in some poset more complex than Str, then more than "order of evaluation" comes into play: we have a real problem in assigning a value to $\mathsf{p}(A, B)$, when $[\![A]\!], [\![B]\!] \in W$, but the interpretation $p : M^2 \to W$ of p is defined only for arguments in M. The obvious way to handle this is to make **value passing** a primitive, one of the "givens" in the structure.

6.1. $\jmath$-STRUCTURES AND DENOTATIONS. A **monotone functional** on a set M to a complete poset W is any monotone mapping

$$f : M^n \times (M^{k_1} \to W) \times \cdots \times (M^{k_m} \to W) \to W,$$

and a $\jmath$-**structure** of signature $\boldsymbol{\tau} = (\{\mathsf{f}_i \mid i \in I\}, \text{kind})$ is a tuple

$$\mathfrak{M} = (M, W, \mathrm{ff}, \mathrm{tt}, \text{cond}, \{\jmath^n \mid n \in \mathbb{N}\}, \{f_i \mid i \in I\}), \tag{17}$$

where the following conditions hold:

($\jmath$1) M is a non-empty set and W is a complete poset, containing the discrete elments ff and tt. We make no special assumptions on the relationship between M and W—it may be that $M = W$, $M \subsetneq W$ or $M \cap W = \emptyset$.

($\jmath$2) cond is a monotone functional of kind $\langle 0, 0, 0, 0\rangle$ satisfying (9).

($\jmath$3) Each f_i is a monotone functional on M to W with $\text{kind}(f_i) = \text{kind}(\mathsf{f}_i)$.

($\jmath$4) Each $\jmath^n : (M^n \to W) \to \text{Mon}(W^n \to W)$ is a monotone mapping which assigns a monotone, n-ary function $\jmath^n(p) : W^n \to W$ to each n-ary function $p : M^n \to W$ and does not disturb the nullary functions, $\jmath^0(p) = p$. We will generally write

$$\jmath(p, \alpha_1, \dots, \alpha_n) = \jmath^n(p)(\alpha_1, \dots, \alpha_n)$$

for the values of these imbeddings, skipping the superscripts and placing the arguments from W on par with the function argument.

To interpret the FLR-expressions in a $\jmath$-structure $\mathfrak{M}$, we let the basic variables vary over M and the function variables over arbitrary functions $p : M^n \to W$. The denotation function $[\![E]\!]_\jmath(\omega)$ is defined exactly like the strict denotation in Section 3, except that the conditional is given by the structure and application uses the given imbeddings. The general definition is a bit messy and we will skip it, but the following adaptation of example (16) explains the idea:

$$[\![\mathsf{p}(\mathsf{u}_1, A, \mathsf{u}_2, \mathsf{u}_3, B)]\!]_\jmath(\omega) = \jmath^2(\lambda(ab)p(u_1, a, u_2, u_3, b), \alpha, \beta), \tag{18}$$

[7]The original manual for C [3] deliberately leaves the order of evaluation of expressions unspecified, and so delegates the definition of the precise semantics of the language to the specific compiler—unless we adopt a non-deterministic interpretation, which is by no means trivial because unrestricted recursion is allowed. (The concurrency model of [6] can account for this.) All the (simple) C-compilers I have used evaluate from right-to-left.

where $\alpha = [\![A]\!]_\jmath(\omega)$, $\beta = [\![B]\!]_\jmath(\omega)$, $p = \omega(\mathsf{p})$ and $u_i = \omega(\mathsf{u}_i)$. For another example, involving one of the givens,

$$[\![\mathsf{f}(A, \mathsf{u}, B, C, \pi_1, \pi_2)]\!]_\jmath = \jmath^3(\lambda(abc)f(a, u, b, c, \sigma_1, \sigma_2), \alpha, \beta, \gamma), \tag{19}$$

with the notation of (18) for the terms A, B, C and the variable u, and $\sigma_i = [\![\pi_i]\!]_\jmath(\omega)$ for the ℓ-terms π_1 and π_2.

6.2. Strict vs. continuous vs. $\jmath$-semantics. Each strict, functional structure has an obvious $\jmath$-associate

$$\mathfrak{M}_\jmath = (M, \{f_i \mid i \in I\})_\jmath = (M, M^\dagger, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \jmath_s, \{f_i \mid i \in I\}),$$

where cond is the strict conditional and $\jmath_s$ is the strict extension map,

$$\jmath_s(p, \alpha_1, \dots, \alpha_n) = \begin{cases} p(\alpha_1, \dots, \alpha_n) & \text{if } \alpha_1, \dots, \alpha_n \in M, \\ \perp & \text{otherwise,} \end{cases}$$

and it is also obvious that $[\![E]\!]_s(\omega) = [\![E]\!]_\jmath(\omega)$—the definitions of these two denotation functions are identical. The same is true for stream structures, which can be identified with $\jmath$-structures with the imbedding

$$\jmath_{str}(p, \alpha_1, \dots, \alpha_n) = \alpha_1 ; \dots ; \alpha_n ; p([\alpha_1], \dots, [\alpha_n]).$$

The situation is just a bit more complex for continuous structures, because for them we restricted the function variables to vary over continuous functions, but the following is true:[8]

6.3. **Proposition.** *Suppose $\mathfrak{W} = (W, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \{f_i \mid i \in I\})$ is a continuous structure in which the poset W has the Scott property, i.e., every two compatible points in W have a least upper bound, for each $p : W^n \to W$ let*

$$\jmath_c^n(p) = \sup\{q \mid q \in \mathrm{Cont}(W^n \to W), q \le p\},$$

define $f_{i,c}$ analogously, and let $\mathfrak{W}_\jmath = (W, W, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, \jmath_c, \{f_{i,c} \mid i \in I\})$.

(1) *Each $\jmath_c^n : (W^n \to W) \to \mathrm{Cont}(W^n \to W)$ is defined and monotone.*

(2) *If $[\![E]\!]_\jmath(\omega)$ is the denotation function on $\mathfrak{W}_\jmath$, then for every expression E and every valuation ω which assigns continuous functions to the function variables, $[\![E]\!]_c(\omega) = [\![E]\!]_\jmath(\omega)$.*

We skip the (simple) proof of this proposition, but it is worth noting that *the imbeddings $\jmath_c^n$ are not continuous.* Perhaps this witnesses the known "inefficiency" of the continuous (call-by-name) interpretation of recursive definitions, but it also makes this unified approach to the known interpretations depend essentially on the full *Monotone Fixpoint Theorem*, rather than the much simpler special case for continuous functionals which is traditionally used in denotational semantics.

These simple observations imply that if $\Gamma \models_\jmath E = F$, then the consequence $\Gamma \models E = F$ is also valid under the strict, the continuous and the stream

[8]We could avoid the restriction of this proposition to posets with the Scott property by interpreting the n-ary function variables in a $\jmath$-structure over some (almost) arbitrary, complete subposet of $(M^n \to W)$, but it is not clear that this increased applicability of the theory is worth the additional complications it introduces.

interpretations of FLR. The converse is not true for any of these semantics. For example, easily,

$$\models_s p(a) \text{ where } \{p(u) = f(u,b)\} = q(b) \text{ where } \{q(v) = f(a,v)\}, \tag{20}$$

$$\models_{str} f(a,b) = p(a) \text{ where } \{p(u) = f(u,b)\}, \tag{21}$$

but (20) fails in stream semantics (and hence in $\jmath$-semantics also), and (21) fails in $\jmath$-semantics, using the right-to-left evaluation on streams. It is implicit in the proof of the Main Theorem 9.1 that (somewhat surprizingly), the input-dependent stream structures (with arbitrary, continuous givens) are sufficient to determine all $\jmath$-valid equations between FLR expressions.

7. THE SYSTEM $\mathcal{E}_0$

A **sequent** of the proof system $\mathcal{E}_0$ is a formal expression $\Gamma \Rightarrow \varepsilon$, where $\Gamma \equiv \varepsilon_1, \dots, \varepsilon_n$ is a *set* of FLR($\boldsymbol{\tau}$) equations, the *hypotheses*, and $\varepsilon \equiv E = F$ is a single equation, the *conclusion*. The **provable sequents** (theorems) of $\mathcal{E}_0$ are defined inductively by the **axioms** and (closure) **rules** in Table 2 and by the comments and explanations below, and we write

$$\Gamma \vdash E = F \text{ if and only if } \Gamma \Rightarrow E = F \text{ is a theorem.}$$

In commenting on the axioms and rules, we will also discuss the more interesting points of the proof of the following result, whose details we will skip:

7.1. THE SOUNDNESS THEOREM. *If* $\Gamma \vdash E = F$, *then* $\Gamma \models_\jmath E = F$.

The *structural* axioms and rules (S1) – (S3) are standard, and so are the axioms for equality (FL1) – (FL4), where f^* in (FL3) is any of the functional symbols or the conditional. (FL5) is the *generality* or *abstraction rule*, and (FL6) is a very restricted form of **β-reduction** which requires some explanation.

7.2. IMMEDIACY AND LINEARITY. An ℓ-term $\ell(z_1, \dots, z_n)A$ is **immediate** if $A \equiv p(\vec{u})$ where $\vec{u}$ is a sequence of basic variables; and **linear** if the variables $z_1, \dots, z_n$ occur *exactly once* in A, and in the same order in which they occur in the prefix. (So $\ell()p(u,s,w,t,w)$, $\ell(s)p(u,s,w,t,w)$, $\ell(u,t)p(u,s,w,t,w)$ and every function variable p ($\equiv \ell(\vec{u})p(\vec{u})$) are immediate and linear, but $\ell(w)p(u,s,w,t,w)$ and $\ell(s,t)p(u,t,w,s)$ are not linear and $\ell(s)p(s,\mathsf{a})$ is linear but not immediate.)

It is quite simple to verify that (FL6) is valid in $\jmath$-semantics, but it cannot be strengthened by omitting either of the restrictions, because of (7) and:

$$\ell(u)p(u) = \ell(u)q(\mathsf{a},u) \not\models_{str} p(\mathsf{b}) = q(\mathsf{a},\mathsf{b}). \tag{22}$$

(Suppose a and b name the constants $\langle a,1\rangle$ and $\langle b,1\rangle$, and let $q(u,v) = \mathrm{tt} = \langle 1\rangle$; now $[\![q(\mathsf{a},u)]\!]_{str} = \langle a,1\rangle$ and we can validate the hypothesis by setting $p(u) = \langle a,1\rangle$; but then $[\![p(\mathsf{b})]\!]_{str} = \langle b,a,1\rangle$, while $[\![q(\mathsf{a},\mathsf{b}]\!]_{str} = \langle a,b,1\rangle$.)

The axioms (C1) – (C3) about the conditional are also natural (and $\jmath$-valid, by definition), but (FR1) – (FR7) about the recursion construct require some

(S1)	$\varepsilon \Rightarrow \varepsilon$
(S2)	From $\Gamma \Rightarrow \varepsilon$, infer $\Gamma, \varepsilon_1 \Rightarrow \varepsilon$
(S3)	From $\Gamma \Rightarrow \varepsilon_1$ and $\Gamma, \varepsilon_1 \Rightarrow \varepsilon$, infer $\Gamma \Rightarrow \varepsilon$
(FL1)	$\Rightarrow E = E,\ E = F \Rightarrow F = E,\ E = F, F = G \Rightarrow E = G$
(FL2)	$X_1 = Y_1,\ \ldots,\ X_n = Y_n \Rightarrow p(X_1, \ldots, X_n) = p(Y_1, \ldots, Y_n)$
(FL3)	$E_1 = F_1,\ \ldots,\ E_n = F_n \Rightarrow f^*(E_1, \ldots, E_n) = f^*(F_1, \ldots, F_n)$
(FL4)	$\Rightarrow p = \ell(\vec{u})p(\vec{u})$
(FL5)	From $\Gamma \Rightarrow A(\vec{z}) = B(\vec{z})$, infer $\Gamma \Rightarrow \ell(\vec{x})A(\vec{x}) = \ell(\vec{y})B(\vec{y})$
	provided none of the variables in $\vec{z}$ occurs free in Γ
(FL6)	$\ell(\vec{x})A(\vec{x}) = \ell(\vec{y})B(\vec{y}) \Rightarrow A(\vec{Z}) = B(\vec{Z})$
	provided *either* $\ell(\vec{x})A(\vec{x})$, $\ell(\vec{y})B(\vec{y})$ are immediate and linear
	or $\vec{Z}$ is a sequence of basic variables
(C1)	$\Rightarrow \text{if } \mathrm{tt} \text{ then } A \text{ else } B \text{ fi} = A$
(C2)	$\Rightarrow \text{if ff then } A \text{ else } B \text{ fi} = B$
(C3)	$\Rightarrow \text{if } \perp \text{ then } A \text{ else } B \text{ fi} = \perp$
(FR1)	$\Rightarrow p(\vec{X}, A, \vec{Y}) = p(\vec{X}, r(), \vec{Y}) \text{ where } \{r() = A\}$
(FR2)	$\Rightarrow f^*(\vec{E}, \ell(\vec{u})A, \vec{F}) = f^*(\vec{E}, r, \vec{F}) \text{ where } \{r(\vec{u}) = A\}$
(FR3)	$\Rightarrow f(\vec{Z}, \vec{p}) = r(\vec{Z}) \text{ where } \{r(\vec{u}) = f(\vec{u}, \vec{p})\}$
(FR4)	$\Rightarrow \big(B_0(\vec{q}, \vec{p}) \text{ where } \{q_1(\vec{v}_1) = B_1(\vec{q}, \vec{p}), \ldots, q_m(\vec{v}_m) = B_m(\vec{q}, \vec{p})\}\big)$
	$\text{where } \big\{p_1(\vec{u}_1) = A_1, \ldots, p_n(\vec{u}_n) = A_n\big\}$
	$= B_0(\vec{r}, \vec{p}) \text{ where } \big\{r_1(\vec{v}_1) = B_1(\vec{r}, \vec{p}), \ldots, r_m(\vec{v}_m) = B_m(\vec{r}, \vec{p}),$
	$p_1(\vec{u}_1) = A_1, \ldots, p(\vec{u}_n) = A_n\big\}$
(FR5)	$\Rightarrow A_0(\vec{p}) \text{ where } \big\{p_1(\vec{u}_1) = B_0(\vec{q}, \vec{p})$
	$\text{where } \big\{q_1(\vec{v}_1) = B_1(\vec{q}, \vec{p}), \ldots, q_m(\vec{v}_m) = B_m(\vec{q}, \vec{p})\big\},$
	$p_2(\vec{v}_2) = A_2(\vec{p}), \ldots, p_n(\vec{u}_n) = A_n(\vec{p})\big\}$
	$= A_0(\vec{p}) \text{ where } \big\{p_1(\vec{u}_1) = B_0(\vec{r}, \vec{p}), r_1(\vec{v}_1) = B_1(\vec{r}, \vec{p}), \ldots,$
	$r_m(\vec{v}_m) = B_m(\vec{r}, \vec{p}), p_2(\vec{v}_2) = A_2(\vec{p}), \ldots, p_n(\vec{u}_n) = A_n(\vec{p})\big\}$
	provided no variable in $\vec{u}_1$ is free in any B_i with $i > 0$
(FR6)	$\Rightarrow A_1 \text{ where } \{p_1(\vec{u}_1) = A_1, \ldots, p_n(\vec{u}_n) = A_n\}$
	$= p_1(\vec{u}_1) \text{ where } \{p_1(\vec{u}_1) = A_1, \ldots, p_n(\vec{u}_n) = A_n\}$
(FR7)	$\Rightarrow A \text{ where } \{\ \} = A$
(FC)	The Finite Closure Rule.
(RI)	The Recursion Inference Rule.

TABLE 2. The axiom system $\mathcal{E}_0$.

Note: All substitutions are assumed free, and the variables r, $\vec{r}$ on the right of (FR1) – (FR5) are fresh—i.e., they do not occur on the left of these equations.

attention, as they express the most basic equational properties of definition by functional recursion. In (FR2), f^* stands for any functional symbol or the conditional. Most interesting are (FR4) and (FR5), the **Bekič-Scott equations**, which reduce nested recursion to mutual recursion; and (FR6), the **fixpoint equation**, which asserts that the recursion construct indeed defines fixed points of the mutual recursion determined by its parts.[9]

The validity of these equations in all $\jmath$-structures can be established easily by standard arguments of fixpoint recursion.

This leaves the last two, basic rules of inference about recursion which require some explanation.

7.3. FINITE ITERATES. The least fixed point of a monotone functional $f(u,p)$ is obtained by iteration starting with $\bot$,

$$\overline{p}^0(u) = f(u, \lambda(u)\bot), \quad \overline{p}^\xi(u) = f(u, \sup_{\eta<\xi}\overline{p}^\eta),$$

and setting $\overline{p} = \overline{p}^\xi$, for the least ordinal ξ such that $\overline{p}^\xi = \sup_{\eta<\xi}\overline{p}^\eta$. The Finite Closure Rule (FC) asserts that if $\xi = k$ is a finite ordinal (a natural number), then $\overline{p} = \overline{p}^k$. It is a bit messy to express in FLR, partly because we must deal with mutual recursions, and (more substantially) because we do not have a general "replacement" rule, and so the finite iterates of a recursive term are also recursive terms.

For a given recursive term

$$A \equiv A_0(\vec{p}) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1,\vec{p}), \dots, p_n(\vec{u}_n) = A_n(\vec{u}_n,\vec{p})\},$$

set, by induction on k, simultaneously for $i = 0, \dots, n$

$$\begin{aligned} \alpha_i^0(\vec{u}_i,\vec{p}) &\equiv A_i(\vec{u}_i,\vec{r}) \text{ where } \{r_1(\vec{u}_1) = p(\vec{u}_1), \dots, r_n(\vec{u}_n) = p(\vec{u}_n)\} \\ \alpha_i^{k+1}(\vec{u}_i,\vec{p}) &\equiv A_i(\vec{u}_i,\vec{r}) \text{ where } \{r_1(\vec{u}_1) = \alpha_1^k(\vec{u}_1,\vec{p}), \\ &\qquad\qquad \dots, r_n(\vec{u}_n) = \alpha_n^k(\vec{u}_n,\vec{p})\}, \end{aligned}$$

where $\vec{r}$ are fresh variables. Now the (syntactic) kth iterate of A is

$$A^k \equiv \alpha_0^k(\vec{\bot}),$$

so that, for example,

$$A^0 \equiv A_0(\vec{r}) \text{ where } \{r_1(\vec{u}_1) = \bot, \dots, r_n(\vec{u}_n) = \bot\}.$$

It is easy to verify (directly from the definitions) that A^k denotes in $\jmath$-structures exactly the kth iterate of the recursion which determines $[\![A]\!]_\jmath$.

The Finite Closure Rule FC. For each recursive term A, with α^k and A^k defined as above,

$$\ell(\vec{u}_1)\alpha_1^2(\vec{u}_1,\vec{\bot}) = \ell(\vec{u}_1)\alpha_1^1(\vec{u}_1,\vec{\bot}), \dots, \ell(\vec{u}_n)\alpha_n^2(\vec{u}_n,\vec{\bot}) = \ell(\vec{u}_n)\alpha_n^1(\vec{u}_n,\vec{\bot}) \\ \Rightarrow A = A^1.$$

This says only that if the recursion stabilizes in the second stage, then this

[9]The fresh variables $\vec{r}$ are needed in (FR4) and (FR5) to avoid conflicts when some q_j occurs free in some A_i; the simpler versions with $r_i \equiv q_i$ are theorems when no variable occurs both free and bound on the left, and we will tacitly assume this in applying the axioms in the sequel.

is the fixpoint, but the general fact about recursions which stop after k steps follows from it easily.

7.4. THE RECURSION INFERENCE RULE. Suppose we want to prove that *if $f(u,v,p)$ is a monotone functional and $f(u,v,p) = f(v,u,p)$, for all u, v, then*

$$\begin{aligned}&p(u,v) \text{ where } \{p(u,v) = f(u,v,p)\}\\ &\qquad = q(v,t,u) \text{ where } \{q(u,t,v) = f(u,v,q(\cdot,t,\cdot)), r(s) = g(s)\},\end{aligned}$$

where $q(\cdot,t,\cdot)) = \lambda(u,v)q(u,t,v)$. First we notice that the occurrence or r is spurious—r never figures in the value of the expression on the right—and then that there is a plausible connection between p on the left and q on the right, namely that for all t,

$$\lambda(u,v)p(u,v) = \lambda(u,v)q(v,t,u). \tag{23}$$

To make the idea precise, we let $\overline{p}^\xi$ and $\overline{q}^\xi$ be the iterates of the recursions which determine the fixpoints $\overline{p}$ and $\overline{q}$, and (using the assumed commutativity of f) show by (trivial, transfinite) induction that for all u, v, ξ and t,

$$\overline{p}^\xi(u,v) = \overline{q}^\xi(v,t,u),$$

from which the desired equation follows immediately.

The Recursion Inference Rule formalizes this method of proof, and is viewed best as a classical *elimination rule* for the recursion construct. It is unfortunately a bit messy to make precise, and all we can do in this space is to put it down and mention that its soundness for $\jmath$-semantics is (quite easily) proved after the manner of the example. The "connections" used in it are generalizations of (23): finite systems of immediate ℓ-terms which relate some of the recursion variables of one recursion term with some of another, enough so that $A = B$ can be established in every $\jmath$-structure by (transfinite) induction, as in the example.

A (basic variable) **projection** is a map

$$\sigma(u_0,\ldots,u_{n-1}) = \langle u_{\pi(0)},\ldots,u_{\pi(m-1)}\rangle \tag{24}$$

on variable tuples determined by a function $\pi : \{i \mid i < m\} \to \{j \mid j < n\}$, where (conventionally) $\sigma(\vec{v}) = \langle\,\rangle$ if $m = 0$. These are useful in describing variable substitutions: e.g., if $\sigma(u,v) = \langle u,v,u\rangle$ and $E(x,y,z)$ is an expression, then $E(\sigma(u,v)) \equiv E(u,v,u)$ and $E(\sigma(x,x)) \equiv E(x,x,x)$.

Suppose Γ is a set of equations and

$$\begin{aligned}A &\equiv A_0 \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1),\ldots,p_m(\vec{u}_m) = A_m(\vec{u}_m)\}\\ B &\equiv B_0 \text{ where } \{q_1(\vec{v}_1) = B_1(\vec{v}_1),\ldots,q_n(\vec{v}_n) = B_n(\vec{v}_n)\},\end{aligned} \tag{25}$$

in which the function variables $p_1,\ldots,p_m,q_1,\ldots,q_n$ are all distinct; and suppose the list $\vec{x} = x_1,\ldots,x_k$ includes all basic variables which are free in either A or B. A finite set of equations

$$\Sigma = \{\pi_1 = \rho_1,\ldots,\pi_c = \rho_c\}$$

between *immediate ℓ-terms* is a **preconnection** of A and B relative to Γ if:

(1) For each i, $\pi_i \equiv \ell(\vec{z})p_k(\sigma_i(\vec{z},\vec{x}))$ and $\rho_i \equiv \ell(\vec{z})q_l(\tau_i(\vec{z},\vec{x}))$, with suitable p_k, q_l, where the basic variables in $\vec{z}$ are fresh for A and B, and σ_i, τ_i are projections;

and its **claims** are the sequents

(2) $\Gamma, \Sigma \Rightarrow A_0 = B_0$, and $\Gamma, \Sigma \vdash A_k(\sigma(\vec{z},\vec{x})) = B_l(\tau(\vec{z},\vec{x}))$, for every equation $\ell(\vec{z})p_k(\sigma(\vec{z},\vec{x})) = \ell(\vec{z})q_l(\tau(\vec{z},\vec{x}))$ in Σ.

The Recursion Inference Rule RI. Suppose A and B are recursive terms as in (25), no p_i occurs free in any B_l, no q_j occurs free in any A_k, and none of the variables in $\vec{p}$, $\vec{q}$ are free in Γ: *from the claims in any preconnection between A and B relative to Γ, infer $\Gamma \Rightarrow A = B$.*

A **connection** is a preconnection whose claims are all provable, and, most often, we use **RI** in the following form:

RI. *If a connection exists between A and B relative to Γ, then $\Gamma \vdash A = B$.*

7.5. Example. To see why the restriction on the variables $\vec{p}$ and $\vec{q}$ is needed, notice that if r does not occur in A, then $\vdash A = A\,\mathsf{where}\,\{r(\vec{u}) = B\}$, for any B; because $A = A\,\mathsf{where}\,\{\ \}$ is an axiom, and the empty set is a connection of $A\,\mathsf{where}\,\{r(\vec{u}) = B\}$ with $A\,\mathsf{where}\,\{\ \}$. (But it is not a connection if r occurs in A, which is why we cannot prove $r() = r()\,\mathsf{where}\,\{r() = r()\}$ (i.e., $r() = \bot$).

8. FORMAL PROOFS

After a few basic facts, we will prove here the key lemma we need for the Main Theorem in the next section.

8.1. **Lemma.** *With the syntactic conventions of Table 1 and assuming that all substitutions are free:*

*1. $\pi = \rho \vdash E(\pi) = E(\rho)$, *if π and ρ are linear and immediate ℓ-terms.*

*2. *The Part Replacement Property. If $\Gamma \vdash A_i = B_i$ for $i = 0, \dots, n$ and no p_i occurs free in Γ, then*

$$\Gamma \vdash A_0\,\mathsf{where}\,\{p_1(\vec{u}_1) = A_1, \dots, p_n(\vec{u}_n) = A_n\}$$
$$= B_0\,\mathsf{where}\,\{p_1(\vec{u}_1) = B_1, \dots, p_n(\vec{u}_n) = B_n\}.$$

*3. *The order in which the parts are listed in a recursive term is irrelevant.*

*4. *The Duplication Equation.*

$$A_0(q,r,\vec{p})\,\mathsf{where}\,\{q(\vec{u}_1) = B(q,\vec{p}), r(\vec{u}_1) = B(r,\vec{p}),$$
$$p_1(\vec{u}_2) = A_1(q,r,\vec{p}), \dots, p_n(\vec{u}_n) = A_n(q,r,\vec{p})\}$$
$$= A_0(q,q,\vec{p})\,\mathsf{where}\,\{q(\vec{u}_1) = B(q,\vec{p}),$$
$$p_1(\vec{u}_2) = A_1(q,q,\vec{p}), \dots, p_n(\vec{u}_n) = A_n(q,q,\vec{p})\}$$

*5. $\vec{X} = \vec{Y} \vdash E(\vec{X}) = E(\vec{Y})$, *if $\vec{X}$ and $\vec{Y}$ are basic variables or terms.*

*6. *If F is obtained from E by a renaming of bound variables, then $\vdash E = F$.*

*7. *The Bekič-Scott Lemma. With r fresh and $r(\vec{u}, \cdot) \equiv \ell(\vec{v})r(\vec{u},\vec{v})$,*

$$B_0(\vec{u},\vec{q}) \text{ where } \{q_1(\vec{v}_1) = B_1(\vec{q}),\dots,q_m(\vec{v}_m) = B_m(\vec{q})\}$$
$$= B_0(\vec{u},\vec{r}(\vec{u},\cdot)) \text{ where } \{r_1(\vec{u},\vec{v}_1) = B_1(\vec{r}(\vec{u},\cdot)), \dots, r_m(\vec{v}_m) = B_m(\vec{r}(\vec{u},\cdot))\}.$$

*8. *The Extended Fixpoint Equation.*
$$p_1(\tau\vec{u}_1) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1)\dots\}$$
$$= A_1(\tau\vec{u}_1) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1)\dots\}.$$

*9. $\vdash A_0 \text{ where } \{p_1(\vec{u}_1) = p_2(\tau\vec{u}_1), p_2(\vec{u}_2) = A_2(\vec{u}_2),\dots\}$
$$= A_0 \text{ where } \{p_1(\vec{u}_1) = A_2(\tau\vec{u}_1), p_2(\vec{u}_2) = A_2(\vec{u}_2),\dots\}.$$

*10. *The Bottom Rule. If* $\Gamma \vdash A_i(\vec{u}_i,\vec{p}) \text{ where } \{\vec{p} = \vec{\perp}\} = \perp$, *for* $i = 1,\dots,n$ *and no variable in* $\vec{p}$ *or any* $\vec{u}_j$ *occurs free in* Γ, *then*
$$\Gamma \vdash A_0(\vec{p}) \text{ where } \{p_1(\vec{u}_1) = A_1,\dots,p_n(\vec{u}_n) = A_n\}$$
$$= A_0(\vec{p}) \text{ where } \{\vec{p} = \vec{\perp}\},$$
where $\vec{p} = \vec{\perp}$ *abbreviates the obvious.*

*11. $\vdash p(\vec{u}) \text{ where } \{p(\vec{u}) = \text{if } p(\tau\vec{u}) \text{ then } A \text{ else } B \text{ fi}\} = \perp.$

Proof. *1 is proved by induction on the length of the given expression $E(r)$, and all the cases are trivial, except perhaps for the following two.

$E(r) \equiv r(\vec{Z})$. If $\pi \equiv \ell(\vec{x})A(\vec{x})$ and $\rho \equiv \ell(\vec{y})B(\vec{y})$, then $E(\pi) \equiv A(\vec{X}^*)$ and $E(\rho) \equiv B(\vec{Y}^*)$, where $\vec{X}^* = \vec{Y}^*$ by the induction hypothesis; now $A(\vec{X}^*) = B(\vec{X}^*)$ by Axiom (FL6) (because $A(\vec{x})$ and $B(\vec{y})$ are linear and immediate), and $B(\vec{X}^*) = B(\vec{Y}^*)$ by (FL2).

$E(r) \equiv A_0(\vec{p},r) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{p},r),\dots,p_n(\vec{u}_n) = A_n(\vec{p},r)\}$. Now $E(\pi)$ and $E(\rho)$ are obtained by substituting π and ρ for r in each $A_i(\vec{p},r)$, the induction hypothesis gives us $A_i(\vec{p},\pi) = A_i(\vec{p},\rho)$, and we can get $E(\pi) = E(\rho)$ from *2—but we do not have *2 yet. Let instead
$$C \equiv A_0(\vec{q},\rho) \text{ where } \{q_1(\vec{u}_1) = A_1(\vec{q},\rho),\dots,q_n(\vec{u}_n) = A_n(\vec{q},\pi)\},$$
where the variables $\vec{q}$ are fresh. By the induction hypothesis,
$$\pi = \rho, \vec{p} = \vec{q} \vdash A_i(\vec{p},\pi) = A_i(\vec{q},\rho),$$
$$\vec{p} = \vec{q} \vdash A_i(\vec{p},\rho) = A_i(\vec{q},\rho),$$
for $i \le n$, and so $\{p_i = q_i \mid i = 1,\dots,n\}$ is a connection which establishes $A(\pi) = C$ and $A(\rho) = C$ from $\pi = \rho$.

*2, *3 and *4 are easily proved by the same method, of introducing fresh recursion variables and applying **RI** twice, with $\{q = r', r = r', \vec{p} = \vec{p}'\}$ the connection required for *4. *5 and *6 are proved by induction on E, using Part Replacement *2 when E is a recursive term, and for *7, we use *1 to verify that $\{\ell(\vec{z}_i)q_i(\vec{z}_i) = \ell(\vec{z}_i)r_i(\vec{u},\vec{z}_i) \mid i = 1,\dots,m\}$ is a connection which establishes the required identity. For *8, abstract on the fixpoint rule (FR6) by (FL5) and then use variable specification, (FL6).

For *9, first insure that no variable occurs both free and bound, and then (showing the variables and with only two parts, for simplicity) compute:

$$\begin{aligned}
&A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = p_2(\tau\vec{u}_1), p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}\\
&\quad= A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = r(\tau\vec{u}_1),\\
&\qquad\qquad r(\vec{u}_2) = A_2(\vec{u}_2,p_1,r), p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}\\
&\qquad\qquad\qquad\text{(by Duplication)}\\
&\quad= A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = r(\tau\vec{u}_1) \text{ where } \{r(\vec{u}_2) = A_2(\vec{u}_2,p_1,r)\},\\
&\qquad\qquad p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}, \text{ (by (FR5))}\\
&\quad= A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = A_2(\tau\vec{u}_1,p_1,r)\\
&\qquad\qquad \text{where } \{r(\vec{u}_2) = A_2(\vec{u}_2,p_1,r)\},\\
&\qquad\qquad p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}\}, \text{ (by *8)}\\
&\quad= A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = A_2(\tau\vec{u}_1,p_1,r),\\
&\qquad\qquad r(\vec{u}_2) = A_2(\vec{u}_2,p_1,r), p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}\\
&\qquad\qquad\qquad\text{by (FR5)}\\
&\quad= A_0(p_1,p_2) \text{ where } \{p_1(\vec{u}_1) = A_2(\tau\vec{u}_1,p_1,p_2), p_2(\vec{u}_2) = A_2(\vec{u}_2,p_1,p_2)\}\\
&\qquad\qquad\qquad\text{(by Duplication).}
\end{aligned}$$

For *10, the hypothesis gives the antecedent for an application of the Finite Closure Rule **FC**, which then yields the required conclusion.

Finally, for *11, first compute:

$$\begin{aligned}
\bot &= \text{if } \bot \text{ then } A \text{ else } B \text{ fi} \quad \text{(C3)}\\
&= \text{if } q() \text{ then } A \text{ else } B \text{ fi where } \{q() = \bot\} \text{ (FR2)}\\
&= \text{if } p(\tau\vec{u}) \text{ then } A \text{ else } B \text{ fi where } \{p(\vec{u}) = \bot\},
\end{aligned}$$

the last via the connection $\{\ell(\vec{u})q() = \ell(\vec{u})p(\vec{u})\}$. Now the Bottom Rule (BR) (with $C(p) \equiv p(\vec{u})$) gives

$$p(\vec{u}) \text{ where } \{p(\vec{u}) = \text{if } p(\tau\vec{u}) \text{ then } A \text{ else } B \text{ fi}\} = p(\vec{u}) \text{ where } \{p(\vec{u}) = \bot\},$$

and the right-hand-side is $= \bot$ by expanding $\bot$ as above and using another trivial connection. ⊣

We can use *7 to get a more general version of the Bekič-Scott Rule without the restriction on the variables $\vec{u}_1$—hence its name; notice that, despite appearances, $\vec{u}$ is not free in $B_1, B_2, \ldots$ on the right side of *7.[10]

8.2. **Lemma** (Chains of conditionals). *If for $i = 1,\ldots,n$, $A_i \equiv p_{i+1}(\tau_i\vec{u}_i)$ or $A_i \equiv$ if $p_{i+1}(\tau_i\vec{u}_i)$ then B_i else C_i fi (with $n+1 = 1$ and arbitrary projections τ_i), then*

$$\vdash p_1(\vec{u}_1) \text{ where } \{p_1(\vec{u}_1) = A_1, \ldots, p_n(\vec{u}_n) = A_n\} = \bot.$$

Proof. The hypothesis (together with (C3)) sets up an application of the Bottom Rule *10, which then yields the required conclusion. ⊣

The significance of the next notion will not become clear until we use it in the next section.

8.3. SIMPLIFIED TERMS. A recursive term (with no free function variables)

$$A \equiv A_0 \text{ where } \{p_1(\vec{u}_1) = A_1, \ldots, p_m(\vec{u}_m) = A_m\} \tag{26}$$

is **simplified** if it satisfies (I) – (V) below, where, in (II) we also classify the recursion parts of these terms into *forms* and we assign to each of them a *character*.

[10]This more general Bekič-Scott Rule was taken as basic in [5].

(I) $A_0 \equiv p_1(\vec{x})$, with some basic variables $\vec{x}$, and these are the only free occurrences of variables of A.

(II) Each recursion part A_k is a **form** and has a **character**, as follows.

(SF1) *Fixed forms.* A_k is $\perp$, ff or tt, and it is its own character.

(SF2) *ȷ-forms.* $A_k \equiv p_i(Z_1, \dots, Z_a)$, where each Z_j is either a basic variable or an immediate term; the character of A_k is the pair $\langle \mathsf{j}, s\rangle$, where s is the number of Z_js which are terms (rather than variables).

(SF3) *Functional forms.* $A_k \equiv \mathsf{f}_i(\vec{u}, \vec{\pi})$, where $\vec{u}$ is a list of basic variables and $\vec{\pi}$ is a list of immediate ℓ-terms; the character of A_k is $\langle \mathsf{f}, i\rangle$.

(SF4) *Conditional forms.* $A_k \equiv$ if C_1 then C_2 else C_3 fi, where C_1, C_2 and C_3 are immediate terms; the character of A_k is cond.

The character of a variable p_i is the same as the character of the form A_i which defines it.

(III) Every ȷ-form $A_k \equiv p_i(Z_1, \dots, Z_a)$ is *strict*, i.e., its character is $\langle \mathsf{j}, n+1\rangle$ for some n (not every Z_j is a variable).

(IV) If $A_k \equiv$ if $p_i(\tau_k \vec{u}_k)$ then C_2 else C_3 fi is a cond-form, then p_i does not have fixed character. (A_k is not *immediately resolvable.*)

(V) There is no *chain of conditionals*, i.e., no sequence of variables

$$q_1 \equiv p_{j_1}, \dots, q_K \equiv p_{j_K},$$

so that for $i = 1, \dots, K$, $A_{j_i} \equiv$ if $q_{i+1}(\tau \vec{u}_{j_i})$ then B_i else C_i fi or $A_{j_i} \equiv q_{i+1}(\tau \vec{u}_{j_i})$ (with $K + 1 = 1$).

8.4. **Lemma.** *Every term without free function variables is provably equal to a simplified one.*

Proof. It is easy to see that by applying repeatedly the axioms (FR1) – (FR7), Lemma 8.1 (especially Part Replacement and the Bekič-Scott Lemma), we can transform every term without free function variables to a provably equal *almost simplified* term, i.e., one which satisfies (I) and (II) in 8.3. To insure that (III) – (V) also hold, we will use the following three facts, where A is almost simplified.

(T1) *Elimination of conditional chains.* If A has a conditional chain as in (V) and A^* is obtained from A by replacing each part $q_i(\vec{v}_i) = A_{j_i}$ by $q_i(\vec{v}_i) = \perp$, then $\vdash A = A^*$.

(T2) *Elimination of resolvable conditionals.* If A has a conditional part

$$p_k(\vec{u}_k) = \text{if } p_i(\tau_k \vec{u}_k) \text{ then } C_2 \text{ else } C_3 \text{ fi},$$

where A_i is a fixed form, then $\vdash A = A^*$, where A^* is obtained by replacing the definition of p_k with its obvious value: $p_k(\vec{u}_k) = \perp$, if $A_k \equiv \perp$, $p_k(\vec{u}_k) = C_2$, if $A_k \equiv$ tt, and $p_k(\vec{u}_k) = C_3$, if $A_k \equiv$ ff.

(T3) *Reduction of non-strict ȷ-forms.* If A has parts $p_k(\vec{u}_k) = p_i(\tau_k \vec{u}_k)$, $p_i(\vec{u}_i) = A_i(\vec{u}_i)$ where $A_i(u_i)$ is not a ȷ-form, and if A^* is obtained from A by replacing the definition of p_k by $p_k(\vec{u}_k) = A_i(\tau_k \vec{u}_k)$, then $\vdash A = A^*$.

These are all easy to verify, using 8.2 for (T1), axioms (C1) – (C3) for (T2), and *9 for (T3). (To show (T1), for example, first insert K copies of the chain by the Duplication Equation *4, each defining the i'th term in the chain; and then isolate each of these by Bekič-Scott and replace it by $q_i(\vec{u}_i) = \bot$, using Lemma 8.2 and Part Replacement.)

The *rank* of an almost simplified term A is the triple $r = \langle o, c, b \rangle$, where o is the number of non-fixed forms in A; c is the number of cond-forms in A; and b is the number of non-strict $\jmath$-forms, those with character $\langle \mathrm{j}, 0 \rangle$. We order ranks *lexicographically.* It is clear that an application of (T1) lowers the rank, and so does an application of (T2)—although it may introduce new chains and increase b; and an application of (T3) also lowers the rank, *if A_i is not a* cond-*form*—although it may make some of the existing conditionals immediately resolvable. We now apply these transformations (with the restriction on (T3)), in any order, as long as it is possible, until we obtain in the end an almost simplified term A^*, provably equivalent to the given A, which satisfies (IV) and (V). It may still contain some non-strict $\jmath$-forms $p_k(\vec{u}_k) = p_i(\tau_k \vec{u}_k)$, with $p_i(\vec{u}_i) = A_i(\vec{u}_i)$ a cond-form, but now we can apply (T3) to eliminate them, clearly without violating (IV) or (V) in the result (although the rank will be increased). $\dashv$

9. COMPLETENESS OF $\mathcal{E}_0$ FOR EQUATIONS

In this section we will outline a proof of the following[11]

9.1. **Main Theorem.** (1) *If $\models_\jmath E = F$, then $\vdash E = F$.* (2) *An expression equation $E = F$ is $\jmath$-valid if and only if it is true in all continuous $\jmath$-structures.* (3) *The class of $\jmath$-valid expression equations is decidable.*

9.2. Reasonably free structures A $\jmath$-structure (17) is **reasonably free** if M has at least two members and the following conditions are satisfied.

(1) Every functional f_i of $\mathfrak{M}$ is injective and never takes on the values $\bot$, tt or ff; and for $i \neq j$, f_i and f_j have disjoint ranges.

(2) If $\beta = \mathrm{cond}(p, q, r) = \mathrm{cond}(p', q', r')$ and $p() \notin \{\bot, \mathrm{tt}, \mathrm{ff}\}$, then β is not in the range of any f_i; $\beta \notin \{\bot, \mathrm{ff}, \mathrm{tt}\}$; and $p = p'$, $q = q'$, $r = r'$.

(3) If p is n-ary with $n > 0$ and $\alpha_1, \dots, \alpha_n \in W$, then $\beta = \jmath(p, \alpha_1, \dots, \alpha_n) \notin \{\bot, \mathrm{tt}, \mathrm{ff}\}$; β is not in the range of any f_i; and $\beta \neq \mathrm{cond}(p', q, r)$ if $p'() \notin \{\bot, \mathrm{tt}, \mathrm{ff}\}$. In addition, $\jmath$ is strongly injective on non-nullary arguments, i.e., if $\jmath(p, \alpha_1, \dots, \alpha_m) = \jmath(q, \beta_1, \dots, \beta_n)$ and $m, n > 0$, then $m = n$ and for $i = 1, \dots, m$, $\alpha_i = \beta_i$.

9.3. **Lemma.** *For each* FLR *signature $\boldsymbol{\tau}$, there exists a continuous, reasonably free $\boldsymbol{\tau}$-structure $\mathfrak{M}$.*

Proof. We let $M = \mathbb{N}$ and $W = (\mathbb{N} \to \mathrm{Str}_{A,\mathbb{N}})$ be the set of functions from $\mathbb{N}$ to the set of streams which return integers and execute acts in the set

$$A = \{\mathrm{f}_i \mid i \in I\} \cup \{\mathrm{j}_{i+1} \mid i \in \mathbb{N}\} \cup \{\mathrm{ff}, \mathrm{tt}, \mathrm{cond}\}$$

[11]For the "propositional" part of FLR, this result is proved in [1], which also gives references to related, older work (about (2)).

of the formal symbols of FLR($\boldsymbol{\tau}$) and a symbol j_n for each $\jmath_n$ with $n > 0$. We let $\mathtt{tt} = \langle 1\rangle$, $\mathtt{ff} = \langle 0\rangle$, and we interpret each f_i by a functional which codes its name and all its arguments, e.g.,[12]

$$f_i(n,p,q)(s) = \langle \mathsf{f}_i, 1\rangle ; \begin{cases} \langle n+2\rangle, & \text{if } (s)_0 = 0,\\ p()((s)_1), & \text{if } (s)_0 = 1,\\ q((s)_1,(s)_2)((s)_3), & \text{if } (s)_0 = 2,\\ \bot, & \text{otherwise.}\end{cases}$$

The imbedding is defined using the same idea, e.g.,

$$\jmath_2(p,\alpha,\beta)(s) = \langle \mathsf{j}_2, 1\rangle ; \begin{cases} \alpha((s)_1), & \text{if } (s)_0 = 0,\\ \beta((s)_1), & \text{if } (s)_0 = 1,\\ p((s)_1,(s)_2)((s)_3), & \text{if } (s)_0 = 2,\\ \bot, & \text{otherwise.}\end{cases}$$

Ditto for the conditional, except that we give the right values when its first argument is $\bot$, $\mathtt{tt}$ or $\mathtt{ff}$—in all other cases we start with **cond** and then we code all three arguments. $\dashv$

9.4. Strict $\jmath$-forms. Suppose $A_k \equiv p_i(Z_1,\dots,Z_a)$ is a recursion part in a simplified term, so that by (III) of 8.3 at least one Z_i is not a variable. We call these *$\jmath$-forms* because their evaluation in $\jmath$-semantics involves the imbedding $\jmath$, and we will sometimes re-write them as

$$A_k \equiv \jmath(\ell(t_{b_1},\dots,t_{b_s})p_i(z_1,\dots,z_a), Z_{b_1},\dots,Z_{b_s}) \tag{27}$$

indicating how they are to be evaluated: here $Z_{b_1},\dots,Z_{b_s}$ are exactly those Z_j's which are terms (not variables), $t_{b_1},\dots,t_{b_s}$ is a sequence of fresh, distinct basic variables to refer to them, and

$$z_j \equiv \begin{cases} Z_j, & \text{if } Z_j \text{ is a variable,}\\ t_{b_j}, & \text{if } Z_j \text{ is a term.}\end{cases}$$

For example, we rewrite

$$p(s_1, r_2(s_1,s_3), s_4, r_3()) \equiv \jmath(\ell(t_2,t_4)p(s_1,t_2,s_4,t_4), r_2(s_1,s_3), r_3()).$$

If all the Z_i's are terms, then the $\jmath$-form is especially simple,

$$\begin{aligned} p(Z_1,Z_2,Z_3) &\equiv \jmath(\ell(t_1,t_2,t_3)p(t_1,t_2,t_3), Z_1,Z_2,Z_3)\\ &\equiv \jmath(p,Z_1,Z_2,Z_3).\end{aligned}$$

Notice that the ℓ-terms which appear in these $\jmath$-forms are linear; this is key both to the interpretation and to the proof theory of the system.

9.5. Notation and terminology. If A is simplified as in (26) and $\mathfrak{M}$ is any structure, we will let $\overline{p}_1,\dots,\overline{p}_m$ be the least fixed points of the system which determines the value of A on $\mathfrak{M}$. Note that these functions do not depend on the interpretation of the free variables of A, which occur only in the head A_0.

[12]Here $(s)_i$ is is the i'th term of the sequence coded by s, relative to some fixed coding of integer-tuples by integers.

Suppose A and B are both simplified, $E = F$ is an equation in the function variables $p_1, \dots, p_m, q_1, \dots, q_n$ which occur in these terms, and $\mathfrak{M}$ is a structure. We will say that $E = F$ **is true** in $\mathfrak{M}$ if every valuation ω which assigns $\overline{p}_i$ to each p_i and $\overline{q}_j$ to each q_j satisfies $E = F$. In particular, a preconnection Σ between A and B is true in $\mathfrak{M}$ if every equation in Σ is true in $\mathfrak{M}$.

9.6. **Lemma.** *If A is simplified as in* (26)*, $\mathfrak{M}$ is reasonably free and*

$$A_k \equiv \text{ if } p_i(\tau\vec{u}_k) \text{ then } C_2 \text{ else } C_3 \text{ fi} \tag{28}$$

is a cond*-form of A, then the functions $\overline{p}_i$ and $\overline{p}_k$ on M never take on values in the set $\{\bot, \mathrm{ff}, \mathrm{tt}\}$.*

Proof. Let us say that the cond-form A_k in (28) has *level* 0 if the character of p_i is not cond, and (inductively) *level* $r+1$ if A_i has level r. Now (V) implies that every cond-form has a level, and we can prove the Lemma by induction on the level.

If A_k has level 0, then A_i cannot be fixed by (IV), and so it is either a functional form or a $\jmath$-form; in the first case, $\overline{p}_i(\vec{u}_i)$ is a value of some functional f_j, and so outside $\{\bot, \mathrm{ff}, \mathrm{tt}\}$ by (1) of 9.2; in the second $\overline{p}_i(\vec{u}_i)$ is a value of $\jmath_n$ for some $n > 0$ since A_i is a strict $\jmath$-form by (III), and so again $\overline{p}_i(\vec{u}_i)$ is outside $\{\bot, \mathrm{ff}, \mathrm{tt}\}$ by (3) of 9.2; and, in either case, $\overline{p}_k$ does not take a value in $\{\bot, \mathrm{ff}, \mathrm{tt}\}$ by (2) of 9.2.

If A_k has level > 0, then $\overline{p}_i$ does not take on any of the forbidden values by the induction hypothesis, and so neither does $\overline{p}_k$ by (2) of 9.2 again. $\dashv$

The next, key Lemma deals only with (basically) **closed** preconnections, whose ℓ-terms have no free basic variables.

9.7. **Lemma.** *Suppose Σ is a closed preconnection between two simplified terms*

$$\begin{aligned} A &\equiv p_1(\vec{x}) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1), \dots, p_m(\vec{u}_m) = A_m(\vec{u}_m)\} \\ B &\equiv q_1(\vec{y}) \text{ where } \{q_1(\vec{v}_1) = B_1(\vec{v}_1), \dots, q_n(\vec{v}_n) = B_n(\vec{v}_n)\} \end{aligned} \tag{29}$$

which contains the identity

$$\ell(\vec{w})p_k(\sigma\vec{w}) = \ell(\vec{w})q_l(\tau\vec{w}), \tag{30}$$

and let $\mathfrak{M}$ be a reasonably free structure.

(1) *If either* (1a) *or* (1b) *are true, then Σ is not true in $\mathfrak{M}$.*

(1a) *A_k and B_l have different character.*

(1b) *$A_k \equiv \mathsf{f}_i(\sigma_k\vec{u}_k, \vec{\pi})$ and $B_l \equiv \mathsf{f}_i(\tau_l\vec{v}_l, \vec{\rho})$ have the same, functional character, but $\sigma_k\sigma\vec{w} \not\equiv \tau_l\tau\vec{w}$.*

(2) *If $A_k \equiv \mathsf{f}_i(\sigma_k\vec{u}_k, \pi_1(\vec{u}_k), \dots, \pi_s(\vec{u}_k))$ and $B_l \equiv \mathsf{f}_i(\tau_l\vec{v}_l, \rho_1(\vec{v}_l), \dots, \rho_s(\vec{v}_l))$ have the same functional character but* (1b) *does not apply, let*

$$\Sigma' = \Sigma \cup \{\ell(\vec{w})\pi'_j(\sigma\vec{w}) = \ell(\vec{w})\rho'_j(\tau\vec{w}) \mid j = 1, \dots, s\},$$

where π'_j and ρ'_j are alphabetic variants of π_j and ρ_j with the same ℓ-prefix and fresh bound variables; then Σ is true in $\mathfrak{M}$ if and only if Σ' is true in $\mathfrak{M}$.

(3) *If* $A_k(\vec{u}_k) \equiv \mathsf{if}\ C_1\ \mathsf{then}\ C_2\ \mathsf{else}\ C_3\ \mathsf{fi}$ *and* $B_l(\vec{v}_l) \equiv \mathsf{if}\ D_1\ \mathsf{then}\ D_2\ \mathsf{else}\ D_3\ \mathsf{fi}$ *both have character* cond *and*

$$\Sigma' = \Sigma \cup \{\ell(\vec{w})\big(C_j\{\vec{u}_k :\equiv \sigma\vec{w}\}\big) = \ell(\vec{w})\big(D_j\{\vec{v}_l :\equiv \tau\vec{w}\}\big) \mid j = 1,2,3\},$$

then Σ *is true in* $\mathfrak{M}$ *if and only if* Σ' *is true in* $\mathfrak{M}$.

(4) *If* A_k *and* B_l *have the same* $\jmath$*-form* $\langle \mathrm{j}, s\rangle$, *so that after a renaming of the bound variables,*

$$A_k \equiv \jmath(\ell(t_{b_1},\dots,t_{b_s})p_i(x_1,\dots,x_a), X_{b_1},\dots,X_{b_s}), \tag{31}$$

$$B_l \equiv \jmath(\ell(t_{b_1},\dots,t_{b_s})q_j(y_1,\dots,y_a), Y_{b_1},\dots,Y_{b_s}), \tag{32}$$

let Σ' *be the extension of* Σ *by the equation*

$$\ell(\vec{w}, t_{b_1},\dots,t_{b_s})\big(p_i(x_1,\dots,x_a)\{\vec{u}_k :\equiv \sigma\vec{w}\}\big) = \ell(\vec{w}, t_{b_1},\dots,t_{b_s})\big(q_j(y_1,\dots,y'_a)\{\vec{v}_l :\equiv \tau\vec{w}\}\big)$$

and also all equations

$$\ell(\vec{w})\big(X_c\{\vec{u}_k :\equiv \sigma\vec{w}\}\big) = \ell(\vec{w})\big(Y_c\{\vec{v}_l :\equiv \tau\vec{w}\}\big) \quad (c = b_1,\dots,b_s);$$

then Σ *is true in* $\mathfrak{M}$ *if and only if* Σ' *is true in* $\mathfrak{M}$.

Proof. (1) For (1a), notice that if p_k and q_l have different characters, then the functions $\overline{p}_k$ and $\overline{q}_l$ take values in disjoint sets, because $\mathfrak{M}$ is reasonably free (and using Lemma 9.6 if one of them has character cond). For (1b), the hypothesis of the Lemma implies that the identity

$$f_i(\sigma\vec{w}, \vec{\pi}) = f_i(\tau\vec{w}, \vec{\rho})$$

is true in $\mathfrak{M}$; but f_i is injective, and so this is not possible if the variable lists $\sigma\vec{w}$ and $\tau\vec{w}$ differ even in one place, because we could then assign distinct values to it (in m, which has at least two elements) and get back distinct values from f_i.

For parts (2) – (4), we need to show that if Σ is true, then so is Σ'.

(2) By the assumed truth of Σ and the fact that the mutual least fixed points satisfy their defining equations, we know that the identity

$$\mathsf{f}_i(\sigma\vec{w}, \pi'_1(\sigma\vec{w}),\dots) = \mathsf{f}_i(\tau\vec{w}, \rho'_1(\tau\vec{w}),\dots)$$

is true, and since f_i is injective, this means that for $j = 1,\dots,s$, $\pi'_j(\sigma\vec{w}) = \rho'_j(\tau\vec{w})$ is true.

(3) By hypothesis, $A_k(\sigma\vec{w}) = B_l(\tau\vec{w})$ is true, and by Lemma 9.6, the values of $C_1\{\vec{u}_k :\equiv \sigma\vec{w}\}$ and $D_1\{\vec{v}_l :\equiv \tau\vec{w}\}$ are not in $\{\bot, \mathrm{ff}, \mathrm{tt}\}$; now (2) of 9.2 implies that

$$C_j\{\vec{u}_k :\equiv \sigma\vec{w}\} = D_j\{\vec{v}_l :\equiv \tau\vec{w}\}$$

is true, for $j = 1,2,3$, which by abstraction gives the required result.

(4) In the notation of 9.4, the hypothesis insures that the equation

$$\big(\jmath(\ell(t_{b_1},\dots,t_{b_s})p_i(x_1,\dots,x_a), X_{b_1},\dots,X_{b_s})\big)\{\vec{u}_k :\equiv \sigma\vec{w}\} = \big(\jmath(\ell(t_{b_1},\dots,t_{b_s})q_j(y_1,\dots,y_a), Y_{b_1},\dots,Y_{b_s})\big)\{\vec{u}_k :\equiv \sigma\vec{w}\}$$

is true in $\mathfrak{M}$, and from this, the required result follows easily by the injectivity of $\jmath$. ⊣

PROOF OF 9.1. If A and B are simplified terms with distinct recursion variables $\vec{p}$ and $\vec{q}$, and if $\vec{z}$ is a list of the variables which occur free in either A or B, then

$$\begin{aligned} A &\equiv p_1(\sigma_h \vec{z}) \text{ where } \{p_1(\vec{u}_1) = A_1(\vec{u}_1), \dots, p_m(\vec{u}_m) = A_m(\vec{u}_m)\} \\ B &\equiv q_1(\tau_h \vec{z}) \text{ where } \{q_1(\vec{v}_1) = B_1(\vec{v}_1), \dots, q_n(\vec{v}_n) = B_n(\vec{v}_n)\} \end{aligned}$$

with suitable projections σ_h and τ_h, and the equation $A = B$ is true on a structure $\mathfrak{M}$ exactly when the closed preconnection

$$\Sigma_0 = \{\ell(\vec{z})p_1(\sigma_h \vec{z}) = \ell(\vec{z})q_1(\tau_h \vec{z})\} \tag{33}$$

is true in $\mathfrak{M}$. We will consider several preconnections between A and B, and we need the following, auxilliary notions.

(a) An equation $\ell(\vec{w})p(\sigma\vec{w}) = \ell(\vec{w})q(\tau\vec{w})$ between ℓ-terms is *tidy*, if every variable in the list $\vec{w}$ occurs either in $p(\sigma\vec{w})$ or in $q(\tau\vec{w})$, and a preconnection Σ is **tidy** if all its equations are tidy. It is clear that every equation is provably equivalent to a tidy one, and so the same is true of preconnections. The useful fact is that (up to alphabetic variance between terms) *there are only finitely many tidy preconnections between A and B*, with an upper bound on their number easily computable from the number and arities of the function variables $\vec{p}$ and $\vec{q}$.

(b) A preconnection Σ is **dead**, if some equation $\pi = \rho$ in it falls under Case (1) of Lemma 9.7.

(c) An equation $\ell(\vec{w})p_k(\sigma\vec{w}) = \ell(\vec{w})q_l(\tau\vec{w})$ in a preconnection Σ is **fulfilled**, if it does not fall under Case (1) of Lemma 9.7 and (briefly) $\Sigma' = \Sigma$, i.e., (the "tidyfication" of) every equation in the preconnection Σ' associated with Σ by the operative case (2) – (4) of Lemma 9.7 is an alphabetic variant of some equation in Σ. It is clear that *we can determine effectively whether* $\pi = \rho$ *is fulfilled in* Σ, since deleting empty abstractions and testing for alphabetic variance are effective operations.

We start now with Σ_0 in (33) and we apply Lemma 9.7 repeatedly to produce preconnections

$$\Sigma_0 \subsetneq \Sigma_1 \subsetneq \Sigma_2 \cdots \subsetneq \Sigma_e,$$

choosing a new unfulfilled identity at each stage, if one exists, and setting $\Sigma_{i+1} = \Sigma_i'$, as long as (1) of the Lemma does not apply to Σ_i; so, at each stage,

$$\Sigma_i \text{ is true } \iff \Sigma_{i+1} \text{ is true.} \tag{34}$$

Now, the procedure cannot continue indefinitely by (a) above, so that eventually we reach one of the following situations.

ALTERNATIVE 1. At Stage e, Σ_e is dead, and then by (34) Σ_0—and hence $A = B$—is not true, on every reasonably free structure.

ALTERNATIVE 2. Every equation in Σ_e is fulfilled, and then Σ_e is a connection between A and B.

The proof of this last fact is very simple now, but it is worth considering an example in Case (4) which involves the restricted form of β-reduction allowed in $\mathcal{E}_0$.

Suppose that $\ell(x,y)p_k(\sigma\langle x,y\rangle) = \ell(x,y)q_l(\tau\langle x,y\rangle)$ is in Σ_e, and that

$$\begin{aligned} A_k(\sigma_k\langle x,y\rangle) &\equiv p(p'(x,x),y,p''(y)) \\ &\equiv \jmath(\ell(t_1,t_2)p(t_1,y,t_2),p'(x,x),p''(y)), \\ B_l(\tau_l\langle x,y\rangle) &\equiv q(x,x,q'(y),y,q''(y,x)) \\ &\equiv \jmath(\ell(t_1,t_2)q(x,x,t_1,y,t_2),q'(y),q''(y,x)). \end{aligned}$$

We must show the equality of these two terms from Σ_e, which now contains by (4) the equations

$$\ell(x,y,t_1,t_2)p(t_1,y,t_2) = \ell(x,y,t_1,t_2)q(x,x,t_1,y,t_2), \tag{35}$$

$$\ell(x,y)p'(x,x) = \ell(x,y)q'(y) \tag{36}$$

$$\ell(x,y)p''(y) = \ell(x,y)q''(y,x). \tag{37}$$

From (35), by variable specification and abstraction, we get

$$\ell(t_1,t_2)p(t_1,y,t_2) = \ell(t_1,t_2)q(x,x,t_1,y,t_2),$$

which is a linear identity; and so by linear specification (FL6), we get

$$p(p'(x,x),y,p''(y)) = q(x,x,p'(x,x),y,p''(y)); \tag{38}$$

and then, from (36) and (37), using *5 of Lemma 8.1 we get

$$q(x,x,p'(x,x),y,p''(y)) = q(x,x,q'(y),y,q''(y,x))$$

which with (38) yields the required equation.

This completes the proof of the Main Theorem for simplified terms, and hence for arbitrary terms with no free function variables by Lemma 8.4. The result follows for arbitrary terms because $A(r) = B(r)$ is valid or provable exactly when $A(g) = B(g)$ is valid or provable, with g a fresh functional symbol, and then for equations between ℓ-terms, easily, by abstraction. $\dashv$

10. THE FULL SYSTEM $\mathcal{E}$

If we think of closed ℓ-terms as expressing programs, then the Main Theorem 9.1 says that the system $\mathcal{E}_0$ is complete for (denotational) *program equivalence.* But this is clearly not enough—we want to be able to prove more complex properties of programs than mere equivalence, and also to reason about programs from hypotheses. We describe here, briefly and without proofs, an extension of $\mathcal{E}_0$ which goes part-way towards achieving this.

10.1. FLR FORMULAS AND THEIR SEMANTICS. The **FLR($\boldsymbol{\tau}$)-formulas** are defined inductively by

$$\phi :\equiv \varepsilon \mid (\neg\phi) \mid (\phi_1 \mathbin{\&} \phi_2) \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \rightarrow \phi_2) \mid (\forall u)\phi \mid (\exists u)\phi \mid (\forall p)\phi \mid (\exists p)\phi,$$

where ε is an equation between FLR($\boldsymbol{\tau}$) expressions.

Formulas are interpreted on $\jmath$-structures of signature $\boldsymbol{\tau}$ in the obvious way. Notice that

$$(\forall p)[A(p) = B(p)] \not\models A(\ell(u,v)C(u,v)) = B(\ell(u,v)C(u,v)),$$

because in the stream structures of Section 5, the hypothesis is true and the conclusion fails when

$$A(p) \equiv p(a,b), B(p) = q(a) \text{ where } \{q(u) = p(u,b)\}, C(u,v) \equiv f(v,u).$$

However, obviously:

$$(\forall p)[A(p) = B(p)] \models A(p) \text{ where } \{p(\vec{u}) = C\} = B(p) \text{ where } \{p(\vec{u}) = C\},$$

even when p occurs in C, and we will need to include this among the axioms of $\mathcal{E}$, to replace the usual "instantantiation" property.

For each formula ϕ and terms $C_1, \dots, C_n$, the formula

$$\phi \text{ where } \{p_1(\vec{u}_1) = C_1, \dots, p_n(\vec{u}_n) = C_n\}$$

is constructed by "pushing" the recursion construct through the first-order operations (and $\ell(u)$) until it applies to FLR-terms, e.g.,

$$\begin{aligned}
&\big(A = B\big) \text{ where } \{p(\vec{u}) = C\} \equiv A \text{ where } \{p(\vec{u}) = C\} = B \text{ where } \{p(\vec{u}) = C\},\\
&\big(\ell(\vec{v})A = \ell(\vec{v})B\big) \text{ where } \{p(\vec{u}) = C\} \equiv (\forall \vec{v})[\big(A = B\big) \text{ where } \{p(\vec{u}) = C\}],\\
&\big(\neg\phi\big) \text{ where } \{p(\vec{u}) = C\} \equiv \neg\big(\phi \text{ where } \{p(\vec{u}) = C\}\big),
\end{aligned}$$

etc. It is clear that the denotation of $\phi \text{ where } \{p(\vec{u}) = C\}$ in a $\jmath$-structure can be computed by first determining the least-fixed-point $\overline{p}$ of $p(\vec{u}) = C$, and then evaluating ϕ for $p := \overline{p}$.

10.2. THE SYSTEN $\mathcal{E}$. **Sequents** (of formulas) are expressions of the form $\Gamma \Rightarrow \phi$, where $\Gamma \equiv \phi_1, \dots, \phi_n$ is now a *set of formulas*, so they include the sequents of equations of $\mathcal{E}_0$, and (again), the **explicit sequents** are those in which where does not occur.

To define the axioms and rules of $\mathcal{E}$, we consider (classical) three-sorted structures of the form

$$(M, W, \{\mathcal{P}_n \mid n \in \mathbb{N}\}, \{\text{ap}_n \mid n \in \mathbb{N}\}, \{\jmath^n \mid n \in \mathbb{N}\}, \{R_i \mid i \in I\}), \quad (39)$$

where M, W are arbitrary, non-empty sets; each $\mathcal{P}_n$ is a set of n-ary functions from M to W; each $\text{ap}_n : M \times \mathcal{P}_n \rightharpoonup W$ stands for function application; the imbeddings $\jmath_n$ are operations of the same kind as in FLR-structures, but without the monotonicity hypothesis (which makes no sense now); and each R_i is a relation on arguments from M and functions on M to W, like the functionals on $\jmath$-structures, but, again, with no monotonicity hypotheses. Let $\boldsymbol{\tau}^*$ be the first-order signature for such structures which has infinitely many relation symbols for each kind, and consider the usual (first-order, three-sorted) language L^*. The terms of L^* are made up from function variables and the ap_n primitives, they can be identified with the simple terms of FLR (with no functional constants) and they are interpreted on structures of the form (39) using the imbeddings $\jmath_n$ to interpret composition, as in FLR. We now let T^* be the theory of structures of the form (39) which have enough functions to interpret their terms, i.e., they satisfy the obvious axioms which assert that the p's vary over functions and (the closure of) $(\exists p)(\forall \vec{u})[p(\vec{u}) = t(\vec{u})]$ for every term $t(\vec{u})$ (which is not a basic variable). The Completeness Theorem for (many-sorted) first-order logic yields a complete axiomatization of T^*.

The **provable formula sequents** of $\mathcal{E}$ are defined by combining the axioms of $\mathcal{E}_0$ with a complete axiomatization of T^*, as follows:[13]

($\mathcal{E}1$) The axioms and rules of $\mathcal{E}_0$ in Table 2 are part of $\mathcal{E}$, re-interpreted so that Γ stands for a finite set of formulas.

($\mathcal{E}2$) If $\Gamma \Rightarrow \phi$ is a valid sequent of T^*, and if $\Gamma' \Rightarrow \phi'$ is obtained from $\Gamma \Rightarrow \phi$ by replacing each prime formula $R(\vec{u}, \vec{p})$ throughout by an FLR-equation $E = F$ with the same free variables, then $\Gamma' \vdash \phi'$.

($\mathcal{E}3$) $(\forall p)\phi(p) \Rightarrow \phi \,\mathsf{where}\, \{p(\vec{u}) = C\}$ for each formula $\phi(p)$ and each term C.

It is clear that $\mathcal{E}$ is **sound** for $\jmath$-semantics, and (of course) it is not complete, but we can easily formalize in it proofs which combine basic properties of recursive definitions with first-order arguments whose soundness depends only on the fact that the p's stand for functions on the universe M to some set. For example:

10.3. LEMMA. *For all terms* $A_1(\vec{u}_1), \dots, A_n(\vec{u}_n)$ *and each formula* $\phi(\vec{p})$*, if*

$$\Gamma, \text{ for } i = 1, \dots, n, (\forall \vec{u}_i)[p_i(\vec{u}_i) = A_i(\vec{u}_i)], \vdash \phi(\vec{p})$$

and no p_i *is free in* Γ*, then*

$$\Gamma \vdash \phi(\vec{p}) \,\mathsf{where}\, \{p_1(\vec{u}_1) = A_1(\vec{u}_1), \dots, p_n(\vec{u}_n) = A_n(\vec{u}_n)\}.$$

10.4. EXAMPLE. Suppose we define addition by some standard recursion

$$m + n \equiv p(m, n) \,\mathsf{where}\, \{p(m, n) = A(m, n, p)\}$$

on the structure of arithmetic (6), and we want to show that it is commutative, $m + n = n + m$, from the usual axioms—including induction. By induction (on first-order formulas),

$$(\forall m, n)[p(m, n) = A(m, n, p)] \vdash p(m, n) = p(n, m),$$

from which, by the Lemma,

$$p(m, n) \,\mathsf{where}\, \{p(m, n) = A(m, n, p)\} \\ = p(n, m) \,\mathsf{where}\, \{p(m, n) = A(m, n, p)\},$$

i.e., $m + n = n + m$.

Speaking a bit loosely, we can use the method of the Example to prove in $\mathcal{E}$ properties of recursive definitions which have unique fixed points, and that is what we usually expect from formal systems about recursion. The system is stronger than this, however, because it extends $\mathcal{E}_0$ which is already complete for (absolutely) valid FLR-equations, and so can verify many properties of recursions which do not have unique solutions.

[13]In a detailed exposition of this, one would naturally put down a specific set of axioms and rules, directly on FLR-formulas, and then prove the first-order completeness.

We end with an extension of the Main Theorem 9.1 which is still fairly weak (and none-too-elegant), but which suggests that there is more completeness here than meets the eye.

10.5. BASIC (FIRST-ORDER) THEORIES. A set of sentences T in FLR(τ) is a *basic* theory if:

(B1) Every function symbol f_i which occurs in a sentence of T has kind $\langle n\rangle$ (so that it stands for an n-ary function), and every θ in T is a classical, first order sentence—with no occurrences of $\mathtt{tt}$, $\mathtt{ff}$, $\ell(\vec{u})$ or conditionals, no where, and no function variables.

(B2) If f_i occurs in some sentence of T, then T also contains the sentence

$$(\forall\vec{u})(\exists v)[\mathsf{f}_i(\vec{u}) = v],$$

so that f_i is interpreted in $\jmath$-structures by a function $f_i : M^n \to M$.

The *basic terms* of T are the explicit (first order) terms constructed from the function symbols which occur in T, so that they will always be interpreted in the basic universe M of every $\jmath$-structure. The *proper poset terms* are those with the opposite property, which will always take values in $W \setminus M$: they are defined by a new clause to Table 1, similar to that for terms except that only symbols f_i which do not occur in T are allowed while the Z_is can be arbitrary terms. For example, $f(g(u), v)$ is a basic term if f and g occur in T and a proper, poset term if f does not occur in T; and $f(p(u))\ \mathsf{where}\ \{p(u) = g(u, p(u))$ is not a basic term, and it is a proper, poset term only if neither f nor g occurs in T.

For example, consider the stream structure

$$\mathfrak{S} = (\mathbb{N}, \mathbb{N} \cup \mathrm{Str}, \mathrm{ff}, \mathrm{tt}, \mathrm{cond}, 0, 1, +, \cdot, Z, ;, \{\langle a, 1\rangle \mid a \in A\}, \{f_i \mid i \in I\}),$$

where $(\mathbb{N}, 0, 1, +, \cdot, Z)$ is classical arithmetic, and let T be Peano arithmetic in just these symbols. The basic formulas of T express first-order properties of the integers, while the proper, poset terms describe "computations" which "necessarily" involve the execution of acts.

10.6. **Basic Completeness Theorem.** *The system $\mathcal{E}$ is complete for $\jmath$-valid sequents $\Gamma \Rightarrow A = B$, where Γ is a basic theory and A, B are proper, poset terms of* Γ.

This is proved very much like the Main Theorem 9.1, (roughly) replacing variables by basic terms throughout.

Department of Mathematics
University of California
Los Angeles

REFERENCES

1. A. J. C. Hurkens, Monica McArthur, Yiannis N. Moschovakis, Lawrence Moss, and Glen T. Whitney. The logic of recursive equations. in preparation.

2. Alexander Kechris and Yiannis N. Moschovakis. Recursion in higher types. In Jon Barwise, editor, *Handbook of mathematical logic*, number 90 in Studies in Logic, pages 681–737. North Holland, 1977.
3. Brian. W. Kernighan and M. Ritchie, Dennis. *The C programming language*. Prentice-Hall, Englewood Cliffs, New Jersey, 1978.
4. Yiannis N. Moschovakis. *Elementary induction on abstract structures*. Studies in Logic, No. 77. North Holland, Amsterdam, 1974.
5. Yiannis N. Moschovakis. The formal language of recursion. *Journal of Symbolic Logic*, 54:1216–1252, 1989.
6. Yiannis N. Moschovakis. A model of concurrency with fair merge and full recursion. *Information and Computation*, 93:114–171, 1991.
7. Yiannis N. Moschovakis. Computable concurrent processes. *Theoretical Computer Science*, 139:243–273, 1995.
8. Yiannis N. Moschovakis and Glen T. Whitney. Powerdomains, powerstructures and fairness. In L. Pacholski and J. Tiuryn, editors, *Computer Science Logic*, number 933 in Lecture Notes in Computer Science, pages 382–396, Berlin, 1995. Springer-Verlag.
9. D. Scott and Strachey. Towards a mathematical semantics for computer languages. In J. Fox, editor, *Proceedings of the Symposium on computers and automata*, pages 19–46, New York, 1971. Polytechnic Institute of Brooklyn Press.