

Research Report

COMPLETENESS OF REWRITE RULES AND REWRITE STRATEGIES FOR FP

Joseph Y. Halpern
John H. Williams
Edward L. Wimmers

IBM Almdaen Research Center
San Jose, CA 95120

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).



Research Division
Yorktown Heights, New York • San Jose, California • Zurich, Switzerland

Copies may be requested from:
IBM Thomas J. Watson Research Center
Distribution Services
Post Office Box 218
Yorktown Heights, New York 10598

Completeness of Rewrite Rules and Rewrite Strategies for FP

Joseph Y. Halpern
John H. Williams
Edward L. Wimmers

IBM Almaden Research Center
San Jose, CA 95120

Abstract: We consider languages whose operational semantics is given by a set of rewrite rules. For such languages, it is important to be able to determine that there are enough rules to be able to compute the correct meaning of all expressions, but not so many that the system of rules is inconsistent. We develop a formal framework in which to give a precise treatment of these completeness and soundness issues, and then investigate them in the context of an extended version of the functional programming language FP. We show that the rewrite rules of FP are sound and complete with respect to three different notions of completeness. In the latter half of the paper we turn our attention to rewrite strategies. In order to implement a language based on rewrite rules, it does not suffice to know that there are "enough" rules in the language; we also need to have a good strategy for determining the order in which to apply them. But what is "good"? Corresponding to each of our notions of completeness there is a notion of a good rewrite strategy. We examine and characterize these notions of goodness, and give examples of a number of natural good strategies. Although we have confined ourselves to FP here, we believe that our techniques (some of which are nontrivial extensions of techniques first used in the context of λ -calculus) will apply well beyond the realm of FP rewriting systems.

Recently there have appeared in the literature a number of programming languages where *rewrite* (or *reduction*) rules have been used to provide operational semantics or a means of proving program equivalence ([PI77,O'D77,Ba78,Tu79,BMS80,Bo82,Wi82,GMP82,HO84]). When rewrite rules are used in this way, there are two obvious questions that can be asked:

1. Do we have too many rules, thus making equivalent two expressions that should be different?
2. Do we have enough rules to be able to compute the correct meaning of all expressions?

In this paper we develop a formal framework in which to give precise answers to these soundness and completeness questions. We investigate these issues in the context of an extended version of the functional programming language FP [Ba78]; however, the applicability of these techniques extends far beyond the realm of FP rewriting systems.

Our first step is to give a formal denotational semantics to FP. This in itself is a nontrivial task. FP can be viewed as an untyped combinatory calculus, so we have the usual difficulties of giving semantics in such a way that every expression can be viewed as both a function and an argument to a function (cf. [Me82]), as well as the problems involved in representing infinite sequences.

Once we have given the semantics, answering the first question above becomes straightforward. We must simply show that any two expressions that we intend to be distinct are assigned different meanings, and then show that the reduction rules preserve meaning. Thus if $x \rightarrow y$ is a reduction rule, we must show that $\mu(x) = \mu(y)$, where μ is the semantic function that assigns a meaning to each FP expression.

Proving completeness is a somewhat more delicate task. Part of the problem is that it is not even clear what it means for there to be “enough” rules. We might hope to be able to prove that whenever two FP expressions x and y have the same meaning (i.e., $\mu(x) = \mu(y)$) it is possible to show that there exists some z such that both x and y reduce to z . This turns out to be too much to expect. Just as for combinatory calculus, we know that all recursive functions can be encoded in FP [Wi81], so doing this would be at least as hard as proving the equality of two recursive functions, a problem well known not to be recursively enumerable (cf. [Ro67]). Thus we must aim for somewhat weaker notions of completeness.

The first notion of completeness we consider depends on the observation that in FP, although programmers can write higher-order functions, programs are usually written to compute *observable expressions*, where an observable expression is one that can be formed from atoms using the sequence constructor. Roughly speaking, what we will show is that if the meaning of an FP expression x is the same as that of an observable expression y , then we have enough rewrite rules to reduce x to y . We call this type of completeness *observable completeness*.

completeness and the corresponding notion for rewrite strategies, L-safety. We characterize L-safety in terms of FP', and then use that characterization to show that S-safety implies L-safety. We conclude in Section 8 with some directions for future work and some remarks on the Church-Rosser property for FP. We have written the paper with two audiences in mind: the researcher in the field of programming semantics and the casual reader with little

background in the field of programming semantics. We have therefore attempted to make the paper completely self-contained, and have deferred many of the more technical proofs to the appendices.

2. Syntax and Semantics

Our language is the result of a desire to extend the language of [Ba78] to allow the use of infinite sequences, lazy evaluation, and programmer defined functionals, and at the same time to preserve as much of the original flavor of FP as possible.

To accommodate these extensions, some modifications to FP are necessary; the primary differences between the language of [Ba78] (here denoted FP78) and our FP are:

1. We allow expressions whose meaning is a sequence, some of whose elements are undefined; i.e., the sequence constructor is no longer required to be strict. This enables us to realize normal order evaluation and laziness (see [FW76, HM76] for further discussion of the advantages of lazy evaluation).
2. We allow expressions whose meaning is an infinite sequence of elements. Thus, in FP78 it is not possible to write an expression whose meaning is the sequence of all positive integers, but in ours it is. Since FP78 sequences are symmetric in the sense that elements can be appended and accessed from either end, the design desideratum of preserving as much of FP78 as possible has led to the notion of (potentially) infinite sequences that are manipulable from either end.
3. We use a conditional *function* rather than a conditional *functional* as our if-then-else construct. This wasn't possible in FP78 since all functions were strict. We employ it to simplify the rewrite rules for conditionals.
4. We allow the definition of new functionals, thus allowing the programmer to define his own combining forms. This is accomplished by making the application of combining forms explicit rather than by using special syntactic forms. Thus, the expression $[f_1, \dots, f_n]$ of [Ba78] becomes $\text{cons}: \langle f_1, \dots, f_n \rangle$ in our treatment. (We remark that the variant FFP ([Ba78]) also allows for programmer defined combining forms. However, it employs the Lisp-like device of using objects to represent functions, resulting in a non-denotational semantics; i.e., the meaning of $f_1:f_2$ cannot be described as the meaning of f_1 applied to the meaning of f_2 .)

Note that this last modification results in all functions being expressed in a prefix notation which very quickly becomes difficult to read, but it allows us to avoid nonessential details in our mathematical analysis. It would be well to view our definitions as an abstract syntax for FP; indeed, when we need to write example programs of any complexity, we use the infix forms of FP78 as abbreviations for our strict, prefix expressions.

We stress that the language we describe here is not intended to be taken as a convenient or expressive language for writing programs. Few of the usual primitive functions that

provide programming ease and power are included; in fact, we do not even have the basic arithmetic operations. To include them would only clutter our proofs with uninteresting details and trivial case analyses, without adding any conceptually different mathematical considerations. In the following definition we present a language that contains the mathematically interesting and challenging essence of extended FP when considered from the viewpoint of rewrite rule soundness and completeness.

Syntax. We assume the existence of a set $\mathcal{A}$ of *atoms* which includes the integers and strings of letters. We require that T and F be atoms. We also assume the existence of a set $\mathcal{F}$ of primitive functions that is disjoint from $\mathcal{A}$. The set $\mathcal{F}$ we consider in this paper is {al, first, tl, ar, last, tr, null, comp, cons, apply, K, id, cond}. This set of functions is sufficiently rich to allow us to encode all recursive functions.

Definition 2.1. The *FP expressions* are the least set containing $\mathcal{A} \cup \mathcal{F}$ and closed under:

1. The formation of sequences: if $x_1, \dots, x_n$ are FP expressions (for $n \geq 0$), then $\langle x_1, \dots, x_n \rangle$ is also an FP expression.
2. Application of expressions: if x_1 and x_2 are FP expressions, then $(x_1:x_2)$ is also an FP expression.

As usual, we include parentheses only for the sake of clarity and assume that application associates to the left, so that we write $x:y:z$ rather than $(x:y):z$. The following are examples of FP expressions: a , $\langle a, \langle b \rangle \rangle$, $\text{null}:\langle \rangle$, $\text{comp}:\langle \text{first}, \text{tl} \rangle : (\text{al}:\langle a, \langle b \rangle \rangle)$, $\text{comp}:\text{ar}$, and $a:\text{tl}$.

Semantics. We provide a denotational semantics for FP as usual by defining a domain D^* and a mapping μ from FP expressions to D^* . Before doing so, however, we give a brief intuitive description of the intended meaning of each of the primitive functions and of our treatment of infinite objects. In this informal description of the primitives we anticipate the definition of μ by using “ $\mu(x) = \mu(y)$ ” as an abbreviation for “the meaning of x is the same as the meaning of y .”

1. **al** takes a pair as argument and forms a new sequence by adding the first argument to the left of the second argument (if the second argument is a sequence); thus $\mu(\text{al}:\langle x, \langle y_1, \dots, y_n \rangle \rangle) = \mu(\langle x, y_1, \dots, y_n \rangle)$.
2. **first** takes a sequence as argument and returns the first element of the sequence; thus $\mu(\text{first}:\langle x_1, \dots, x_n \rangle) = \mu(x_1)$.
3. **tl** removes the first element of a sequence, so $\mu(\text{tl}:\langle x_1, \dots, x_n \rangle) = \mu(\langle x_2, \dots, x_n \rangle)$.
4. There are symmetric operations **ar**, **last**, and **tr** which operate on the right side of the sequence instead of the left in an analogous manner to **al**, **first**, and **tl**.
5. **null** returns T when given the empty sequence and F when given a non-empty sequence or an atom. Thus, for example, $\mu(\text{null}:\langle \rangle) = \mu(T)$ and $\mu(\text{null}:\langle x_1, \dots, x_n \rangle) = \mu(F)$.

6. **cons** takes a sequence of functions $\langle f_1, \dots, f_n \rangle$ as an argument and returns the function that, when applied to x , returns the sequence $\langle f_1:x, \dots, f_n:x \rangle$. Thus $\mu((\text{cons}: \langle f_1, \dots, f_n \rangle):x) = \mu(\langle f_1:x, \dots, f_n:x \rangle)$. (We sometimes use the expression $[f_1, \dots, f_n]$ of [Ba78] as an abbreviation for $\text{cons}: \langle f_1, \dots, f_n \rangle$.)
7. **cond** is the if-then-else function, so $\mu(\text{cond}: \langle T, x, y \rangle) = \mu(x)$ and $\mu(\text{cond}: \langle F, x, y \rangle) = \mu(y)$.
8. **comp** denotes composition, so $\mu(\text{comp}: \langle f, g \rangle : x) = \mu(f:(g:x))$. (Again, we view $f \circ g$ as an abbreviation for $\text{comp}: \langle f, g \rangle$.)
9. **apply**, when given a pair of elements, applies the first to the second; thus $\mu(\text{apply}: \langle f, x \rangle) = \mu(f:x)$.
10. **K** is the familiar combinator from combinatory calculus that converts an expression x to a constant function that always returns x ; thus $\mu((K:x):y) = \mu(x)$.
11. **id** is the identity function, so $\mu(\text{id}:x) = \mu(x)$.

In order to accomplish the modifications mentioned above and to define the primitives formally, we require a richer semantic domain than that of FP78. For example, to incorporate laziness, the semantics of **al** and **first** must be such that an entire sequence need not be evaluated before its first element can be accessed; i.e., we want $\mu(\text{first}:(\text{al}: \langle x, y \rangle)) = \mu(x)$ to hold for all FP expressions x and y . Therefore there must be some element z in the domain such that $\text{al}: \langle a, \perp \rangle = z$ and $\text{first}:z = a$, where $\mu(\text{al}) = \text{al}$, $\mu(\text{first}) = \text{first}$, and $\mu(a) = a$.

We also wish to write expressions whose meanings are infinite sequences. For example, we want $f:a$ to have the meaning $\langle a, a, \dots \rangle$, where f is the recursive function $f = \text{al} \circ [\text{id}, f]$. (Note that although our syntax doesn't provide for such recursive definitions, it is possible to write the **fix** function in terms of our primitives, where the meaning of $\text{fix}:g$ is the same as the meaning of $g:(\text{fix}:g)$, cf. [Wi81]. Thus for f we can take the expression $\text{fix}:(\text{comp} \circ [K:\text{al}, \text{cons} \circ [K:\text{id}, \text{id}]])$. Again, we view $f = \text{al} \circ [\text{id}, f]$ as an abbreviation for this expression.) Thus the semantic domain must include such items as the infinite sequence of all a 's.

We handle both of these extensions by including sequences of the form $\langle x_1, \dots, x_m, *, x_{m+1}, \dots, x_n \rangle$ in the domain where, intuitively, $*$ is a place-holder for some (as yet unknown) sequence of items. Roughly speaking, we will take the meaning of the expression $f:a$ defined above to be the "limit" or sup of the sequence $\langle a, * \rangle, \langle a, a, * \rangle, \dots$. Similarly, we take the meaning of $\text{al}: \langle a, \perp \rangle$ to be $\langle a, * \rangle$. Note that $*$ acts quite differently from $\perp$; $*$ stands for a list of items yet to be computed, while $\perp$ stands for a single item yet to be computed. Thus, as we shall see, $\langle a, * \rangle \leq \langle a, a, a \rangle$, while $\langle a, \perp \rangle$ and $\langle a, a, a \rangle$ are incomparable.

We will construct a domain D^* that satisfies the domain equation $D^* = \mathcal{A} + \text{Seqs}(D^*) + (D^* \rightarrow D^*)$ where $\text{Seqs}(D^*)$ contains all sequences (including infinite sequences) whose elements are members of D^* . More precisely, every element of $\text{Seqs}(D^*)$ will have one of the five forms (where all x_i and y_i are members of D^*): $\langle x_1, \dots, x_n \rangle$,

$\langle x_1, \dots, x_m, *, y_1, \dots, y_n \rangle$, $\langle x_1, \dots, x_m, \dots, *, y_1, \dots, y_n \rangle$ (an infinite number of x_i 's), $\langle x_1, \dots, x_m, *, \dots, y_n, \dots, y_1 \rangle$ (an infinite number of y_i 's), or $\langle x_1, \dots, x_m, \dots, *, \dots, y_n, \dots, y_1 \rangle$ (an infinite number of both x_i 's and y_i 's). To construct a suitable domain we would like to be able simply to give the domain equation and then rely upon the well-developed body of work on denotational semantics to guarantee us that its solution would serve as the desired domain ([St77, SP82]). Unfortunately, because of the two-sided nature of our sequences, no solution to the above equation is possible using the standard constructors $+$, $\times$, and $\rightarrow$. For, consider the set of partial sequences $\{\langle a, * \rangle, \langle *, a \rangle\}$. Both $\langle a \rangle$ and $\langle a, *, a \rangle$ are minimal upper bounds for this set, but since they are incomparable this set has no *least* upper bound. But we know ([SP82]) that any solution to a domain equation built with the constructors $+$, $\times$, and $\rightarrow$ has the property that any set in the domain that has an upper bound *does* have a least upper bound. Nickl ([Ni85]) encountered a similar difficulty in attempting to provide a denotational semantics for a somewhat different approach to extending FP ([DM84]). She started with standard constructions but then was forced to make some special *ad hoc* modifications to obtain the proper domain.

We choose to begin from basics, both because we feel that doing so results in a semantic domain that is simpler and more easily understood by the programmer, and because by doing so we can illuminate the difference between this approach and others and develop useful terminology for our subsequent completeness proofs. Our approach is of course very similar to ([St77, Wa77]) in that we define an ascending chain of domains each with a $\leq$ relation and close off under a limiting process. However, we are able to avoid the use of inverse limits. In order to describe our domains, we recall a few standard technical definitions.

Definition 2.2. A *domain* $(D, \leq)$ consists of a set D and a reflexive, transitive ordering $\leq$. Intuitively, $x \leq y$ if y has at least as much "information" as x . (Note that we do not require that $\leq$ be anti-symmetric, so that it is not in general a partial order. Thus we may have distinct elements x and x' with $x \leq x'$ and $x' \leq x$; in this case we write $x \equiv x'$.) A subset X of D is *directed* iff $x_1, x_2 \in X$ implies that there exists $x_3 \in X$ such that $x_1 \leq x_3$ and $x_2 \leq x_3$. $(D, \leq)$ is *complete* iff every directed subset of D has a least upper bound in D . If X is a directed set with a least upper bound x , we denote this by $x \in \sqcup X$. Note there may be more than one least upper bound for X , since $\leq$ is not necessarily anti-symmetric. (However, if both x and x' are least upper bounds of X , then we must have $x \equiv x'$.) If $(D, \leq)$ and $(D', \leq')$ are two domains and $f: D \rightarrow D'$, then f is said to be *monotonic* iff, whenever $d_1, d_2 \in D$ such that $d_1 \leq d_2$, then $f(d_1) \leq' f(d_2)$; f is *continuous* iff, whenever X is a directed subset of D with a least upper bound x' , then $\{f(x) \mid x \in X\}$ is a directed subset of D' and has $f(x')$ as a least upper bound. Note that a continuous function must be monotonic. x is a *finite element* of D iff whenever $x \leq y$ and $y \in \sqcup Y$ for some nonempty directed subset Y of D , then $x \leq y'$ for some $y' \in Y$.

We now inductively define a sequence of domains $(D_0, \leq_0), (D_1, \leq_1), \dots$ so that $D_n \subseteq D_{n+1}$ and $\leq_n$ is the restriction of $\leq_{n+1}$ to D_n . In addition, we define auxiliary functions $P_m^n: D_n \rightarrow D_m$ for $0 \leq m \leq n$, and $\cdot_n$ mapping $D_n \times D_{n-1}$ into D_n for $1 \leq n$. Intuitively, P_m^n projects elements of D_n to D_m , so that $P_m^n(x)$ is the best approximation to x in D_m in a sense to be made precise below, while $f \cdot_n x$ is the result of applying f to x (although $\cdot_n$ must be defined even in cases where f is not a function). (Note we write $f \cdot_n x$, using infix notation, rather than $\cdot_n(f, x)$.) These domains will serve as steppingstones in the construction of D^* , the domain in which we will be ultimately interested; D^* will be defined as the completion of their union.

The elements of D_n , $n = 0, 1, \dots$ are called *items*. D_0 consists of $\perp$ and, for each atom $a \in \mathcal{A}$, a corresponding *atomic item* a . (Note that we have adopted the notational convention of using boldface for syntactic entities and italics for their semantic counterparts.) For $x, y \in D_0$, we define $x \leq_0 y$ iff $x = \perp$ or $x = y$. Finally, we define $P_0^0(x) = x$ for $x \in D_0$.

Suppose we have defined $(D_0, \leq_0), \dots, (D_N, \leq_N)$, P_m^n , $m \leq n \leq N$, and $\cdot_n$, $n \leq N$, such that the following seven conditions all hold. Using clauses 1-4, we then define D_{N+1} , P_m^{N+1} , $\cdot_{N+1}$, and $\leq_{N+1}$. The other clauses are useful as induction hypotheses to make the construction go through.

1. For $1 \leq n \leq N$, D_n consists of all the items in D_{n-1} together with all items of the form:
 - a. $\langle x_1, \dots, x_k \rangle$, where $\{x_1, \dots, x_k\} \subseteq D_{n-1} \cup \{*\}$ and at most one $x_i = *$ (where $*$ is a distinct symbol not equal to any item); if some $x_i = *$ then we must have $k \geq 2$. Such an item is called a *sequence item*, and if no $x_i = *$, it is further called a *sequence (item) of definite length*.
 - b. f , where f is a continuous function from D_{n-1} into itself. Such domain elements are called *function items*. The set of all such function is denoted $(D_{n-1} \rightarrow D_{n-1})$.
2. For $1 \leq m \leq n \leq N$, the projection functions P_m^n satisfy:

$$P_0^n(x) = \begin{cases} x & \text{if } x \text{ is an atom} \\ \perp & \text{otherwise} \end{cases}$$

$$P_m^n(x) = x, \text{ if } x \text{ is an atom or } \perp$$

$$P_m^n(\langle x_1, \dots, x_k \rangle) = \langle P_{m-1}^{n-1}(x_1), \dots, P_{m-1}^{n-1}(x_k) \rangle \quad (\text{where we take } P_{m-1}^{n-1}(*) = *)$$

$$P_m^n(f) = \begin{cases} f & \text{if } f \in (D_k \rightarrow D_k), k < m \\ g & \text{if } f \in (D_k \rightarrow D_k), k \geq m \end{cases}$$

where $g \in (D_{m-1} \rightarrow D_{m-1})$ is defined by $g(x) = P_{m-1}^{n-1}(f(x))$.

3. For $n \leq N$, we have

$$f \cdot_n x = \begin{cases} f(P_k^{n-1}(x)) & \text{if } f \in (D_k \rightarrow D_k) \\ \perp & \text{otherwise.} \end{cases}$$

4. For $1 \leq n \leq N$, $\leq_n$ is the least reflexive, transitive relation on D_n satisfying:

- a. $\perp \leq_n x$ for any item $x \in D_n$,
- b. $\langle x_1, \dots, x_l \rangle \leq_n \langle y_1, \dots, y_l \rangle$ if $x_j \leq_{n-1} y_j$ or $x_j = * = y_j$, for $j = 1, \dots, l$,

- c. $\langle \alpha_1, \dots, \alpha_l, *, y_1, \dots, y_m \rangle \leq_n \langle \alpha_1, \dots, \alpha_l, z_1, \dots, z_k, y_1, \dots, y_m \rangle$ for $k \geq 0$,
 - d. $f \leq_n g$ if $\forall x \in D_{n-1} (f \cdot_n x \leq_{n-1} g \cdot_n x)$, where f and g are function items.
5. For $n \leq N$, if X is a bounded, directed subset of D_n , then there is an $x \in D_n$ that is a least upper bound (with respect to D_n) of X .
 6. For $1 \leq n \leq N$, $\cdot_n$ is continuous in each of its arguments. Thus, if X, Y are directed sets, $X \cup \{x\} \subseteq D_n$, $Y \cup \{y\} \subseteq D_{n-1}$, $x \in \sqcup X$, $y \in \sqcup Y$, then for all $z \in D_{n-1}$, $x \cdot_n z$ is a least upper bound of the directed set $\{x' \cdot_n z \mid x' \in X\}$ and for all $z' \in D_n$, $z' \cdot_n y$ is a least upper bound of the directed set $\{z' \cdot_n y' \mid y' \in Y\}$.
 7. P_m^n is a continuous function for $m \leq n \leq N$.

We now define D_{N+1} using clause 1 above (with n replaced by $N+1$ of course), P_m^{N+1} for $m \leq N+1$ using clause 2, $\cdot_{N+1}$ using clause 3, and $\leq_{N+1}$ using clause 4. We must check that P_m^{N+1} is well defined; in particular, we must check that if $f \in (D_k \rightarrow D_k)$, $k \geq m$, then $P_m^{N+1}(f)$ as defined is indeed a continuous function, and so is an element of $(D_{m-1} \rightarrow D_{m-1})$. But since in this case $(P_m^{N+1}(f))(x) = P_{m-1}^N(f(x))$, the result follows from the continuity of f and P_{m-1}^N . By definition, clauses 1-4 hold; all that remains is to check that clauses 5-7 hold. This requires some work; the details can be found in Appendix A.

This completes the definition of our chain of domains. A few remarks are now in order. Note that in clause 1 above we explicitly disallow $\langle * \rangle$ as an item; we identify it with $\perp$. We could have included it and treated it as the unique minimal sequence item, but we chose not to, since there is no primitive that is capable of distinguishing it from $\perp$. The reader will also notice that at the "top level" of a sequence item we allow only one $*$. This is all that we require since, as will be seen when we give the meanings of the primitive functions, we have chosen to allow "access" to a sequence only from the left or right hand sides (via first and last). Therefore, we can safely identify such sequences as $\langle \alpha, *, y \rangle$ and $\langle \alpha, *, z, *, y \rangle$ (since the z between the $*$'s in the latter sequence can never be accessed).

The definition of $\leq_n$ guarantees that function items, sequence items, and atomic items are pairwise incomparable. By clause 5, we also know that *bounded* directed sets in D_n have least upper bounds in D_n . However, $(D_n, \leq_n)$ is not complete if $n \geq 1$; for example, the directed set $\{\langle a, * \rangle, \langle a, a, * \rangle, \langle a, a, a, * \rangle, \dots\}$ has no least upper bound in D_n , for all $n \geq 1$. We remark that bounded sets of items that are not directed also do not necessarily have least upper bounds. For example, consider $\{\langle a, * \rangle, \langle *, a \rangle\}$. This subset of D_1 is bounded by both $\langle a \rangle$ and $\langle a, *, a \rangle$, but these two items are incomparable. It is easy to check that both of these items are minimal upper bounds and hence there is no *least* upper bound.

Also note $\leq_n$ is not a partial order since it is not antisymmetric. For example, if f is the function in D_3 that maps all elements in D_2 into a and g is the function in D_5 that maps all elements of D_4 into a , then $f \equiv_5 g$. (This is why we have been careful to talk about *a* least upper bound rather than *the* least upper bound; upper bounds are not necessarily unique.)

It is easy to see that clause 1 guarantees that $D_n \subseteq D_{n+1}$. Similarly, clauses 2-4 guarantee that P_k^n is the restriction of P_{m+1}^{n+1} to D_n , $\leq_n$ is the restriction of $\leq_{n+1}$ to D_n , and $\cdot_n$ is the restriction of $\cdot_{n+1}$ to $D_n \times D_{n-1}$. It is thus natural to consider the union of this chain $(D, \leq_I)$ (the I is for semantic *item*) defined by $D = \bigcup_n D_n$ and $\leq_I = \bigcup_n \leq_n$. Similarly, we can define $P_m: D \rightarrow D_m$ as the union (over n) of the P_m^n 's and define $\cdot$ mapping $D \times D$ to D as the union of the $\cdot_n$'s.

Lemma 2.3. *The projection functions satisfy the following properties.*

1. $P_k(x) = x$ if $x \in D_k$
2. $P_k(P_m(x)) = P_{\min(k,m)}(x)$
3. $P_{m+1}(f) \cdot x = P_m(f \cdot P_m(x))$
4. $P_k(x) \leq_I x$
5. $P_k(x)$ is the best approximation to x in D_k ; i.e., if $y \leq_I x$ and $y \in D_k$, then $y \leq_I P_k(x)$.

Proof. Parts 1, 2, 4, and 5 follow by a straightforward induction on k , with a subinduction on the structure of x , while part 3 follows from the definition of $\cdot$. $\square$

The domain D has a crucial property of which we make essential use in the sequel. Let Fin_n be the set of finite elements in D_n , and let $Fin = \bigcup_n Fin_n$. The following proposition shows that every element of D_n is the limit of its finite approximations.

Proposition 2.4. *Suppose $x \in D_n$. Then $Fin_n(x) = \{x' \leq_n x \mid x' \in Fin_n\}$ is directed, and $x \in \sqcup Fin_n(x)$.*

Proof. See Appendix B.

In the proof of Proposition 2.4 we also give a complete characterization of the finite elements of D_n . It easily follows from the characterization that if $x \in Fin_{n'}$ and $n' \leq n$, then $x \in Fin_n$. It is interesting to note that an element of Fin_n is *not* necessarily a finite element of D . The finite elements of D form a strict subset of Fin . We give an example in Appendix B (see Example 10.3) of an element $h \in Fin_2$ and a directed subset Y of D (which of course is not a subset of D_n for any n) such that h is a least upper bound of Y in D , but $h \not\leq_I y$ for all $y \in Y$. Related examples show that D does not have some of the nice properties enjoyed by the D_n 's. In particular, we extend Example 10.3 to show that $\cdot$ is not continuous in its first argument, while Example 10.4 shows that in contrast to D_n , bounded, directed subsets of D do not necessarily have least upper bounds.

These examples suggest that the domain D we have constructed, while it has many useful properties, does not suffice as a semantic domain for FP. Its most important drawback from our point of view is that it is not complete (since, for example, it does not have infinite sequences). D contains arbitrarily good approximations to infinite sequences (and to infinitely nested sequences), but it does not contain the infinite sequences themselves. Thus we consider the *completion* of D , which intuitively bears the same relation to the D as the real numbers bear to the rationals. The idea is to add to D all least upper bounds of directed sequences.

Our approach conveniently identifies elements that are indistinguishable under $\leq_I$, so that the resulting domain is in fact a partial order. To make our construction precise, we need a few more technical definitions.

Definition 2.5. Let X be a subset of D . $X^d = \{x \mid x \leq_I y, y \in X\}$ is called the *downward closure* of X . $X^l = \{x \mid \exists n, X' (x \in \sqcup X', X' \subseteq X \cap D_n, \text{ and } X' \text{ is directed})\}$ is called the *limit closure* of X . X is *downward closed* iff $X = X^d$, and X is *limit closed* iff $X = X^l$. We abbreviate $(X^d)^l$ by X^{dl} .

We now define the semantic domain $(D^*, \sqsubseteq)$ by taking

$$D^* = \{X \subseteq D \mid X \text{ is nonempty, downward closed, limit closed, and directed}\}.$$

Note that elements in D^* , which we again call items, are ordered by the subset relation, so that D^* is a partial order. In particular, this means we can write $X = \sqcup \mathcal{X}$ to denote that X is the least upper bound of the directed set $\mathcal{X}$, since least upper bounds are now unique. We can essentially view D as a subset of D^* by identifying the item $x \in D$ with the downward closure of $\{x\}$, (although, since $\leq_I$ is not antisymmetric, two distinct elements of D may be identified with the same element of D^*). Although, as we mentioned above, the finite elements of D form a strict subset of Fin , it is easy to show that the finite elements of D^* are precisely the elements of Fin (viewed as elements of D^*).

If $X_1, \dots, X_n$ are elements of D^* , then we can view $\langle X_1, \dots, X_n \rangle$ as an element of D^* by identifying it with $\{\langle y_1, \dots, y_n \rangle \mid y_i \in X_i\}^{dl}$ (this set is an element of D^* by Proposition 2.6). Thus we can view D^* as closed under the formation of sequences. We extend the terminology for D to D^* by saying that an item X in D^* is an atom (resp. sequence item, function item) iff one of the elements of X is an atom (resp. sequence item, function item). Since X is directed, this terminology is well defined.

We would like to extend the $:$ operation to D^* . If $X, Y \in D^*$ it may seem that the obvious candidate for $X:Y$ is $\{x:y \mid x \in X, y \in Y\}$. While it can be shown that this set is directed and downward closed, in general it is *not* limit closed, and so may not be an element of D^* ! (See Example 11.7 in Appendix C.) Thus, we define $:$ on $D^* \times D^*$ by

$$X:Y = \{x:y \mid x \in X, y \in Y\}^{dl}.$$

In order to show that $X:Y$ is an element of D^* , we need

Proposition 2.6. *If X is a directed subset of D , then $X^{dl} \in D^*$.*

Proof. See Appendix C. $\square$

Corollary 2.7. *If $X, Y \in D^*$, then $X:Y \in D^*$.*

Corollary 2.8. *D^* is a cpo (complete partial order). Moreover, if $\mathcal{X}$ is a directed subset of D^* , then $\sqcup \mathcal{X} = (\cup_{X \in \mathcal{X}} X)^{dl}$.*

Proof. Clearly D^* is partially ordered by $\sqsubseteq$. Let $\mathcal{X}$ be a directed subset of D^* . Note that if $X_1, X_2 \in \mathcal{X}$, then, because $\mathcal{X}$ is directed, there must be some $X_3 \in \mathcal{X}$ such that $X_1 \sqsubseteq X_3$ and $X_2 \sqsubseteq X_3$. Let $Y = \cup_{X \in \mathcal{X}} X$. If $y, y' \in Y$, then (by the observation) for some $X \in \mathcal{X}$, we must have

$y, y' \in X$. Since X is directed, there must be some $y'' \in X$ such that $y, y' \leq_I y''$. Since $y'' \in Y$ by construction, Y is directed. By Proposition 2.6, it follows that $Y^{dl} \in D^*$. It is clear that Y^{dl} is the least upper bound of $\mathcal{X}$ in D^* . $\square$

Since in many instances, an element of D^* is presented in the form X^{dl} where X is a directed subset of D , it is very useful to have a proposition that tells us how to apply $\cdot$ in terms of the underlying sets. The next proposition gives us just such a technique.

Lemma 2.9. *If X and Y are directed subsets of D , then $X^{dl} \cdot Y^{dl} = \{x \cdot y \mid x \in X, y \in Y\}^{dl}$.*

Proof. See Appendix D. $\square$

Finally, we justify our definition of application ($\cdot$) in D^* by showing that it is continuous in each of its arguments and that it makes our function items *extensional*; i.e., if two function items give the same result when applied to all elements in D^* , then they must be equal.

Proposition 2.10.

1. $\cdot$ as defined on D^* is continuous in each of its arguments.
2. D^* is isomorphic to $\mathcal{A} + \text{Seqs}(D^*) + (D^* \rightarrow D^*)$.
3. If $F, G \in D^*$ are function items and $F \cdot X = G \cdot X$ for all $X \in D^*$, then $F = G$.

Proof. See Appendix D. $\square$

We remark that we required elements of D^* to be limit closed precisely so that we could get extensionality for function items. Without this requirement, extensionality would not hold. For example, for $1 \leq n \leq \infty$, let $h_n(x) = T$ if for some $m < n$, x is a sequence of definite length m , and $h_n(x) = \perp$ otherwise. Then $\{h_n \mid n \in \mathbb{N}\}^d$ acts the same (as a function) as $\{h_\infty\}^d$, yet they are different sets.

We are now ready to give a meaning in the semantic domain D^* to each syntactic expression. We define such a meaning function μ from syntactic expressions to D^* as follows:

$$\begin{aligned} \mu(a) &= \{a\}^d \text{ for any atom } a \\ \mu(\langle x_1, \dots, x_n \rangle) &= \langle \mu(x_1), \dots, \mu(x_n) \rangle \\ \mu(x \cdot y) &= \mu(x) \cdot \mu(y). \end{aligned}$$

All that remains is to define $\mu(f)$ for each primitive function symbol f . Our first step in doing so is to define approximating functions f_{n+1} for each primitive function symbol f by restricting its domain to D_n in such a way that $P_m(f_n) = f_m$ for $m \leq n$. For example, al_{n+1} is that function in $(D_n \rightarrow D_n)$, which, when applied to an element in D_n of the proper form, yields the proper result (eg, $al_3 \cdot \langle a, \langle b, c \rangle \rangle = \langle a, b, c \rangle$) and when applied to anything else yields $\perp$. We then use these approximations to define the full meaning in D^* of each primitive.

In presenting these approximating functions, we use the notation

$$f_{n+1} \cdot \begin{cases} x_1 = y_1 & \text{if } p_1 \\ \vdots & \vdots \\ x_m = y_m & \text{if } p_m \end{cases}$$

to stand for the *ordered* conditional definition, “ f_{n+1} yields y_1 if applied to an argument (in D_n) of the form x_1 and the predicate p_i is true, otherwise ..., otherwise f_{n+1} yields y_m if applied to an argument (in D_n) of the form x_m and the predicate p_m is true, otherwise f_{n+1} yields $\perp$.”

Using this notation, the definition of al_{n+1} is

$$al_{n+1} : \begin{cases} \langle x, \langle y_1, \dots, y_m \rangle \rangle = \langle x, y_1, \dots, y_m \rangle & \text{if } x \neq * \\ \langle x, y \rangle = \langle x, * \rangle & \text{if } x \neq * \neq y \end{cases}$$

which says that any item x appended to the left of a sequence item yields the usual result, and appended to the left of a non-sequence item (such as an atom, a function, or $\perp$) yields $\langle x, * \rangle$. We note in passing that there are several interesting design decisions embodied in this definition for al . This specification is the least monotonic function consistent with the requirement that $al : \langle a, \perp \rangle = \langle a, * \rangle$ as discussed earlier. Thus, $al : \langle a, b \rangle$ can no longer be an error as in FP78, but must be at least as defined as $\langle a, * \rangle$. Many other choices are possible, such as $al : \langle a, b \rangle = \langle a, b \rangle$, and result in computationally interesting variations on the language we present. Since our interest here is the mathematical treatment of rewriting rules, we refrain from pursuing such considerations and leave them to be described in a forthcoming paper which will concentrate on programming language design issues.

We have already seen the definition of al_{n+1} ; the remaining approximating functions are:

$$ar_{n+1} : \begin{cases} \langle \langle y_1, \dots, y_m \rangle, x \rangle = \langle y_1, \dots, y_m, x \rangle & \text{if } x \neq * \\ \langle y, x \rangle = \langle *, x \rangle & \text{if } y \neq * \neq x \end{cases}$$

$$first_{n+1} : \langle x_1, \dots, x_m \rangle = x_1 \text{ if } m \geq 1 \text{ and } x_1 \neq *$$

$$last_{n+1} : \langle x_1, \dots, x_m \rangle = x_m \text{ if } m \geq 1 \text{ and } x_m \neq *$$

$$tl_{n+1} : \begin{cases} \langle x_1, * \rangle = \perp \\ \langle x_1, \dots, x_m \rangle = \langle x_2, \dots, x_m \rangle & \text{if } m \geq 1 \text{ and } x_1 \neq * \\ \langle *, x_2, x_3, \dots, x_m \rangle = \langle *, x_3, \dots, x_m \rangle & \text{if } m \geq 3 \end{cases}$$

$$tr_{n+1} : \begin{cases} \langle *, x_2 \rangle = \perp \\ \langle x_1, \dots, x_m \rangle = \langle x_1, \dots, x_{m-1} \rangle & \text{if } m \geq 1 \text{ and } x_m \neq * \\ \langle x_1, \dots, x_{m-1}, x_m, * \rangle = \langle x_1, \dots, x_{m-1}, * \rangle & \text{if } m \geq 2 \end{cases}$$

$$null_{n+1} : \begin{cases} \langle \rangle = T \\ a = F & \text{if } a \text{ an atom} \\ \langle x_1, \dots, x_m \rangle = F & \text{if } m \geq 1 \end{cases}$$

$$cons_{n+1} : \begin{cases} \langle f_1, \dots, f_m \rangle = h \in (D_{n-1} \rightarrow D_{n-1}) & \text{if } f_i \neq * , \\ & \text{where } h(x) = \langle f_1 : x, \dots, f_m : x \rangle \\ \langle f_1, \dots, f_k, *, f_{k+1}, \dots, f_m \rangle = h \in (D_{n-1} \rightarrow D_{n-1}) , \\ & \text{where } h(x) = \langle f_1 : x, \dots, f_k : x, *, f_{k+1} : x, \dots, f_m : x \rangle \end{cases}$$

$$cond_{n+1} : \begin{cases} \langle T, u, v \rangle = u & \text{if } u \neq * \neq v \\ \langle F, u, v \rangle = v & \text{if } u \neq * \neq v \end{cases}$$

$$comp_{n+1} : \langle f, g \rangle = h \in (D_{n-1} \rightarrow D_{n-1}) \text{ if } f \neq * \neq g, \text{ where } h(x) = f:(g:x)$$

$$apply_{n+1} : \langle f, x \rangle = f:x \text{ if } f \neq * \neq x$$

$$K_{n+1} : x = h \in (D_{n-1} \rightarrow D_{n-1}), \text{ where } h(z) = P_{n-1}(x)$$

$$id_{n+1} : x = x.$$

We now complete the definition of μ by taking, for each primitive function symbol f ,

$$\mu(f) = \{f_n \mid n \in \mathbb{N}\}^{dl}.$$

Of course, we must check that $\mu(f)$ is well defined for each primitive function symbol f . In particular, we must check that f_n is indeed a continuous function. The details in each case are straightforward and left to the energetic reader. We remark, however, that we must use the fact that $\cdot_n$ is continuous in each of its arguments to prove that $cons_{n+1}$, $comp_{n+1}$, and $apply_{n+1}$ are continuous. Also note that $\mu(f) \in D^*$ by Proposition 2.6.

Most of the definitions of the primitive functions are straightforward consequences of our desiderata for extending FP, but the behaviour of *tail* when applied to a partially defined sequence may appear somewhat unusual. Essentially, we have chosen it to be as defined as possible and still be monotonic. Since $\langle *, x, y \rangle \leq_I \langle z_1, \dots, z_k, x, y \rangle$ for any elements $z_1, \dots, z_k$, we must have $tl:\langle *, x, y \rangle \leq_I \langle z_2, \dots, z_k, x, y \rangle$. Since we might have $k = 0$, we also require that $tl:\langle *, x, y \rangle \leq_I \langle y \rangle$. The greatest element that meets these requirements is $\langle *, y \rangle$. Again, other consistent choices are also possible.

3. Rewrite Rules, Observable Completeness, and Strong Completeness

Now that we have a denotational semantics, the next step is to describe the set of rewrite rules that we use as the basis of computation in FP and to show that the rules are sound with respect to the semantics. The entire set of rewrite rules (which are sometimes also called reduction rules) is given in Figure 1 below.

As usual, each of these is actually a rule schema, which stands for all the substitution instances gotten by substituting actual FP expressions for variables. E.g., $al:\langle a, \langle \langle b \rangle, c \rangle \rangle \rightarrow \langle a, \langle b \rangle, c \rangle$ is a substitution instance of rule 1. The infinite collection of rules thus obtained is taken to be the basis of a *rewrites to* relation ($\rightarrow$). Of course, we also want to allow rewriting to occur if a substitution instance appears as a subexpression of a larger expression.

Definition 3.1. An expression is a *redex* iff it is a substitution instance of the left hand side of some rule. A *context* is an FP expression with a “hole” in it. (This is just the FP analogue for the corresponding definition in the λ -calculus.) If C is a context, we write $C[x]$ for the result of filling the hole in context C by x .

$\text{al: } \langle x, \langle y_1, \dots, y_n \rangle \rangle \rightarrow \langle x, y_1, \dots, y_n \rangle$
 $\text{ar: } \langle \langle y_1, \dots, y_n \rangle, x \rangle \rightarrow \langle y_1, \dots, y_n, x \rangle$
 $\text{first: } \langle x_1, \dots, x_n \rangle \rightarrow x_1 \ (n \geq 1)$
 $\text{first:}(\text{al: } \langle x, y \rangle) \rightarrow x$
 $\text{first:}(\text{ar: } \langle \text{al: } \langle x, y \rangle, z \rangle) \rightarrow x$
 $\text{first:}(\text{ar: } \langle \text{ar: } \langle x, y \rangle, z \rangle) \rightarrow \text{first:}(\text{ar: } \langle x, y \rangle)$
 $\text{last: } \langle x_1, \dots, x_n \rangle \rightarrow x_n \ (n \geq 1)$
 $\text{last:}(\text{ar: } \langle x, y \rangle) \rightarrow y$
 $\text{last:}(\text{al: } \langle x, \text{ar: } \langle y, z \rangle \rangle) \rightarrow z$
 $\text{last:}(\text{al: } \langle x, \text{al: } \langle y, z \rangle \rangle) \rightarrow \text{last:}(\text{al: } \langle y, z \rangle)$
 $\text{tl: } \langle x_1, \dots, x_n \rangle \rightarrow \langle x_2, \dots, x_n \rangle \ (n \geq 1)$
 $\text{tl:}(\text{al: } \langle x, \text{al: } \langle y, z \rangle \rangle) \rightarrow \text{al: } \langle y, z \rangle$
 $\text{tl:}(\text{al: } \langle x, \text{ar: } \langle y, z \rangle \rangle) \rightarrow \text{ar: } \langle y, z \rangle$
 $\text{tl:}(\text{ar: } \langle \text{al: } \langle x, y \rangle, z \rangle) \rightarrow \text{ar: } \langle y, z \rangle$
 $\text{tl:}(\text{ar: } \langle \text{ar: } \langle x, y \rangle, z \rangle) \rightarrow \text{ar: } \langle \text{tl:}(\text{ar: } \langle x, y \rangle), z \rangle$
 $\text{tr: } \langle x_1, \dots, x_n \rangle \rightarrow \langle x_1, \dots, x_{n-1} \rangle \ (n \geq 1)$
 $\text{tr:}(\text{ar: } \langle \text{ar: } \langle x, y \rangle, z \rangle) \rightarrow \text{ar: } \langle x, y \rangle$
 $\text{tr:}(\text{ar: } \langle \text{al: } \langle x, y \rangle, z \rangle) \rightarrow \text{al: } \langle x, y \rangle$
 $\text{tr:}(\text{al: } \langle x, \text{ar: } \langle y, z \rangle \rangle) \rightarrow \text{al: } \langle x, y \rangle$
 $\text{tr:}(\text{al: } \langle x, \text{al: } \langle y, z \rangle \rangle) \rightarrow \text{al: } \langle x, \text{tr:}(\text{al: } \langle y, z \rangle) \rangle$
 $\text{null: } \langle \rangle \rightarrow T$
 $\text{null: } a \rightarrow F \text{ (for any atom } a)$
 $\text{null: } \langle x_1, \dots, x_n \rangle \rightarrow F \ (n \geq 1)$
 $\text{null:}(\text{al: } \langle x, y \rangle) \rightarrow F$
 $\text{null:}(\text{ar: } \langle x, y \rangle) \rightarrow F$
 $(\text{cons: } \langle f_1, \dots, f_n \rangle):x \rightarrow \langle f_1:x, \dots, f_n:x \rangle$
 $(\text{cons:}(\text{al: } \langle f, g \rangle)):x \rightarrow \text{al: } \langle f:x, (\text{cons:g}):x \rangle$
 $(\text{cons:}(\text{ar: } \langle f, g \rangle)):x \rightarrow \text{ar: } \langle (\text{cons:f}):x, g:x \rangle$
 $\text{cond: } \langle T, x, y \rangle \rightarrow x$
 $\text{cond: } \langle F, x, y \rangle \rightarrow y$
 $(\text{comp: } \langle f, g \rangle):x \rightarrow f:(g:x)$
 $\text{apply: } \langle f, x \rangle \rightarrow f:x$
 $(K:x):y \rightarrow x$
 $\text{id: } x \rightarrow x$

Figure 1: The FP rewrite rules.

For example, if C is the context $\text{al: } \langle _, \langle a \rangle \rangle$, then $C[b]$ is $\text{al: } \langle b, \langle a \rangle \rangle$. An expression x *rewrites to* an expression y , written $x \rightarrow y$, iff there is some context C such that $x = C[x']$, $y = C[y']$, and $x' \rightarrow y'$ is an instance of a rewrite rule. As usual, the reflexive,

transitive closure of the $\rightarrow$ relation is denoted $\rightarrow^*$; thus we write $x \rightarrow^* y$ if there is a sequence $x_0, \dots, x_k$ such that $x = x_0$, $y = x_k$, and $x_i \rightarrow x_{i+1}$ for $i = 0, \dots, k-1$.

Some of our rewrite rules may seem surprisingly complex. For instance the treatment of *tail* could seemingly be simplified to $tl:(al: \langle x, y \rangle) \rightarrow y$. Such a rule would not be consistent with the semantic definition of *tl* and *al* however. The meaning of $tl:(al: \langle a, b \rangle)$ is not b (since $al:\langle a, b \rangle = \langle a, * \rangle$ and $tl:\langle a, * \rangle = \perp$). What is true is that $tl:(al: \langle x, y \rangle) \rightarrow y$, *provided y is a sequence*. Syntactically, we are guaranteed that y is a sequence exactly if it has one of the forms $\langle \dots \rangle$, $al: \langle x, y \rangle$, or $ar: \langle x, y \rangle$. So, for example, the second and third rules for *tl* are really instances of the rule $tl:(al: \langle x, y \rangle) \rightarrow y$, specialized to cases where y has the appropriate form. Similar remarks hold for the fourth and fifth *tl* rules.

This also accounts for the remarkable behavior mentioned in the introduction, that an expression of the form $tl:x$ which is not a redex can be rewritten to $tl:x'$ which is a redex, which can in turn be rewritten to $tl:x''$ which is not a redex. To see this, consider $tl:(id:(al: \langle a, al: \langle b, \langle c \rangle \rangle \rangle))$. This expression is not itself a redex, but it can be rewritten to $tl:(al: \langle a, al: \langle b, \langle c \rangle \rangle \rangle)$, which is a redex (w.r.t. the rule $tl:(al: \langle x, al: \langle y, z \rangle \rangle) \rightarrow al: \langle y, z \rangle$). This in turn can be rewritten to $tl:(al: \langle a, \langle b, c \rangle \rangle)$ which is again *not* a redex. Of course, this last expression can be rewritten to an expression (namely $tl: \langle a, b, c \rangle$) to which a tail rule will again apply.

We are now ready to address the first question posed in the introduction, “Do we have too many rules, thus making equivalent two expressions that should be different?” To show that this is not the case, we need to show that the $\rightarrow^*$ relation preserves meaning, i.e., that the rewriting rules are sound.

Theorem 3.2. *If $x \rightarrow^* y$, then $\mu(x) = \mu(y)$.*

Proof. It clearly suffices to show that if $x \rightarrow y$ is an instance of a rewrite rule, then $\mu(x) = \mu(y)$. We proceed by a case-by-case analysis of each rewrite rule. Since the proofs in all cases are similar and quite straightforward, we limit ourselves to two representative cases here.

First consider the rule $(cons: \langle f_1, \dots, f_n \rangle):x \rightarrow \langle f_1:x, \dots, f_n:x \rangle$. In this case we have

$$\begin{aligned}
& \mu((cons: \langle f_1, \dots, f_n \rangle):x) \\
&= \{cons_m \mid m = 1, 2, 3, \dots\}^{dl} : \{\langle g_1, \dots, g_n \rangle \mid g_i \in \mu(f_i)\}^{dl} : \{y \mid y \in \mu(x)\}^{dl} \quad (\text{by definition of } \mu) \\
&= \{(cons_m: \langle g_1, \dots, g_n \rangle):y \mid g_i \in \mu(f_i), y \in \mu(x), m = 1, 2, 3, \dots\}^{dl} \quad (\text{by Lemma 2.9}) \\
&= \{\langle g_1:y, \dots, g_n:y \rangle \mid g_i \in \mu(f_i) \cap D_{m-1}, y \in \mu(x), m = 1, 2, 3, \dots\}^{dl} \quad (\text{by definition of } cons_m) \\
&= \{\langle g_1:y, \dots, g_n:y \rangle \mid g_i \in \mu(f_i), y \in \mu(x)\}^{dl} \\
&= \mu(\langle f_1:x, \dots, f_n:x \rangle) \quad (\text{by definition of } : \text{ and } \mu).
\end{aligned}$$

The reader will note that most of the work in this case is involved in checking that the mathematical machinery of D^* is working correctly; only the fourth step above depends on the particular computational details of *cons*.

The other case that we consider is $tl:(ar: \langle ar: \langle x, y \rangle, z \rangle) \rightarrow ar: \langle tl:(ar: \langle x, y \rangle), z \rangle$; this is one of the rules that involves the most computational detail. In our discussion, we omit the repetition of the machinations of D^* , and present instead only the analogue of the crucial step above. Note that the meaning of each side of the rule varies according to the meaning of x . There are four possibilities: $\mu(x)$ is the null sequence, a non-null sequence with first element $\neq *$, a non-null sequence with first element $= *$, or a non-sequence. We show that in each case the meaning of the two sides is the same by the following table of values of the component subexpressions according to the meaning of x . Note we write x for $\mu(x)$, ar for $\mu(ar)$, etc.

x	$\langle \rangle$	$\langle x_1, \dots, x_n \rangle \ (n \geq 1)$	$\langle *, x_2, \dots, x_n \rangle \ (n \geq 2)$	non-seq
$ar: \langle x, y \rangle$	$\langle y \rangle$	$\langle x_1, \dots, x_n, y \rangle$	$\langle *, x_2, \dots, x_n, y \rangle$	$\langle *, y \rangle$
$ar: \langle ar: \langle x, y \rangle, z \rangle$	$\langle y, z \rangle$	$\langle x_1, \dots, x_n, y, z \rangle$	$\langle *, x_2, \dots, x_n, y, z \rangle$	$\langle *, y, z \rangle$
$tl:(ar: \langle ar: \langle x, y \rangle, z \rangle)$	$\langle z \rangle$	$\langle x_2, \dots, x_n, y, z \rangle$	$\langle *, x_3, \dots, x_n, y, z \rangle$	$\langle *, z \rangle$
$tl:(ar: \langle x, y \rangle)$	$\langle \rangle$	$\langle x_2, \dots, x_n, y \rangle$	$\langle *, x_3, \dots, x_n, y \rangle$	$\perp$
$ar: \langle tl:(ar: \langle x, y \rangle), z \rangle$	$\langle z \rangle$	$\langle x_2, \dots, x_n, y, z \rangle$	$\langle *, x_3, \dots, x_n, y, z \rangle$	$\langle *, z \rangle$

Observe that rows 3 and 5 in the table, representing the left and right sides of the rewrite rule, are identical. Thus, for all possible meanings of x , the rule is sound. $\square$

Besides soundness, we also want to know, as we said in the introduction, that we have *enough* rewrite rules. Do the rewrite rules really capture the semantics of the language? Do they allow us to carry out all the computing we want?

Ideally we would like to show that if $\mu(x) = \mu(y)$, then we have some z such that $x \rightarrow^* z$ and $y \rightarrow^* z$. As we mentioned in the introduction, this goal is unattainable. However, it is attainable if we restrict x and y to be expressions whose meaning is observable. Since these are the expressions the programmer most cares about, this is a reasonable sense in which we do indeed have enough rules. To make this precise, we need some definitions.

Definition 3.3. Let the *observable expressions* be those that can be formed starting with atoms and closing off under the sequence constructor. More formally, the set of observable expressions is the least set $\mathcal{O}$ such that:

1. $\mathcal{A} \subseteq \mathcal{O}$,
2. if $x_1, \dots, x_n \in \mathcal{O}$, then $\langle x_1, \dots, x_n \rangle \in \mathcal{O}$.

To each observable expression there corresponds an *observable item* which is obtained by replacing $\langle, \rangle$, and boldface wherever they occur in the observable expression by $\langle, \rangle$, and italics respectively. Note that neither $\perp$ nor $*$ appears in any sequence item which is

observable. It is easy to check that every observable item is finite (i.e., an element of Fin). Given an observable item x , let x^e be the observable expression to which it corresponds.

Thus, one reasonable notion of there being “enough” rewrite rules is:

(*) If x is an FP expression such that $\mu(x)$ is an observable item, then $x \rightarrow^* \mu(x)^e$.

For example, if we have an expression x such that $\mu(x) = \langle a, \langle b \rangle \rangle$, where a and b are atomic items, then from (*) it would follow that $x \rightarrow^* \langle a, \langle b \rangle \rangle$, even if x contains subexpressions whose meanings are functions or infinite sequences.

We call the kind of completeness described in (*) *observable completeness*. If we could show that observable completeness held for FP, then we would know that, at least for expressions whose meanings are observable, there would be enough rewrite rules in FP. In this precise sense, (sound) rewrite rules such as $\text{first}:(\text{cons}:x):y \rightarrow (\text{first}:x):y$ or $\text{tl}:(\text{cons}:x):y \rightarrow (\text{cons}:(\text{tl}:x)):y$ would not be needed in our system.

But it is also desirable to be able to say something about FP expressions whose meanings are not observable. For example, if we have an expression whose meaning is a sequence, then we would want there to be enough rewrite rules to enable us to rewrite the expression to one that is a sequence, or at least one that begins with al or ar . Similarly, we might expect that an expression whose meaning is a function rewrites to something that looks like a function. These intuitions motivate the following definition.

Definition 3.4. An FP expression x is *transparent* iff

1. if $\mu(x) = a$ for some atomic item a , then $x = a$,
2. if $\mu(x) = \langle w_1, \dots, w_n \rangle$, then x is of the form $\langle y_1, \dots, y_n \rangle$, $\text{al}: \langle y, z \rangle$, or $\text{ar}: \langle y, z \rangle$; if $\mu(x)$ is a sequence item of definite length, then x is of the form $\langle y_1, \dots, y_n \rangle$, and
3. if $\mu(x)$ is a function item, then x is either a primitive function symbol f , or of the form $K:y$, $\text{comp}:y$, or $\text{cons}:y$.

Note that if $\mu(x) = \perp$, then x is automatically transparent. Also note that if x is transparent and $x \rightarrow y$, then y is also transparent.

By looking at a transparent expression, provided its meaning is not $\perp$, we can immediately tell whether it is an atomic item, sequence item, or function item (hence the name *transparent*). This leads us to the following notion of having “enough” rules.

Definition 3.5. A set of rewrite rules is *strongly complete* for FP iff for all FP expressions x there is a transparent expression y such that $x \rightarrow^* y$.

We remark that in the sequel, when we discuss strong completeness or observable completeness, we always take the set of rewrite rules under question to be those listed above.

Proposition 3.6. *Strong completeness implies observable completeness.*

Proof. Suppose we have strong completeness for FP. We show by a straightforward induction

on the number of brackets ($\langle \rangle$'s) in $\mu(x)$ that if $\mu(x)$ is an observable item, then $x \rightarrow^* \mu(x)^e$. If there are no brackets in $\mu(x)$, then $\mu(x)$ is an atomic item and it is immediate from the definition of strong completeness that $x \rightarrow^* \mu(x)^e$. For the general case, suppose $\mu(x) = \langle y_1, \dots, y_n \rangle$. Note that since we are dealing with observable items here, no $y_i = *$. From strong completeness it follows that $x \rightarrow^* \langle y_1, \dots, y_n \rangle$. By the soundness of the FP rewrite rules, it follows that $\mu(y_i) = y_i$, $i = 1, \dots, n$. But y_i has fewer brackets than x and is also an observable item. By the induction hypothesis, $y_i \rightarrow^* y_i^e$, $i = 1, \dots, n$. Since $\mu(x)^e = \langle y_1^e, \dots, y_n^e \rangle$, it follows that $x \rightarrow^* \mu(x)^e$ as desired. $\square$

In Section 5, we prove that strong completeness holds for FP. By Proposition 3.6, observable completeness also holds for FP. The proof is nontrivial. As we shall see, the proof is strongly intertwined with notions of normalization, which is the subject of the next section. The techniques introduced there will also allow us to give a relatively easy proof of strong completeness for a large subset of FP.

4. Normalization

Since rewriting can be viewed as a computing process, we might want to ask if that process always terminates. (Some computation paths may terminate by producing an expression to which no rules apply, while other computation paths may run forever.) Intuitively, we say an expression x is weakly normalizable iff there is some computation starting with x that terminates; x is strongly normalizable iff all computations starting with x terminate, or equivalently, iff there are no infinite computations starting with x . This motivates the following definitions.

Definition 4.1. An expression is in *normal form* iff no subexpression is a redex. An expression is *weakly normalizable* (respectively, *strongly normalizable*) iff some (respectively, every) reduction sequence for it leads to a normal form. A language is *weakly normalizable* (respectively, *strongly normalizable*) iff every expression in that language is weakly normalizable (respectively, strongly normalizable).

Not all FP expressions are strongly normalizable or even weakly normalizable for that matter; for example, it is easy to check that `fix:id` is not weakly normalizable. However, large subsets of FP expressions are strongly normalizable. Let $\text{FP-}\{\text{apply}\}$ (resp. $\text{FP-}\{\text{cons}\}$, $\text{FP-}\{\text{comp}\}$) consist of all FP expressions which do not contain the primitive function `apply` (resp. `cons`, `comp`).

Theorem 4.2. *FP-}\{\text{apply}\}, FP-}\{\text{cons}\}, and FP-}\{\text{comp}\} are all strongly normalizable.*

Proof. The idea in all cases is to assign a natural number to the expressions in each of these sublanguages in such a way that every rewrite rule forces the number of the expression to decrease. For the case $\text{FP-}\{\text{apply}\}$, we do this as follows:

$$\begin{aligned} N(a) &= 1 \text{ if } a \text{ is an atom or primitive function,} \\ N(\langle x_1, \dots, x_n \rangle) &= N(x_1) + \dots + N(x_n) + 1, \end{aligned}$$

$$N(f:g) = 2^{N(f)} \times N(g).$$

Whenever we apply a rewrite rule (except for the apply rule), the number of the expression goes down. We prove this for the comp rule:

$$\begin{aligned} N((\text{comp}: \langle x, y \rangle):z) &= 2^{1 \times (N(x)+N(y)+1)} \times N(z) \\ &> 2^{N(x)} \times 2^{N(y)} \times N(z) \\ &= N(x:(y:z)). \end{aligned}$$

All the other rewrite rules (of FP-{apply}) are just as easy.

For FP-{comp}, we need another way of assigning numbers to expressions. The following one works:

$L(a) = 1$ for any atom or primitive function a other than null, K, or id,

$L(\text{null}) = L(K) = L(\text{id}) = 2$,

$L(\langle x_1, \dots, x_n \rangle) = 1 + L(x_1) + \dots + L(x_n)$,

$$L(f:g) = \begin{cases} L(f) \times L(g)^{L(g)} & \text{if } f \neq \text{al, ar} \\ L(g) & \text{otherwise.} \end{cases}$$

We again leave it to the reader to check that for any rewrite rule $x \rightarrow y$ other than that for comp, we have $L(x) > L(y)$. Thus FP-{comp} is strongly normalizable.

For the case FP-{cons}, we assign numbers to FP expressions in a different fashion. Note that all the rules (other than the ones for cons) have a right-hand side whose length is no longer than the left-hand side. Indeed, all the rules except $\text{tl}:(\text{ar}: \langle \text{ar}: \langle x, y \rangle, z \rangle) \rightarrow \text{ar}: \langle \text{tl}:(\text{ar}: \langle x, y \rangle), z \rangle$ and the corresponding rule for *tr decrease* length. Thus, for any FP expression x , we define

$$M(x) = \langle |x|, N(x) \rangle,$$

where $|x|$ is the length of x roughly viewed as a string of symbols. More formally, we define $|x| = 1$ if x is an atom or a primitive function symbol, $|\langle x_1, \dots, x_n \rangle| = 1 + |x_1| + \dots + |x_n|$, and $|y:z| = 1 + |y| + |z|$. If we now define the obvious lexicographic order on pairs of integers ($\langle m, n \rangle \leq \langle m', n' \rangle$ iff $m < m'$, or $m = m'$ and $n \leq n'$), then it is an easy matter to check that for every rewrite rule $x \rightarrow y$ other than those involving cons, we have $M(x) > M(y)$. Thus FP-{cons} is strongly normalizable. $\square$

This result is essentially optimal, since we know from the examples given above that full FP is not weakly normalizable. The examples together with the theorem just proved indicate that somehow it is the *interaction* of apply, cons, and comp, rather than any of them individually, that makes FP computation so complicated.

The following theorem relates the notions of normal form and strong completeness.

Theorem 4.3. *If x is an FP expression in normal form, then x is transparent.*

Proof. We proceed by induction on the structure of x . The cases where x is an atom, primitive function symbol, or a sequence are all trivial. Suppose that x is of the form $x_1:x_2$. Since x is in normal form, then so must be x_1 and x_2 . If $\mu(x_1)$ is not a function item, then $\mu(x) = \perp$, and again the theorem holds vacuously. Suppose that $\mu(x_1)$ is a function item. By the induction hypothesis, x_1 is of the form f for some primitive function symbol f , or of the form $K:y$, $\text{comp}:y$, or $\text{cons}:y$. We show that in each of these cases one of the following happens: (a) $x_1:x_2$ is a redex (contradicting the assumption that x is in normal form), (b) $\mu(x) = \perp$ (in which case the theorem holds vacuously), (c) $\mu(x)$ is a function item and x is of the form $K:y$, $\text{comp}:y$, or $\text{cons}:y$, so the theorem holds, or (d) $\mu(x)$ is a sequence item of indefinite length and x is of the form $\text{al}: \langle y, z \rangle$ or $\text{ar}: \langle y, z \rangle$, so again the theorem holds.

If x_1 is one of K, comp , or cons , then it is easy to see that one of (b) or (c) must hold. If x_1 is of the form id or $K:y$, then we immediately have that $x_1:x_2$ is a redex. If x_1 is of the form apply , then either $\mu(x_1:x_2) = \perp$, or $\mu(x_2) = \langle f, z \rangle$, where $f, z \neq *$. But then by the induction hypothesis, x_2 is of the form $\langle f, z \rangle$, so that $x_1:x_2$ must be a redex. The argument in the case where x_1 is of the form cond , null , $\text{cons}:y$, or $\text{comp}:y$ is similar and left to the reader. If x_1 is of the form al , then either $\mu(x) = \perp$ or $\mu(x_2)$ is of the form $\langle y, z \rangle$, where $y, z \neq *$. Thus, $\mu(x)$ is of the form $\langle w_1, \dots, w_m \rangle$ and, by the induction hypothesis, x_2 is of the form $\langle y, z \rangle$. If $\mu(x)$ is a sequence of definite length, then in fact z must be a sequence of definite length. Thus, by the induction hypothesis, z is of the form $\langle z_1, \dots, z_n \rangle$, and x is a redex. The arguments in the case that x_1 is ar , tl , tr , first , or last are similar and again left to the reader. $\square$

As an immediate corollary to Theorem 4.3, we get

Corollary 4.4. *If x is in normal form and $\mu(x)$ is an observable item, then $x = \mu(x)^e$.*

Proof. By an easy induction on the depth of nesting of $\langle \rangle$'s in $\mu(x)$, similar to the proof of Proposition 3.6. $\square$

From Theorem 4.3, it immediately follows that strong completeness holds for a large subset of FP terms, namely all those terms in $(FP - \{\text{apply}\}) \cup (FP - \{\text{comp}\}) \cup (FP - \{\text{cons}\})$. Indeed, Theorem 4.3 shows that strong completeness holds for all weakly normalizable FP expressions. However, as we have shown, there are some FP expressions that are not weakly normalizable. Nevertheless, we can show that strong completeness holds for full FP: every expression can be rewritten to a transparent expression. This is the subject of the next section.

5. FP' and strong completeness for FP

The key idea in the proof of strong completeness is to construct a variant of FP called FP' on which we impose a notion of *type*. The proof then proceeds in three steps. The first, quite simple, is to show that any FP expression can be approximated arbitrarily well by a

corresponding FP' expression (Proposition 5.3). Secondly, we show that FP' reduction sequences correspond in a precise sense to FP reduction sequences (Proposition 5.8). Next, by extending techniques of typed λ -calculus (cf. [FLO83]), we show that FP' is essentially strongly normalizable, in a sense to be made precise below. We then prove an analogue to Theorem 4.3 for FP' . The results then follow using techniques similar to those sketched at the end of the previous section.

We now define FP' , a meaning function μ' that maps FP' expressions to elements of D , a type function τ that associates with every FP' expression a natural number (its type), and a “pull-back” function ρ that associates with each FP' expression a corresponding FP expression. FP' expressions look essentially like FP expressions, except that primitive function symbols are subscripted by integers; ρ removes these subscripts. For technical reasons, we also allow expressions of the form $P_m(x)$ in FP' , where P_m projects the meaning of x to D_m . We also allow expressions of the form $\Omega(x)$, where x is an FP expression, and elements of the form $*(x_1, \dots, x_n)$ to appear in sequences, where $x_1, \dots, x_n$ are FP expressions. For every FP' expression x , $\mu'(x)$ (which is in D) can be viewed as a “finite” approximation to $\mu(\rho(x))$ (which is in D^*).

Remark. The version of FP' we define here is somewhat different from that of an earlier version of this paper ([HW³85]). In [HW³85] primitive function symbols were subscripted by elements of D rather than integers, and there were no expressions of the form $P_m(x)$. The changes we have made allow us to simplify our proof and also bring our techniques closer to those used by Wadsworth ([Wa78]) to prove limiting completeness for the untyped λ -calculus. Wadsworth considered a typed variant of the λ -calculus in his proof. Our ρ corresponds to his $*$. Wadsworth required that all expressions have integer subscripts, which essentially corresponds in our case to applying P_m to the expression. However, not all FP' expressions are of the form $P_m(y)$. By considering this version of FP' , it will be easier for us to point out the essential differences between FP and λ -calculus.

Definition 5.1. FP' expressions, and the functions μ' , τ , and ρ are defined inductively as follows:

1. Every atom in $\mathcal{A}$ is an FP' expression; $\mu'(a) = a$, $\tau(a) = 0$, and $\rho(a) = a$.
2. If $f \in \mathcal{F}$, then f_n is an FP' expression for every integer n ; $\mu'(f_n) = f_n$, $\tau(f_n) = n$, and $\rho(f_n) = f$.
3. If x is an FP expression, then $\Omega(x)$ is an FP' expression; $\mu'(\Omega(x)) = \perp$, $\tau(\Omega(x)) = 0$, and $\rho(\Omega(x)) = x$.
4. If $x_1, \dots, x_n$ are FP' expressions, then $\langle x_1, \dots, x_n \rangle$ is an FP' expression; $\mu'(\langle x_1, \dots, x_n \rangle) = \langle \mu'(x_1), \dots, \mu'(x_n) \rangle$, $\tau(\langle x_1, \dots, x_n \rangle) = 1 + \max\{\tau(x_1), \dots, \tau(x_n)\}$, and $\rho(\langle x_1, \dots, x_n \rangle) = \langle \rho(x_1), \dots, \rho(x_n) \rangle$.
5. If $x_1, \dots, x_n$, $n \geq 1$, are FP' expressions, and $z_1, \dots, z_k$, $k \geq 0$, are FP expressions, then $\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle$ is an FP' expression; $\mu'(\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle) = \langle \mu'(x_1), \dots, \mu'(x_i), *, \mu'(x_{i+1}), \dots, \mu'(x_n) \rangle$,

$$\tau(\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle) = 1 + \max\{\tau(x_1), \dots, \tau(x_n)\}, \quad \text{and}$$

$$\rho(\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle) = \langle \rho(x_1), \dots, \rho(x_i), z_1, \dots, z_k, \rho(x_{i+1}), \dots, \rho(x_n) \rangle$$

6. If x and y are FP' expressions, then $x:y$ is an FP' expression; $\mu'(x:y) = \mu'(x):\mu'(y)$ and $\rho(x:y) = \rho(x):\rho(y)$. The definition of τ splits into three cases:

a. if $n \geq 1$ and $m = \min(m_1, \dots, m_k)$, then $\tau(al_n:P_{m_1}(\dots(P_{m_k}(\langle z_1, z_2 \rangle))\dots)) = \min[n-1, \max(\min(m, \tau(z_1)+1), \min(m-1, \tau(z_2)))]$,

b. if $n \geq 1$ and $m = \min(m_1, \dots, m_k)$, then $\tau(ar_n:P_{m_1}(\dots(P_{m_k}(\langle z_1, z_2 \rangle))\dots)) = \min[n-1, \max(\min(m-1, \tau(z_1)), \min(m, \tau(z_2)+1))]$,

c. $\tau(x:y) = \max\{0, \tau(x) - 1\}$ otherwise.

7. If x is an FP' expression, then $P_m(x)$ is an FP' expression; $\mu'(P_m(x)) = P_m(\mu'(x))$, $\tau(P_m(x)) = \min(m, \tau(x))$, and $\rho(P_m(x)) = \rho(x)$.

We remark that the special case for al and ar in the definition of τ is needed to make Proposition 5.6 below go through. Although the expression looks somewhat complicated, the intuitive idea is motivated by the fact that since $al:\langle z, \langle z_1, \dots, z_k \rangle \rangle \rightarrow^* \langle z, z_1, \dots, z_k \rangle$, for sufficiently large n we want $\tau(al_n:\langle z, \langle z_1, \dots, z_k \rangle \rangle) = \tau(\langle z, z_1, \dots, z_k \rangle)$.

The following proposition shows that an expression in FP' semantically approximates the corresponding FP expression. The second half of the proposition shows that this approximation can be made arbitrarily good.

Definition 5.2. A *star-free* FP' expression is one with no subexpressions of the form $*(z_1, \dots, z_k)$.

Proposition 5.3.

1. For any FP' expression x' , we have $\mu'(x') \leq_I \mu(\rho(x'))$.
2. If x is an FP expression, $x_0 \leq_I \mu(x)$, and $x_0 \in Fin$, then there exists a star-free FP' expression x' such that $x_0 \leq_I \mu'(x')$ and $\rho(x') = x$.

Proof. The first part is immediate by induction on the structure of FP' expressions. For the second part, we proceed by induction on the structure of the FP expression x . If x is an atom, then we can clearly take $x' = x$. If x is a primitive function symbol f , we take $x' = f_n$, where $x_0 \in D_n$. If x is a sequence of the form $\langle x_1, \dots, x_n \rangle$, then there exist $y_1, \dots, y_n \in Fin$ such that $x_0 \leq_I \langle y_1, \dots, y_n \rangle$ and $y_j \leq \mu(x_j)$. (We can actually choose $y_1, \dots, y_n$ so that $x_0 = \langle y_1, \dots, y_n \rangle$ unless x_0 has a $*$ as one of its components, in which case the corresponding y_i 's are taken to be $\perp$'s.) From the induction hypothesis, it follows that we can find FP' expressions $y_1, \dots, y_n$ such that $y_j \leq_I \mu'(y_j)$ and $\rho(y_j) = x_j$. Thus we can take $x' = \langle y_1, \dots, y_n \rangle$. Finally, if x is of the form $y:z$ and $x_0 \leq_I \mu(x)$, then there must be $y_0, z_0 \in Fin$ such that $y_0 \leq_I \mu(y)$, $z_0 \leq_I \mu(z)$, and $x_0 \leq_I y_0:z_0$. To see this, note that by Proposition 2.4 and Lemma 2.9, we have

$$\mu(x) = \mu(y):\mu(z) = (Fin \cap \mu(y))^{dl} : (Fin \cap \mu(z))^{dl} = \{y:z \mid y \in Fin \cap \mu(y), z \in Fin \cap \mu(z)\}^{dl}.$$

Since x_0 is finite, we must have $x_0 \in \{y:z \mid y \in Fin \cap \mu(y), z \in Fin \cap \mu(z)\}^d$. Thus we can find the required y_0 and z_0 . By the induction hypothesis we can find FP' expressions y' and z' such

that $\rho(y') = y$, $\rho(z') = z$, $y_0 \leq_I \mu'(y')$, and $z_0 \leq_I \mu'(z')$. It is easy to check that $x' = y':z'$ satisfies the proposition. $\square$

Lemma 5.4. *If x' is an FP' expression and $\tau(x') = n$, then $\mu'(x') \in D_n$.*

Proof. A straightforward induction on the structure of x' . $\square$

Intuitively, we view FP' as a typed language, where the type of an expression x is given by $\tau(x)$. We define an FP' expression x to be *well-typed* iff for every subexpression of x of the form $y:z$ we have $\tau(z) < \tau(y)$. Note that there are legal FP' expressions that are not well-typed; however, we will restrict FP -style rewriting in FP' to expressions that are, among other things, well-typed.

Actually, we will have four kinds of FP' rewrite rules: *FP-style rules*, *lowering rules*, *P-rules*, and *bottom rules*. FP -style rules simulate the rewrite rules of FP , lowering rules allow us to reduce an expression to a well-typed expression (in general, by introducing P_m 's into the expression), P -rules allow us to manipulate the P_m 's, and bottom rules allows us to rewrite an FP expression that is the argument of an Ω or $*$. We list the FP -style rewrite rules in Appendix E, since they are so similar to the FP rewrite rules; the remaining FP' rewrite rules are given in Figure 2 below. Note we use $\rightarrow_a$ to denote FP' rewriting. We remark that the restrictions in the second lowering rule and the fifth P -rule ($P_m(y):z \rightarrow_a P_{m-1}(y:z)$) have been added for purely technical reasons (see Proposition 5.7, Lemma 14.11).

Remark: It is easy to see that FP' does not have the Church-Rosser property. For example, consider the expression $P_7(\text{comp}_8): \langle \text{id}_4, \text{id}_4 \rangle$. We can rewrite this to both $\text{comp}_7: \langle \text{id}_4, \text{id}_4 \rangle$ and $P_6(\text{comp}_8: \langle \text{id}_4, \text{id}_4 \rangle)$. These last two expressions are both in normal form, and hence cannot be rewritten to a common third expression.

Definition 5.5. An expression to which no lowering rules or P -rules apply is said to be *well-formed*.

Note that a well-formed expression must be well-typed (otherwise a lowering rule would apply to it). We impose the requirement that *FP-style rewrite rules may be applied only to well-formed expressions*.

Proposition 5.6. *If x', y' are FP' expressions such that $x' \rightarrow_a^* y'$, then $\mu'(x') = \mu'(y')$; i.e., FP' rewriting is sound. Moreover, $\tau(x') \geq \tau(y')$.*

Proof. It suffices to check that each instance of a rewrite rule is sound and does not increase type. The cases are all straightforward, so we limit ourselves here to a few comments and leave details to the reader. Note that the result would not be true if we did not insist that the left-hand side of an FP -style rewrite rule be well-formed. For example, in general it is not the case that $\mu'(\text{comp}_m: \langle f, g \rangle : x) = \mu'(f:(g:x))$, but equality does hold if $\text{comp}_m: \langle f, g \rangle : x$ is well-typed. Well-formedness is also necessary to ensure that type goes down. The type of $\text{comp}_m: \langle f, g \rangle : x$ is $m - 2$; if this expression is well-formed, then the type of f is at most

Lowering rules:

$$\begin{aligned} x &\rightarrow_a \Omega(\rho(x)) \text{ if } \mu'(x) = \perp \\ x:y &\rightarrow_a x:P_{m-1}(y) \text{ if } \tau(x) = m, m \geq 1, \tau(y) \geq m, \mu(x:y) \neq \perp \end{aligned}$$

P -rules:

$$\begin{aligned} P_m(x) &\rightarrow_a x \text{ if } m \geq \tau(x) \\ P_m(f_n) &\rightarrow_a f_{\min(m,n)} \\ P_m(\langle x_1, \dots, x_n \rangle) &\rightarrow_a \langle P_{m-1}(x_1), \dots, P_{m-1}(x_n) \rangle \text{ if } m \geq 1, x_i \neq * (z_1, \dots, z_k) \\ P_m(\langle x_1, \dots, *(z_1, \dots, z_k), \dots, x_n \rangle) &\rightarrow_a \langle P_{m-1}(x_1), \dots, *(z_1, \dots, z_k), \dots, P_{m-1}(x_n) \rangle \text{ if } m \geq 1 \\ P_m(y):z &\rightarrow_a P_{m-1}(y):z \text{ if } m \geq 1, P_m(y):z \text{ is well-typed, and } \mu'(P_m(y):z) \neq \perp \\ P_m(al_n: \langle y, z \rangle) &\rightarrow_a al_n: \langle P_{m-1}(y), P_m(z) \rangle \text{ if } m \geq 1 \\ P_m(ar_n: \langle y, z \rangle) &\rightarrow_a ar_n: \langle P_m(y), P_{m-1}(z) \rangle \text{ if } m \geq 1 \\ P_m(P_n(z)) &\rightarrow_a P_{\min(m,n)}(z) \end{aligned}$$

Bottom rules:

$$\begin{aligned} \Omega(x) &\rightarrow_a \Omega(y) \text{ if } x \rightarrow y \\ \langle x_1, \dots, *(z_1, \dots, z_i, \dots, z_k), \dots, x_n \rangle &\rightarrow_a \langle x_1, \dots, *(z_1, \dots, z'_i, \dots, z_k), \dots, x_n \rangle \text{ if } z_i \rightarrow z'_i \end{aligned}$$

Figure 2: Lowering rules, P -rules, and bottom rules for FP' .

$m-2$, so the type of $f(g;x)$ is at most $m-3$. Note that the P_{m-2} in the rule $K_m:x;y \rightarrow_a P_{m-2}(x)$ is necessary both to ensure soundness and the fact that type goes down. Finally, note that the special cases in the definition of τ for expressions of the form $al_m: \langle y, z \rangle$ and $ar_m: \langle y, z \rangle$ are needed to show that P -rules involving al and ar do not increase type. $\square$

The following proposition assures us that an FP' expression can always be rewritten to a well-formed expression in at most finitely many steps. Recall that $|x|$ denotes the length of an FP expression viewed as a string of symbols. We extend this notation to FP' expressions as well (where we take $|f_m| = 1$, $|P_m(x')| = 1 + |x'|$, $|\Omega(x)| = 1 + |x|$ and $|\langle x'_1, \dots, *(z_1, \dots, z_k), \dots, x'_m \rangle| = 1 + |x'_1| + \dots + |z_1| + \dots + |z_k| + \dots + |x'_m|$.) It is easy to see that if x' is an FP' expression, then $|x'| \geq |\rho(x')|$.

Proposition 5.7. *No more than $4^{|x'|}$ lowering rules and P -rules can be applied before x' is reduced to a well-formed expression.*

Proof. For an FP' expression x' , we assign a number $W(x')$ by induction on the structure of x' as follows:

$$\begin{aligned} W(a) &= 1 \text{ for an atom } a \\ W(f_m) &= 2 \text{ for a primitive function symbol } f \\ W(\Omega(x)) &= 1 \\ W(P_m(y')) &= W(y') + |\rho(y')| \\ W(\langle x'_1, \dots, x'_m \rangle) &= W(x'_1) + \dots + W(x'_m) + 1 \text{ (where we take } W(*) = 0) \end{aligned}$$

$$W(y':z') = \begin{cases} W(y') \times (|\rho(z')| + 3) + W(z') & \text{if } \tau(y') > \tau(z'), \mu'(y':z') \neq \perp \\ W(y') \times (|\rho(z')| + 3) + W(z') + |\rho(z')| + 1 & \text{otherwise} \end{cases}$$

A straightforward induction on structure now shows that $W(x') \leq 4^{|x'|}$ for all FP' expressions x' . It is also not hard to see that if $x' \rightarrow_a y'$ using a P -rule or lowering rule, then $W(y') < W(x')$. The proof is by a case-by-case analysis of all the P -rules and lowering rules. The only difficulty comes in showing that if $\tau(x'_1) > \tau(x'_2)$ and $x'_1:x'_2 \rightarrow_a y'_1:y'_2$ using a lowering rule or a P -rule, then either $\mu'(x'_1:x'_2) = \perp$ or $\tau(y'_1) > \tau(y'_2)$. (Note that if this property did not hold, then by the second clause of the definition of W , we might have $W(y'_1:y'_2) \geq W(x'_1:x'_2)$.) It is easy to see that all the P -rules preserve type, as does the second lowering rule. Thus the only problem comes if we apply the rule $z' \rightarrow_a \Omega(\rho(z'))$ to some subexpression z' of x'_1 . But it is also easy to show by induction on structure that if $x'_1 \rightarrow_a y'_1$ as a result of using this rule, then either $\tau(x'_1) = \tau(y'_1)$ or $\mu'(x'_1)$ is not a function item, so that $\mu'(x'_1:x'_2) = \perp$. $\square$

The following proposition assures us that in a precise sense, FP' rewriting simulates FP rewriting.

Proposition 5.8.

1. If x' and y' are FP' expressions such that $x' \rightarrow_a^* y'$, then $\rho(x') \rightarrow^* \rho(y')$.
2. If x is an FP redex and x' is a well-formed FP' expression such that $\rho(x') = x$ and $\mu'(x') \neq \perp$, then the FP-style rule corresponding to the FP rule for which x is a redex applies to x' , or x' is of the form $P_m(x'')$ and the FP-style rule applies to x'' .
3. If x and y are FP expressions such that $x \rightarrow y$ and x' is an FP' expression such that $\rho(x') = x$, then there exists an FP' expression y' such that $\rho(y') = y$ and $x' \rightarrow_a^* y'$ using exactly one FP-style or bottom rule.

Proof. It suffices to show that part (1) holds if $x' \rightarrow_a y'$ is an instance of an FP' rewrite rule. If $x' \rightarrow_a y'$ via a lowering rule or a P -rule, then it is easy to see that $\rho(y') = \rho(x')$, and we can just take $y = x$. For bottom rules the result is also immediate from the definition. If $x' \rightarrow_a y'$ via an FP-style rewrite rule, then it is easy to see that $\rho(x') \rightarrow \rho(y')$ via the corresponding FP rewrite rule.

For part (2), we check all the rewrite rules on a case-by-case basis. For example, suppose $\rho(x') = \text{comp} : \langle f, g \rangle : y$ and $\mu'(x') \neq \perp$. *A priori*, it is possible that x' is of the form $\text{comp}_k : P_m(\langle f, g \rangle) : y$, so the comp FP-style rule does not apply. However, if x' were of this form, then it would not be well-formed. Note that we added P -rules for pushing P_m past al and ar so that an expression such as $\text{first}_4 : (P_3(\text{al}_6 : \langle \text{id}_5, b \rangle))$ would not be well formed. By using P -rules, we can rewrite it to $\text{first}_4 : (\text{al}_6 : \langle \text{id}_2, b \rangle)$, which is well formed, and the appropriate FP-style rule applies.

For part (3), it suffices to show the result in case $x \rightarrow y$ is an instance of an FP rewrite rule. If $\mu(x') = \perp$, then $x' = \Omega(x)$ or $x \rightarrow_a \Omega(x)$ via a lowering rule, and $\Omega(x) \rightarrow_a \Omega(y)$ via a bottom rule. And if $\mu(x') \neq \perp$, then we can rewrite x' to a well-formed expression by

Proposition 5.7. If x'' is the resulting expression, then we clearly have $\rho(x'') = \rho(x')$. The result is now immediate from part (2). $\square$

FP' is not strongly normalizable or even weakly normalizable. Consider, for example, $\Omega(x)$, where x is any FP expression that is not weakly normalizable. Since $x \rightarrow y$ iff $\Omega(x) \rightarrow_a \Omega(y)$, we must have that $\Omega(x)$ is not weakly normalizable. However, it turns out that this is essentially the only counterexample to the strong normalizability of FP' . Proposition 5.7 already shows that lowering rules and P -rules by themselves cannot be applied infinitely often to an FP' expression. In the remainder of this section, we will show that FP-style rules - the ones we are mainly interested in - also cannot be applied infinitely often to an FP' expression. This motivates the following definition.

Definition 5.9. An FP' expression x' is in *essential normal form (e.n.f.)* iff the only $\rightarrow_a$ rules that apply are instances of bottom rules.

It is interesting to note that although no FP-style rules may apply to an expression in e.n.f., it may happen that an FP-style rule can be made to apply to an expression in e.n.f. after applying a bottom rule. For example, consider the expression $x' = al_4: \langle a, \Omega(id: \langle a \rangle) \rangle$. It is easy to check that x' is in e.n.f. However, we can rewrite x' to $x'' = al_4: \langle a, \Omega(\langle a \rangle) \rangle$, to which an FP-style rule does apply. This and the corresponding case with ar are the only sources of this unusual phenomenon.

We would like to show that every FP' expression can be rewritten to one in essential normal form. Note that this is essentially a weak normalization result. The strategy would then be to prove an analogue to Theorem 4.3 for FP' and pull this back to a strong completeness result for FP using Proposition 5.8.

Showing that typed λ -calculus is weakly normalizable is relatively straightforward. The idea is the following (cf. [Wa78]). The *degree* of a redex $(\lambda x.y)z$ is defined to be the type of $\lambda x.y$. We say an expression has *maximal degree* n iff it contains a redex of degree n , but no redexes of higher degree. Suppose we have an expression x of maximal degree n . It is not hard to show that if we reduce (i.e. apply a β -rewrite rule to) the rightmost redex in x of degree n , the resulting expression has one fewer redex of degree n , and no redexes of degree greater than n . By repeating this process, we end up with an expression of maximal degree at most $n - 1$. Continuing, we eventually get an expression of maximal degree 0, which must be in normal form.

We might hope that this idea would also work for FP' . However, there are a number of difficulties involved in applying it, mainly stemming from the added structure of FP rewrite rules. Suppose we define the degree of an FP-style redex of the form $f_n: x$ to be n , by analogy with the λ -calculus case, ignoring for now what to do for redexes involving $comp$, $cons$, and K . Consider an expression of the form $tl_4: (al_3: \langle a, \langle b \rangle \rangle)$. Now the only redex in this expression is $al_3: \langle a, \langle b \rangle \rangle$, and it has degree 3. But this expression rewrites to $tl_4: \langle a, b \rangle$,

which has degree 4. Thus, if we define the maximal degree of an FP' expression in the naive way, then the maximal degree of this expression goes from 3 to 4 (when we apply the only redex possible).

We can get around this problem, essentially by taking into account *potential redexes* as well as actual redexes when computing the maximal degree of an expression, where a potential redex is an expression that can be rewritten to a redex (so, for example, $tl_4:(al_3: \langle a, \langle b \rangle \rangle)$ is a potential redex although it is not a redex). However, there is a much more serious problem that arises due to primitive function symbols such as `comp` whose rewrite rules involve two arguments. Consider an expression such as $apply_4: \langle P_2(comp_7: \langle f, g \rangle), a \rangle$. The `apply` redex is the only one contained in the expression, and it has degree 4. But after rewriting this and applying a P -rule, we end up with the expression $P_1(comp_7: \langle f, g \rangle :a)$, which has a redex of degree 7!

It is possible, with some effort, to get a proof in the same spirit as the typed λ -calculus weak normalization proof which shows that every FP' expression can be reduced to one in essential normal form. We do not present this proof here. Instead, we will prove a stronger result, showing that every FP' expression is *essentially strongly normalizable* where:

Definition 5.10. An FP' expression is *essentially strongly normalizable* (e.s.n.) iff every reduction sequence for it has at most finitely many FP -style rules.

Remark: We cannot hope to show that every reduction sequence leads to an expression in e.n.f. To see this, suppose that x is an FP expression which is not weakly normalizable. Then starting with an expression of the form $first_n: \langle a, \Omega(x) \rangle$, we can keep applying bottom rules to the expression $\Omega(x)$ without ever applying the obvious FP -style rule. This gives us an infinite reduction sequence that does not lead to an expression in e.n.f.

Theorem 5.11. Every FP' expression is *essentially strongly normalizable*.

Proof. Not surprisingly, we prove our result by adapting techniques used to show that typed λ -calculus is strongly normalizable (cf. [FLO83]). However, just as in the case of weak normalizability, adapting the proof to FP requires some effort, mainly due to the greater structure of FP rewrite rules. The details can be found in Appendix F. $\square$

Corollary 5.12. For all FP' expressions x' , there exists an FP' expression y' such that $x' \rightarrow_a^* y'$ and y' is in essential normal form.

Proof. We first show that for every FP' expression z'_0 there exists an FP' expression z'_1 such that $z'_0 \rightarrow_a^* z'_1$ and either z'_1 is in e.n.f. or the reduction used at least one FP -style rule. By Proposition 5.7, $z'_0 \rightarrow_a^* z''$ for some well-formed z'' . If z'' is in e.n.f., then the claim follows by taking $z'_1 = z''$. If z'' is not in e.n.f., then z'' contains an FP -style redex (since it has a non-bottom rule redex and no P -rule or lowering rule redexes) and so $z'' \rightarrow_a z'_1$ by an FP -style rule and again the claim follows. Since Theorem 5.11 implies that no rewrite sequence has

infinitely many reductions involving FP-style rules, it easily follows from the claim that we can eventually rewrite every FP' expression x' to an expression y' in e.n.f. $\square$

We need one more fact before we can prove strong completeness for FP. The following is the FP' analogue of Theorem 4.3. Its proof is quite similar to that of Theorem 4.3, so we omit it here.

Proposition 5.13. *Let x' be an FP' expression such that x' is in essential normal form and let $x = \rho(x')$. Then:*

1. *if $\mu'(x') = a$ for some atomic item a , then $x = a$,*
2. *if $\mu'(x') = \langle w_1, \dots, w_n \rangle$, then x is of the form $\langle y_1, \dots, y_n \rangle$, $al: \langle y, z \rangle$, or $ar: \langle y, z \rangle$; if $\mu'(x')$ is of definite length, then in fact x must be of the form $\langle y_1, \dots, y_n \rangle$,*
3. *if $\mu'(x')$ is a function item, then x is either a primitive function symbol f , or of the form $K:y$, $comp:y$, or $cons:y$.*

We are finally ready to prove strong completeness for FP.

Theorem 5.14. *For all FP expressions x there is a transparent expression y such that $x \rightarrow^* y$.*

Proof. Suppose x is an FP expression such that $\mu(x)$ is a sequence item. By Proposition 5.3, there is an FP' expression x' such that $\rho(x') = x$ and $\mu'(x')$ is also a sequence item. If $\mu(x)$ is of definite length, then we can also assume without loss of generality that $\mu'(x')$ is of definite length. By Corollary 5.12, there is some y' such that $x' \rightarrow_a^* y'$ and y' is in e.n.f. Since $\rightarrow_a$ rewriting is sound, we must have $\mu'(y') = \mu'(x')$, and so $\mu'(y')$ is also a sequence item. Thus, by Proposition 5.13, $\rho(y')$ is of the form $\langle y_1, \dots, y_n \rangle$ or $al: \langle y_1, y_2 \rangle$ or $ar: \langle y_1, y_2 \rangle$, with the first case holding if $\mu(x)$ is of definite length. By Proposition 5.8, $x \rightarrow^* \rho(y')$, so we are done.

The argument in the case that $\mu(x)$ is a function item or an atomic item is completely analogous and left to the reader. $\square$

6. Good Strategies for Rewriting

Theorem 5.14 shows that we can always rewrite an expression to a transparent expression (and, as a corollary, always rewrite an expression whose meaning is an observable item to the corresponding observable expression). While this is an interesting theoretical result (and provides justification for our claim that we have provided enough rewrite rules for FP), it is not sufficient for an implementer of FP. Simply knowing that there is some way to rewrite an expression to a transparent expression is not enough. We need a good strategy for doing so. In this section we consider several notions of “good rewriting strategy”, characterize them in terms of FP' , and give examples of good rewriting strategies for FP.

We begin by defining the notion of a rewrite strategy. Intuitively, a rewrite strategy is an algorithm that, given an FP expression x , selects a subexpression of x which is a redex. The appropriate rewrite rule is then applied to this redex, and the process begins all over

again. Since the redex selected might depend on which reductions have already been performed, we allow the algorithm to depend on the entire history of rewriting. More formally, we have

Definition 6.1. A *rewrite strategy* R is a function $R : (\text{FP expressions}) \times \mathbb{N} \rightarrow (\text{FP expressions})$ such that $R(x, 0) = x$ for all FP expressions x , and for all n , either $R(x, n) \rightarrow R(x, n+1)$ or $R(x, n) = R(x, n+1)$. Thus $R(x, n)$ describes the n^{th} step of rewriting when starting with x according to strategy R . We write $x \rightarrow_R y$ iff for some n we have $R(x, n) = y$.

We say R is *history independent* iff, intuitively, it performs the same rewriting when it reaches an expression y starting from x as it does starting from y . More formally, R is history independent iff for all x and n , if $R(x, n) = y$ and $R(y, 1) = y'$, then $R(x, n+1) = y'$. Note that if R is history independent, then R is completely determined once we give $R(x, 1)$ for all FP expressions x .

R is said to *stop prematurely* if for some x and n , we have $R(x, n)$ not in normal form, but $R(x, n') = R(x, n)$ for all $n' \geq n$. If x is strongly normalizable, then it is easy to see that any rewrite strategy that does not stop prematurely will reduce x to normal form.

In obvious analogy to observable completeness, we say a strategy R is *O-safe* iff for any FP expression x such that $\mu(x)$ is an observable item, $x \rightarrow_R \mu(x)^e$. We also want an analogue to strong completeness. It would seem that the obvious definition is to say that R is *S-safe* iff $x \rightarrow_R y$ for some transparent expression y . Unfortunately, with this definition we do not get an analogue to Proposition 3.6; S-safety would not imply O-safety. For example, suppose we have an expression x such that $\mu(x) = \langle a \rangle$. For an O-safe strategy R , we have $x \rightarrow_R \langle a \rangle$. But if we define S-safe as suggested above, then if R is an "S-safe" strategy, we can conclude $x \rightarrow_R \langle y \rangle$ where $\mu(y) = a$, and that $y \rightarrow_R a$. However, we have no way of concluding that $x \rightarrow_R \langle a \rangle$, since, for example, R may stop prematurely at $\langle y \rangle$.

Intuitively, the problem is that although transparency guarantees that $x \rightarrow_R \langle y \rangle$, there is no guarantee that y itself is transparent. These observations motivate the following definition.

Definition 6.2.

1. An FP expression x is *1-transparent* iff it is transparent.
2. An FP expression x is *(n+1)-transparent* iff x is transparent and, if x is of the form $\langle x_1, \dots, x_k \rangle$, then x_i is *n-transparent* for $i = 1, \dots, k$, and if x is of the form $y : z$ and $\mu(x) \neq \perp$, then y, z are both *n-transparent*.

Definition 6.3. R is *S-safe for* x iff for all n , there exists an *n-transparent* y such that $x \rightarrow_R y$. R is *S-safe* iff R is S-safe for all FP expressions x .

An argument by induction on the depth of nesting of brackets, analogous to Proposition 3.6, can be used to show that

Proposition 6.4. Every S-safe rewrite strategy is also O-safe.

We remark that it follows by an easy induction on n that an expression in normal form is n -transparent for all n (using Theorem 4.3), and that for all x , there is some n -transparent y such that $x \rightarrow_a^* y$ (using Theorem 5.14). Thus, we can conclude that there does exist a (not necessarily recursive) S-safe strategy. In the remainder of this section we will show that there are also some natural (and recursive) S-safe strategies.

Observe that the two rewrite strategies perhaps most often considered in the literature, *leftmost-outermost* and *leftmost-innermost* (cf. [MNV73]), are *not* S-safe; in fact, they are not even O-safe. Consider the leftmost-outermost strategy, LO . LO is history independent and $LO(x, 1)$ is the result of applying the rewrite rule corresponding to the redex which starts lexically leftmost in x . To see that LO is not O-safe, let x be the expression $tl(al: \langle fix:id, id: \langle a \rangle \rangle)$. It is easy to check that the meaning of this expression is the observable $\langle a \rangle$. It is also easy to see that for all n , $LO(x, n)$ is of the form $tl(al: \langle y, id: \langle a \rangle \rangle)$, where y is the result of rewriting $fix:id$. Therefore the leftmost-outermost strategy does not rewrite this expression to $\langle a \rangle$, and hence is not O-safe.

The leftmost-innermost strategy chooses the leftmost redex which does not have another redex as a subexpression. The same example used above also shows that leftmost-innermost is not O-safe either.

It is interesting to note that we can modify the leftmost-outermost strategy to get a natural rewrite strategy which is not S-safe but *is* O-safe.

Definition 6.5. The *modified leftmost strategy* ML is history independent. Thus, in order to describe it, we need only give $ML(x, 1)$ for each FP expression x . If x is in normal form, then clearly $ML(x, 1) = x$. Otherwise we compute $ML(x, 1)$ by taking the first rule in the list below that applies:

1. if x is itself a redex, then $ML(x, 1)$ is (the unique) y such that $x \rightarrow y$ is an instance of a rewrite rule
2. if $x = \langle x_1, \dots, x_n \rangle$, then $ML(x, 1) = \langle x_1, \dots, x_{i-1}, ML(x_i, 1), x_{i+1}, \dots, x_n \rangle$ where i is such that x_i is not in normal form and x_j is in normal form for all $j < i$
3. if $x = y:z$ and y is not in normal form, then $ML(x, 1) = ML(y, 1):z$
4. if $x = al: \langle x_1, x_2 \rangle$ and x_2 is not in normal form, then $ML(x, 1) = al: \langle x_1, ML(x_2, 1) \rangle$
5. otherwise, x must be of the form $y:z$ with z not in normal form, and we take $ML(x, 1) = y:ML(z, 1)$.

Note that rule 4 is the only one that distinguishes modified leftmost and leftmost-outermost. It is also interesting to note that al and ar are not treated symmetrically in this rule.

It is easy to see that the modified leftmost strategy is not S-safe. Consider the expression $\langle fix:id, id:a \rangle$. The modified leftmost strategy will spend all its time working on $fix:id$, and never get around to rewriting $id:a$ to a . However, we have

Proposition 6.6. *The modified leftmost strategy is O-safe.*

Proof. See Appendix G. $\square$

We now characterize S-safe rewrite strategies in terms of FP' . This characterization plays a key role in our analysis of rewrite strategies.

Definition 6.7. A *tracking sequence* for (R, x) is a sequence $x'_0, x'_1, \dots$ of well-formed FP' expressions such that $\rho(x'_i) = R(x, i)$ and $x'_i \rightarrow_a^* x'_{i+1}$ using at most one FP' -style or bottom rewrite rule, for $i = 0, 1, 2, \dots$. We say that a tracking sequence *reaches essential normal form* iff some expression in the tracking sequence is in e.n.f.

Lemma 6.8. *If R is a rewrite strategy, x is an FP expression, and x' is a well-formed FP' expression such that $\rho(x') = x$, then there is a tracking sequence for (R, x) starting with x' .*

Proof. Immediate from Proposition 5.8 and Proposition 5.7. $\square$

Theorem 6.9. *The following are equivalent:*

1. *R is S-safe for x .*
2. *Every tracking sequence for (R, x) reaches e.n.f.*

Proof. The proof is quite technical, and is deferred to Appendix H. $\square$

We next define *fair* strategies, a large class of strategies that are provably S-safe. Intuitively, a fair strategy is one in which every redex either “disappears” or is eventually rewritten. In order to make this precise, we must first describe what it means for a given redex (called the *parent* redex) to have a *child*. Suppose $x \rightarrow y$ and x' is a subexpression of x which is a redex. Then a subexpression y' of y is a child of x' iff it is a redex that “corresponds” to x' in y ; in particular, both x' and y' are instances of the left-hand sides of the same rewrite rule and occupy the same “position” in x and y , respectively. Rather than giving a formal definition of child here, we give a more easily understood informal definition, supplemented by examples.

A child can occur in one of three ways:

1. The parent redex is disjoint from the rewritten redex. For example, if we take $x = \langle id:a, id:b \rangle$, $y = \langle id:a, b \rangle$, and $x' = id:a$, then x' is disjoint from the rewritten redex $id:b$, and the occurrence of $id:a$ in y is a child of x' .
2. The rewritten redex is a *subredex* of (i.e. subexpression of the instantiation of a schema variable in the rewrite rule schema of) the parent redex. For example, if we take $x = x' = first: \langle id:a \rangle$ and $y = y' = first: \langle a \rangle$, then the rewritten redex $id:a$ is a subredex of x' and y' is a child of x' .
3. The parent redex is a subredex of the rewritten redex. For example, if $x = cons: \langle z_1, z_2 \rangle : (id:a)$, $y = \langle z_1:(id:a), z_2:(id:a) \rangle$, and $x' = id:a$, then the parent redex x' is a subredex of x . In this case x' has two children in y ; i.e., the two occurrences of $id:a$.

Note that the rewritten redex never has any children and that in cases 1 and 2, the parent always has exactly one child. Also note, for example, that in the reduction $\text{first}:(\text{al}: \langle a, \langle b \rangle \rangle) \rightarrow \text{first}: \langle a, b \rangle$, no redexes have any children. Given a reduction sequence $x_0 \rightarrow \dots \rightarrow x_k$, we say that a subexpression y_k of x_k is a *descendant* of a subexpression y_0 of x_0 iff there are occurrences of redexes y_i in x_i such that y_i is a child of y_{i-1} for $i = 1, \dots, k$.

Definition 6.10. A rewrite strategy R is *fair* for x iff for all n and all redexes y that are subexpressions of $R(x, n)$, there exists $n' > n$ such that y has no descendants in $R(x, n')$. Intuitively, either the redex y was applied by the time $R(x, n)$ was rewritten to $R(x, n')$, or it “disappeared” (for example, if we rewrite $K:x_1:x_2 \rightarrow x_1$, any redexes that are subexpressions of x_2 disappear). R is *fair* iff R is fair for all FP expressions x .

Theorem 6.11. *If R is fair for x , then R is S-safe for x .*

Proof. Suppose R is fair for x . By Theorem 6.9, in order to show that R is S-safe for x , it suffices to show that every tracking sequence for (R, x) reaches e.n.f. So let $x'_0, x'_1, \dots$ be a tracking sequence for (R, x) . Since FP' is e.s.n., we know there must exist some n such that no FP-style rules are applied from x'_n on. Since x'_n is well formed, it must be the case that $x'_{n'+1}$ is the result of applying a bottom rule to some subexpression of x'_n for all $n' \geq n$. We claim that x'_n is actually in e.n.f. Suppose not. Then there is a subexpression y' of x'_n such that $\mu'(y') \neq \perp$ and $\rho(y')$ is an FP redex. Since R is fair, there must exist $n' > n$ such that $\rho(y')$ has no descendants in $R(y, n')$. We now get an immediate contradiction by applying the following lemma, which is proved by a case-by-case analysis of all the FP-style rewrite rules.

Lemma 6.12. *If $x' \rightarrow_a y'$ is an instance of a bottom rule, where x' is a well-formed FP' expression, and x'_0 is an FP-style redex that is a subexpression of x' , then there is a subexpression y'_0 of y' such that y'_0 is an FP-style redex and $\rho(y'_0)$ is a child of $\rho(x'_0)$.*

Proof. Left to the reader. (Note that cases 1 and 2 in the definition of *child* are the only ones that apply.) $\square$

Corollary 6.13. *Every fair strategy is S-safe.*

We are now in a position to describe a number of strategies that are provably fair and hence S-safe. These strategies are not history independent. The idea for each of them is that we proceed in phases: for all FP expressions x there is a sequence of numbers $n_0, n_1, \dots$, where $n_0 = 0$, but all the other numbers depend on x , such that all the redexes that appear in $R(x, n_i)$ have no descendants in $R(x, n_{i+1})$.

Definition 6.14. We define the strategies *breadth-first* (BF), *cons-last* (CN), *comp-last* (CM), and *apply-last* (AP) as follows. Each strategy proceeds in phases, where all the redexes that appear at the beginning of the i^{th} phase have no descendants by the end of the i^{th} phase.

- **Breadth-first:** Given x , suppose inductively we have found numbers $0 = n_0 \leq \dots \leq n_i$ such that all the redexes that appear in $BF(x, n_j)$, $j < i$, have no descendants in $BF(x, n_{j+1})$. We now define n_{i+1} and $BF(x, n)$ for $n_i \leq n \leq n_{i+1}$. Number the redexes in $BF(x, n_i)$ in any order. Suppose there are k such redexes. We then rewrite the first redex, then all the descendants of the second redex in the resulting expression, all the descendants of the third redex in that resulting expression, and so on up to the k^{th} redex. If this takes n' steps, then $n_{i+1} = n_i + n'$.
- **cons-last:** The $(i+1)^{\text{st}}$ phase consists of two subphases. In the first subphase we rewrite all the non-cons redexes (i.e., all those not of the form $\text{cons}:x:y$) until there are no more, while in the second subphase we rewrite all the cons redexes until there are no more (at this point, of course, we will in general have non-cons redexes appearing again). We omit the formal definition here.
- **comp-last:** CM is analogous to CN , except that we first rewrite the non-comp redexes, and then rewrite the comp redexes.
- **apply-last:** Again, AL is analogous to CN , except that we first rewrite the non-apply redexes.

It should be clear from the definitions that BF is indeed a fair strategy. The only problem in showing that CN is a fair strategy is that, *a priori*, it is not clear that n_{i+1} is finite; it may be the case that the $(i+1)^{\text{st}}$ phase as defined never terminates. However, from Theorem 4.2, we know that $FP\text{-}\{\text{cons}\}$ is strongly normalizable, so it is easy to see that the first subphase terminates in a finite number of steps. And since $FP\text{-}\{\text{comp}\}$ is strongly normalizable, it follows that the second subphase also terminates in a finite number of steps. Similar arguments can be used to show that CM and AP are fair strategies. Note that any hybrid of these strategies (for example, one where we alternate between CM and CN in consecutive phases) will also be fair.

7. Limiting Completeness

The notion of strong completeness we have defined has a strong syntactic component, since it requires that an expression with a certain meaning can be rewritten to another expression with a prescribed syntactic form. We would like to be assured that our rewriting also produces arbitrarily good semantic approximations. To understand this point, consider an expression x whose meaning is the limit of $\langle a, *, b \rangle$, $\langle a, a, *, b, b \rangle$, $\langle a, a, a, *, b, b, b \rangle$, ... Thus, the meaning of x is a double-sided infinite sequence with a 's added from the left and b 's added from the right. Strong completeness assures us that x can be rewritten to an expression of the form $al: \langle a, x_1 \rangle$ or $ar: \langle x_1, b \rangle$. Furthermore, it also assures us that in either case, x_1 can be rewritten to an expression of the form $al: \langle a, x_2 \rangle$ or $ar: \langle x_2, b \rangle$. We can continue this argument indefinitely. However, it is consistent with the definition of strong completeness that we can only rewrite x to an expression of the form $al: \langle a, al: \langle a, al: \dots \rangle \rangle$, without ever getting information about the right-hand sequence of b 's in $\mu(x)$. This seems somehow unsatisfactory. To handle this phenomenon, we are led to the notion of *limiting completeness*,

which is the FP analogue to the notion with the same name defined by Wadsworth for λ -calculus (cf. [Wa78]). Of course, once we have defined limiting completeness, we will consider the corresponding notion for rewrite strategies, which we call *L-safety*. Perhaps surprisingly, it turns out that in the context of FP, S-safety implies L-safety, although not conversely.

The intuitive idea behind limiting completeness in the λ -calculus is that we rewrite an expression for awhile, replace all the redexes in the resulting expression by a special symbol Ω (whose meaning is $\perp$), apply some obvious rewrite rules involving Ω (namely, $\Omega x \rightarrow \Omega$ and $\lambda x. \Omega \rightarrow \Omega$), and then consider the meaning of the resulting expression. Limiting completeness says that we can approximate the meaning of an expression arbitrarily well by this process.

We capture this in FP by defining a meaning function μ_r (the *r* stands for redex) which maps redexes to bottom. The relationship between the μ_r meaning of an expression and its syntactic structure is described in Proposition 7.2 below. We define μ_r by induction on structure:

$$\mu_r(x) = \mu(x) \text{ if } x \text{ is an atom or primitive function symbol}$$

$$\mu_r(\langle x_1, \dots, x_n \rangle) = \langle \mu_r(x_1), \dots, \mu_r(x_n) \rangle$$

$$\mu_r(x:y) = \begin{cases} \mu_r(x) : \mu_r(y) & \text{if } x:y \text{ is not a redex} \\ \perp & \text{if } x:y \text{ is a redex.} \end{cases}$$

The following lemma relates μ_r to μ and μ' .

Lemma 7.1.

1. $\mu_r(x) \leq_I \mu(x)$ for all FP expressions x .
2. $\mu_r(x) = \mu(x)$ if x is an FP expression in normal form.
3. If x' is an FP' expression in essential normal form, then $\mu'(x') \leq_I \mu_r(\rho(x'))$.
4. If $d \in \text{Fin}$, $d \leq_I \mu_r(x)$, then there exists a star-free FP' expression x' in essential normal form such that $\rho(x') = x$ and $d \leq_I \mu'(x')$.

Proof. Parts 1, 2, and 4 are proved by induction on the structure of x , while part 3 is proved by induction on the structure of x' . The details are straightforward. $\square$

The following proposition, which generalizes Theorem 4.3, relates the μ_r semantics to syntax. The proof is much like that of Theorem 4.3, so we omit it here.

Proposition 7.2.

1. If $\mu_r(x) = a$ for some atomic item a , then $x = a$.
2. If $\mu_r(x)$ is a sequence item, then x is of the form $al: \langle x_1, x_2 \rangle$, $ar: \langle x_1, x_2 \rangle$, or $\langle x_1, \dots, x_n \rangle$; if $\mu_r(x)$ has definite length, then in fact x has the form $\langle x_1, \dots, x_n \rangle$.
3. If $\mu_r(x)$ is a function item, then x is either a primitive function symbol or of the form $K:y, \text{comp}:y$, or $\text{cons}:y$.

Using μ_r , we can now capture the intuition of approximating the semantic meaning of an expression arbitrarily closely, and get another notion of having “enough” rules.

Definition 7.3. A set of rewrite rules has the *limiting completeness* property iff for all FP expressions x and all $d \in \text{Fin}$ such that $d \leq_I \mu(x)$ there is an FP expression y such that $x \rightarrow^* y$ and $d \leq_I \mu_r(y)$. A rewriting strategy R is *L-safe for x* iff for all $d \in \text{Fin}$ such that $d \leq_I \mu(x)$, there exists n such that $d \leq_I \mu_r(R(x, n))$. R is *L-safe* iff it is L-safe for all FP expressions x .

Remark: There is one intuition behind the notion of limiting completeness in the λ -calculus which is *not* captured by our definition of μ_r . To explain our point, we need a little notation. If M is a λ -calculus expression, let M_r be the result of replacing all redexes in M by Ω and then applying the rewrite rules $\Omega x \rightarrow \Omega$ and $\lambda x. \Omega \rightarrow \Omega$ until there are no more redexes of this form. If $M \rightarrow^* N$, then it is easy to see that M_r and N_r have the same structure and N_r is more defined than M_r ; i.e., N_r looks just like M_r except that some Ω 's in M_r may be replaced by other, more completely defined expressions. This is *not* the case in FP because of the greater structure of the rewrite rules. For example, consider the expression $x = \text{al} : \langle a, \text{id} : \langle a \rangle \rangle$. The only redex here is $\text{id} : \langle a \rangle$; thus $\mu_r(x) = \text{al} : \langle a, \perp \rangle = \langle a, * \rangle$. Had we taken the route in FP of replacing redexes by Ω , we would get $x_r = \text{al} : \langle a, \Omega \rangle$. However, x rewrites to $y = \text{al} : \langle a, \langle a \rangle \rangle$, which is a redex, so $\mu_r(y) = \perp$ and $y_r = \Omega$. Thus y_r is *less* defined than x_r . Moreover, y rewrites to $z = \langle a, a \rangle$, and clearly $z_r = z$. Thus z_r has a different structure from x_r , although $x \rightarrow^* z$. The root cause of this situation is that, due to the greater structure of FP rewrite rules, it is possible that $x \rightarrow y$ but $\mu_r(y) <_I \mu_r(x)$. In order to avoid this situation, we must not only replace redexes by Ω , but also *potential redexes* (expressions that rewrite to redexes). This leads us to define a semantic function μ_{pr} , which is just like μ_r except that it maps potential redexes to $\perp$ (the *pr* stands for *potential redex*). We investigate μ_{pr} further in Appendix J. While μ_r and μ_{pr} are distinct, similar definitions in the λ -calculus would not result in distinct notions because of the simpler structure of the β -rule.

Note that limiting completeness is just what we need to show that if the meaning of x is a double-sided infinite sequence with a 's added from the left and b 's added from the right, as in the example at the beginning of this section, we can rewrite x to an expression with arbitrarily many a 's and arbitrarily many b 's.

Theorem 7.4. *FP has the limiting completeness property.*

Proof. Given an FP expression x and $d \in \text{Fin}$ with $d \leq_I \mu(x)$, by Proposition 5.3, we can find an FP' expression x' such that $\rho(x') = x$ and $d \leq_I \mu'(x')$. By Theorem 5.11, we can find y' in e.n.f. such that $x' \rightarrow_a^* y'$. Since $\rightarrow_a$ rewriting is sound, we have $\mu'(y') = \mu'(x')$, so that $d \leq_I \mu'(y')$. Since y' is in e.n.f., by part 3 of Lemma 7.1, we know that $\mu'(y') \leq_I \mu_r(\rho(y'))$, so that $d \leq_I \mu_r(\rho(y'))$. Finally, by Proposition 5.8, $x \rightarrow^* \rho(y')$. $\square$

Theorem 7.5. *L-safety implies O-safety.*

Proof. Suppose R is L-safe and $\mu(x)$ is observable. Thus $\mu(x) \in \text{Fin}$ and since R is L-safe, there must be some y such that $x \rightarrow_R^* y$ and $\mu(x) \leq_I \mu_r(y)$. An easy induction on the number of brackets ($\langle \rangle$'s) in $\mu(x)$, using Proposition 7.2, can now be used to show that $y = \mu(x)^e$. Thus $x \rightarrow^* \mu(x)^e$, and R is O-safe. $\square$

The converse to Theorem 7.5 does not hold. The modified leftmost strategy is O-safe, but the same example that we used to show it is not S-safe, namely the expression $\langle \text{fix}; \text{id}, \text{id}; a \rangle$, also shows that it is not L-safe.

In order to show that S-safety implies L-safety, we characterize L-safety in terms of FP' tracking sequences, in the same spirit as our characterization of S-safety. In order to formulate our characterization, we need to define a generalization of $\rightarrow_a$ rewriting for FP' (which is also used in Appendix F).

Definition 7.6. The rewrite rules for $\rightarrow_b$ consist of all the rewrite rules of $\rightarrow_a$ together with:

$$x' \rightarrow_b P_m(x')$$

$$x' \rightarrow_b \Omega(\rho(x'))$$

Although $\rightarrow_b$ rewriting is clearly not sound, it is easy to see that if $x' \rightarrow_b^* y'$, then $\mu'(y') \leq_I \mu'(x')$ (cf. Lemma 14.2).

By analogy to the notion of a tracking sequence, we define a $\rightarrow_b$ tracking sequence for (R, x) to be a sequence $x'_0, x'_1, \dots$ of well-formed FP' expressions such that $\rho(x'_n) = R(x, n)$ and $x'_i \rightarrow_b^* x'_{i+1}$ using at most one FP -style rule or bottom rule.

Theorem 7.7. *Let R be a rewrite strategy. Then the following are equivalent:*

1. R is L-safe for x .
2. For all $d \in \text{Fin}$ with $d \leq_I \mu(x)$, there exists a $\rightarrow_b$ tracking sequence $x'_0, x'_1, \dots$ for (R, x) such that for some n , x'_n is in e.n.f. and $d \leq_I \mu'(x'_n)$.

Proof. See Appendix I. $\square$

Corollary 7.8. *If R is S-safe for x , then R is L-safe for x . Thus S-safety implies L-safety.*

Proof. Suppose R is S-safe for x . Thus, given $d \in \text{Fin}$ with $d \leq_I \mu(x)$, choose x'_0 such that $d \leq_I \mu(x'_0)$ and $\rho(x'_0) = x$ and x'_0 is well-formed (such an x'_0 exists by Proposition 5.3 and Proposition 5.7). Consider any $(\rightarrow_a)$ tracking sequence for (R, x) starting with x'_0 , say $x'_0, x'_1, \dots$. By Theorem 6.9, there is some n such that x'_n is in e.n.f. Since $\rightarrow_a$ rewriting is sound, $d \leq_I \mu'(x'_n)$. Since d was chosen arbitrarily, by the characterization of L-safety given in Theorem 7.7, it follows that R is L-safe for x . $\square$

Of course, it is immediate from the corollary that all the fair strategies described in the previous section are also L-safe. The converse of Corollary 7.8 is false as the next example shows.

Example 7.9. Let $x = \text{comp}: \langle K:a, \text{id}:b \rangle$. Then x is 2-transparent but not 3-transparent because of the subexpression $\text{id}:b$. On the other hand, $\mu(x) = \mu(K:a) = \mu_r(x)$. Let R be a strategy that does nothing on x , i.e., $R(x, n) = x$ for all n . Then R is L-safe for x but not S-safe for x .

8. Conclusions

We have answered the two questions posed in the introduction. We have given a denotational semantics for FP where distinct observable expressions get different meanings, shown that the rewrite rules are sound for that semantics, and shown that they are complete in three different senses. For each of our notions of completeness, we have defined a corresponding notion of a “good” rewrite strategy. We have characterized and compared these notions, and given natural examples of good strategies. While we have proved these results in the context of FP, we believe that our techniques will readily apply to other languages.

Throughout this paper we have not discussed, nor have we needed to discuss, the Church-Rosser property for FP. (CR would say that for all FP expressions x , y_1 , and y_2 , if $x \rightarrow^* y_1$ and $x \rightarrow^* y_2$, then there exists z such that $y_1 \rightarrow^* z$ and $y_2 \rightarrow^* z$.) We conjecture that FP with the particular set of rewrite rules we have described is in fact CR, but we have not yet proved this. But even if we had proved CR, it is not clear how interesting it would be.

To understand this point, first note that CR for expressions whose meaning is observable follows immediately from our results on observable completeness. Indeed, observable completeness assures us that CR holds for observables in the following strong sense: We can add arbitrarily many new (sound) rules to the system and still be assured that expressions with observable meanings are CR. This is because the addition of rules necessarily preserves observable completeness (as well as the other properties we have considered), and observable completeness implies CR for expressions with observable meanings.

Our feeling is that any basic property of the language should be *robust*, in the sense that it is preserved under the addition of sound rewrite rules. But consider what happens when we add the following two (sound) rules to our system:

$$\text{comp}: \langle \text{comp}: \langle f, g \rangle, \text{comp}: \langle h, i \rangle \rangle \rightarrow \text{comp}: \langle f, \text{comp}: \langle g, \text{comp}: \langle h, i \rangle \rangle \rangle$$

$$\text{comp}: \langle \text{comp}: \langle f, g \rangle, \text{comp}: \langle h, i \rangle \rangle \rightarrow \text{comp}: \langle \text{comp}: \langle \text{comp}: \langle f, g \rangle, h \rangle, i \rangle$$

(Note that the soundness of these rules follows from the associativity of composition; it is perhaps more easily seen if we rewrite the rules using infix notation as $(f \circ g) \circ (h \circ i) \rightarrow f \circ (g \circ (h \circ i))$ and $(f \circ g) \circ (h \circ i) \rightarrow ((f \circ g) \circ h) \circ i$ respectively.) It is easy to see that if f, g, h , and i are replaced by primitive function symbols, then both of the above right hand sides are themselves in normal form. Thus, even if CR held for the system using the rewrite rules described in Figure 1, it would not be a robust property.

While the language we have presented is universal, it is by no means sufficient as a useful programming language. There are many more functions to be added, some defined in terms

of the primitives given and some as new primitives, and there are useful extensions that could be made to some of our functions. Although the addition of new primitives will necessarily make our proofs longer (there being more cases to consider and rules to check), we believe that the structure of the proofs is such that it will be relatively easy to make modifications to the language. In fact, in the development of this paper we have had occasion to make various perturbations in the definitions of the primitive functions, and we have found the proofs of completeness and safety to be remarkably insensitive to such modifications. Moreover, though it is computationally Spartan, we believe that the language already includes instances of all the essential complexities. For example, it might be useful to extend the meaning of `comp` to allow the rule `comp:(al: < f, g >):x → f:(comp:g:x)`, but the complications that would result from such an extension have already been considered in our treatment of `cons`.

At the same time, while we don't expect that such additions will pose any significant mathematical difficulties for our proof technique, there is a great deal of interesting work to be done to make the language more usable computationally. We are presently engaged in developing an extension of this kernel language that will have the same richness of algebraic properties as the original language of [Ba78]. As we extend the language, we hope to be able to use the degree to which these mathematical considerations are complicated as a criterion for choosing among various alternative extensions.

Acknowledgements: We would like to thank John Backus, David Chase, Ron Fagin, Matt Kaufmann, Albert Meyer, John Mitchell, Mike O'Donnell, Gordon Plotkin, Moshe Vardi, Tim Winkler, David Wise, and the members of IFIP WG 2.2 for their useful comments and criticisms.

9. Appendix A: Properties of D_{N+1}

We want to prove clauses 5-7 in the construction of D_{N+1} .

For clause 5, let X be a bounded, directed set in D_{N+1} . The proof breaks into cases depending on the type of items in X . If X contains an atom a , then because it is directed, X must either be $\{\perp, a\}$ or $\{a\}$, and in either case a least upper bound is a . If X contains a sequence item, then all of its elements must be sequence items or $\perp$ (since function items and sequence items are incomparable with respect to $\leq_{N+1}$), and any upper bound of X must also be a sequence item. We only consider the case where X has no sequence items of definite length and leave the case with sequence items of definite length to the reader. Since X is bounded there must be a smallest i and j so that every member of X has at most i components to the left of the $*$ and at most j components to the right of the $*$. We leave it to the reader to check that X has a least upper bound of the form $\langle y_1, \dots, y_i, *, y_{i+1}, \dots, y_{i+j} \rangle$ where each of the y 's is the least upper bound of the appropriate component set. (Note that we need that X is directed since $\{\langle a, * \rangle, \langle *, a \rangle\}$ has no least upper bound.)

If X consists of function items, define $x: D_N \rightarrow D_N$ by $x(y) \in \sqcup \{x':_{N+1} y \mid x' \in X\}$. x is well-defined since the set is a bounded, directed subset of D_N . We leave it to the reader to check that x is a continuous function and hence an element of D_{N+1} . x is clearly a least upper bound of X .

We leave it to the reader to show that $\cdot_{N+1}$ is continuous in its first argument (using a proof similar to the proof that a bounded, directed set of function items has a least upper bound) and that $\cdot_{N+1}$ is continuous in its second argument (using the fact that P_m^N is continuous).

Finally, for clause 7, we must show that P_m^{N+1} , $m \leq N+1$ is continuous. We proceed by induction. For $m=0$ the argument is obvious. If $m > 0$, let X be a directed set in D_{N+1} with $x' \in \sqcup X$. Let $Y = \{P_m^{N+1}(x) \mid x \in X\}$. It is easy to check that P_m^{N+1} is monotonic. Thus Y is directed and bounded by $P_m^{N+1}(x')$. Since Y is a bounded, directed subset of D_m , it has a least upper bound by clause 5. (The existence of a least upper bound is the principal reason for including clause 5 in the induction hypothesis.) We want to show that $P_m^{N+1}(x') \in \sqcup Y$. We focus on the case where x' is a function item and leave the other cases to the reader. For all $z \in D_{m-1}$, we have that

$$\begin{aligned} & P_m^{N+1}(x') \cdot_{N+1} z \\ &= P_{m-1}^N(x' \cdot_{N+1} z) \text{ (by definition of } P_m^{N+1}) \\ &\in \sqcup \{P_{m-1}^N(x \cdot_{N+1} z) \mid x \in X\} \text{ (since } P_{m-1}^N \cdot_{N+1} \text{ are continuous)} \\ &\equiv \sqcup \{P_m^{N+1}(x) \cdot_{N+1} z \mid x \in X\} \text{ (by definition of } P_m^{N+1}) \\ &\equiv (\sqcup \{P_m^{N+1}(x) \mid x \in X\}) \cdot_{N+1} z \text{ (since } \cdot_{N+1} \text{ is continuous and } \sqcup \{P_m^{N+1}(x) \mid x \in X\} \text{ exists)} \end{aligned}$$

Since $P_m^{N+1}(x') \cdot_{N+1} z \equiv (\sqcup Y) \cdot_{N+1} z$ for all $z \in D_{m-1}$, we have that $P_m^{N+1}(x') \in \sqcup Y$, as desired. This completes our proof.

10. Appendix B: Finite Elements in D_n

Our primary goal in this section is to prove Proposition 2.4, which states that every item is the limit of its finite approximations. In order to do this, it is helpful to define for all m a subset Fin'_m of Fin_m that essentially contains all the elements of Fin_m (in that for all $y \in Fin_m$ there exists some $y' \in Fin'_m$ such that $y \equiv_m y'$). The elements of Fin'_m are easy to characterize, and this characterization will be useful in our proofs. The only subtlety comes in characterizing the function items in Fin'_m , $m \geq 1$. These turn out to be “step functions” of the form $[x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k]$ where $x_1, \dots, x_k, y_1, \dots, y_k \in Fin'_{m-1}$ and for all $z \in D_{m-1}$, we define $[x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k](z) = y_i$, where x_i is the best approximation (among $x_1, \dots, x_k$) to z . In order to guarantee that the notion of “best approximation” is well defined, we need the following definition.

Definition 10.1. $M \subseteq D_n$ has best approximations in D_n iff $M \cap \{x' \in D_n \mid x' \leq_n x\}$ is directed for all $x \in D_n$.

Note that if M is finite and has best approximations in D_n , then for all $x \in D_n$, if there is some $z \in M$ such that $z \leq_n x$, then there is a *best approximation* y to x in M , i.e., $y \in M$ and if $y' \in M$ and $y' \leq_n x$, then $y' \leq_n y$. In particular, if $\perp \in M$, then for all $x \in D_n$ there is a best approximation to x in M .

We now define Fin'_m by induction on m , taking $Fin'_0 = D_0$, and $Fin'_{m+1} \subseteq D_{m+1}$ to consist of Fin'_m together with all the items of the form:

1. $\langle y_1, \dots, y_k \rangle$, where $y_i \in Fin'_m \cup \{*\}$ for $i = 1, \dots, k$
2. $[x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k]$ where
 - a. $\{x_1, \dots, x_k, y_1, \dots, y_k\} \subseteq Fin'_m$
 - b. $\{x_1, \dots, x_k\}$ has best approximations in D_m
 - c. $x_j' \leq_m x_j$ implies $y_j' \leq_m y_j$
 - d. $y_j' \equiv_m y_j$ implies $y_j' = y_j$,
and for $z \in D_m$, we define

$$[x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k](z) = \begin{cases} \perp & \text{if there does not exist } j \text{ such that } x_j \leq_m z \\ y_j & \text{if } x_j \text{ is a best approximation in } \{x_1, \dots, x_k\} \text{ to } z. \end{cases}$$

This completes the definition of Fin'_{m+1} . We must check that $h = [x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k]$ is a well-defined, continuous function. First note that although it may be the case that there is more than one best approximation to z in $\{x_1, \dots, x_k\}$, clauses 2(c) and 2(d) above guarantee that $h(z)$ is well defined. Now assume that $z \in \sqcup Z$, where Z is a directed subset of D_m . If there does not exist j such that $x_j \leq_m z$, then $h(z) = \perp$ and $h(z') = \perp$ for all $z' \in Z$ so $h(z) \in \sqcup \{h(z') \mid z' \in Z\}$. So assume that x_j is a best approximation to z in $\{x_1, \dots, x_k\}$. Since x_j is finite, $x_j \leq_m z_0$ for some $z_0 \in Z$. Hence, $h(z) = y_j = h(z_0)$. Thus, $h(z) \in \sqcup \{h(z') \mid z' \in Z\}$ by clause 2(c) above.

We are now ready to prove a result from which Proposition 2.4 follows as an immediate corollary.

Proposition 10.2. *For all $m \geq 0$, we have*

1. $Fin'_m \subseteq Fin_m$
2. *if X is a finite subset of Fin'_m , then there exists a finite subset $X' \supseteq X \cup \{\perp\}$ of Fin'_m which has best approximations in D_m*
3. *for all $y \in D_m$, the set $Fin'_m(y) = \{y' \leq_m y \mid y' \in Fin'_m\}$ is directed*
4. *for all $y \in D_m$, $y \in \sqcup Fin'_m(y)$*
5. *for all $y \in Fin_m$, there exists $y' \in Fin'_m$ such that $y \equiv_m y'$.*

Proof. We proceed by induction on m . The case $m = 0$ is straightforward. For $m > 0$, suppose $x \in Fin'_m$. If x is an atom or a sequence item, the fact that $x \in Fin_m$ follows from the induction hypothesis. If x is a function item, then x is of the form $[x_1 \rightarrow y_1, \dots, x_k \rightarrow y_k]$. Suppose Z is a directed subset of D_m , $z' \in \sqcup Z$, and $x \leq_m z'$. Thus $y_j \leq_{m-1} z' :_m x_j$ for $j = 1, \dots, k$. By the continuity of $:_m$, it follows that $z' :_m x_j \in \sqcup \{z :_m x_j \mid z \in Z\}$. Since y_j is finite, we must have $y_j \leq_m z_j :_m x_j$ for some $z_j \in Z$, $j = 1, \dots, m$. And since Z is directed, there is some $z_0 \in Z$ such that $z_j \leq_m z_0$, $j = 1, \dots, m$. Thus $x \leq_m z_0$. Since Z was chosen arbitrarily, it follows that $x \in Fin_m$.

To prove part 2, given X , a finite subset of Fin'_m , let X_{atom} , X_{seq} , and X_{func} consist, respectively, of the atoms, sequence items, and function items in X . It clearly suffices to find sets X'_{atom} , X'_{seq} , and X'_{func} with the appropriate properties. We can simply take $X'_{atom} = X_{atom} \cup \{\perp\}$. To find X'_{seq} , let Y consist of all components of sequences in X_{seq} . Thus Y is a finite subset of Fin'_{m-1} . By the induction hypothesis, there is a finite subset Y' of Fin'_{m-1} containing $Y \cup \{\perp\}$ which has best approximations in D_{m-1} . Suppose the longest sequence in X_{seq} has (either definite or indefinite) length N . Let X' consist of all sequences (all of whose components are members of Y') that either have definite length at most $2N$ or have indefinite length and at most N components to the left of the $*$ and at most N components to the right of the $*$. We leave it to the reader to check that $X'_{seq} \cup \{\perp\}$ has the required properties. Finally, to find X'_{func} , let Y (resp. Z) be the set of all y (resp. z) such that a function of the form $[\dots, y \rightarrow z, \dots] \in X_{func}$. Clearly Y and Z are finite subsets of Fin'_{m-1} , so, by the induction hypothesis, there exist finite subsets Y' and Z' of Fin'_{m-1} containing $Y \cup \{\perp\}$ and $Z \cup \{\perp\}$ respectively such that Y' and Z' have best approximations in D_{m-1} . Let X''_{func} be the set of all elements of Fin'_m of the form $[y_1 \rightarrow z_1, \dots, y_n \rightarrow z_n]$ where $\{y_1, \dots, y_n\} \subseteq Y'$ and $\{z_1, \dots, z_n\} \subseteq Z'$. We leave it to the reader to show that $X'_{func} = X''_{func} \cup \{\perp\}$ has the required properties.

To prove property 3, suppose $y \in D_m$ and $y_1, y_2 \in Fin'_m(y)$. By property 2, there exists a finite set Y such that $\{\perp, y_1, y_2\} \subseteq Y \subseteq Fin'_m$ and Y has best approximations in D_m . In particular, there is a best approximation to y in Y , say y_3 . Thus $y_3 \in Fin'_m(y)$, and $y_1, y_2 \leq_m y_3$. Since y_1 and y_2 were chosen arbitrarily, we have shown that $Fin'_m(y)$ is directed.

Property 4 is immediate if y is an atom and follows readily from the induction hypothesis if y is a sequence item. If y is a function item, clearly y is an upper bound of $Fin'_m(y)$. To show it is a least upper bound, let y' be any other bound. By the continuity of y and induction hypothesis 4, it suffices to show that $y \cdot_m z \leq_{m-1} y' \cdot_m z$ for all $z \in Fin'_{m-1}$. So let $z \in Fin'_{m-1}$ and let $t \in Fin'_{m-1}(y \cdot_m z)$. Then $[z \rightarrow t] \in Fin'_m(y)$, so $[z \rightarrow t] \leq_m y'$. Thus, $t \leq_m y' \cdot_m z$. Since $t \in Fin'_{m-1}(y \cdot_m z)$ was arbitrary, $y' \cdot_m z$ is an upper bound of $Fin'_{m-1}(y \cdot_m z)$. By induction hypothesis 4, $y \cdot_m z \leq_{m-1} y' \cdot_m z$ as desired.

Finally, for property 5, suppose $y \in Fin_m$. By property 4, $y \in \sqcup Fin'_m(y)$. Since y is finite, there must be some $y' \in Fin'_m(y)$ such that $y \leq_m y'$. By definition of $Fin'_m(y)$, we also have $y' \leq_m y$. Thus $y' \equiv_m y$. $\square$

This completes the proof of Proposition 2.4. We conclude this appendix with two examples showing that D lacks a number of properties enjoyed by D_m . In particular, we show that $:$ is not continuous in its first argument and that bounded, directed sets in D do not necessarily have least upper bounds in D . The fact that $:$ is not continuous follows immediately from the following example of an element of Fin_2 which is not a finite element of D .

Example 10.3. Let $nest_0 = \perp$ and $nest_{i+1} = \langle nest_i \rangle$. Let $f_i = [nest_i \rightarrow a]$ and let $g_i = [f_i \rightarrow b]$. Let $h = [[\perp \rightarrow \perp] \rightarrow b]$. Note that $h \in Fin_2$, $nest_i \in Fin_i$, $f_i \in Fin_{i+1}$, and $g_i \in Fin_{i+2}$ for all i . Moreover, $\{..., nest_i, ...\}$ is an increasing sequence, so $\{..., f_i, ...\}$ is a decreasing sequence, and $\{..., g_i, ...\}$ is an increasing sequence. Also note that $f_i \cdot x = \perp$ for all $x \in D_{i-1}$. Thus, $P_i(f_i) \equiv_i [\perp \rightarrow \perp]$. Clearly, h is an upper bound of the directed set $Y = \{g_i \mid i \in \mathbb{N}\}$. Notice that $h \not\leq_I g_k$ since $h : [\perp \rightarrow \perp] = b$ but $g_k : [\perp \rightarrow \perp] = \perp$. Let h' be an upper bound of Y . Hence, $h' \in (D_k \rightarrow D_k)$ for some k . Thus, $b = g_k : f_k \leq_I h' : f_k = h' (P_k(f_k)) = h' ([\perp \rightarrow \perp])$. Hence, $h \leq_I h'$. Thus, h is a least upper bound of Y , and so cannot be a finite element of D . Since $g_i : [\perp \rightarrow \perp] = \perp$, while $h : [\perp \rightarrow \perp] = b$, we have that $:$ is not continuous in its first argument (whereas, by contrast, we have shown that $:_n$ is for all n). Also note this example shows that P_2 is not continuous on D , since $P_2(g_i) = [[\perp \rightarrow a] \rightarrow b]$ while $P_2(h) = h$.

Example 10.4. We now show that bounded, directed sets in D do not necessarily have least upper bounds in D . Define $deep_0(x) = x$ and $deep_{j+1}(x) = \langle deep_j(x) \rangle$ for any $x \in D_0$. For $i \geq 1$, let $f_i = [deep_1(a) \rightarrow b, \dots, deep_i(a) \rightarrow b]$, and let $g_i = [deep_1(a) \rightarrow b, \dots, deep_{i-1}(a) \rightarrow b, deep_i(\perp) \rightarrow b]$. Since $P_i(deep_j(x)) = deep_i(\perp)$ for $i < j$, each g_k is an upper bound of the directed set $X = \{f_i \mid i \in \mathbb{N}\}$. Suppose X had a least upper bound h . Then $h \in (D_k \rightarrow D_k)$ for some k . Since h is a least upper bound, $h \leq_I g_k$. Furthermore, $b = f_{k+1} : deep_{k+1}(a) \leq_I h : deep_{k+1}(a) = h(P_k(deep_{k+1}(a))) = h(deep_k(\perp))$. Therefore, $g_k \leq_I h$, so we have $h \equiv_I g_k$. Notice that $g_k \not\leq_I g_{k+1}$ since $g_k : deep_k(c) = b$ while $g_{k+1} : deep_k(c) = \perp$. But this contradicts the fact that h is a least upper bound since g_{k+1} is an upper bound of X and $h \not\leq_I g_{k+1}$.

11. Appendix C: Properties of D^*

Our primary goal is to prove Proposition 2.6, i.e., given a directed set $X \subseteq D$, we want to show that $X^{dl} \in D^*$. Thus we must show that X is limit closed, downward closed, and directed. It is easy to see that X^d is downward closed and directed. The fact that X^{dl} is downward closed and limit closed follows from the following lemma.

Lemma 11.1. *If X is downward closed, then X^l is downward closed and limit closed.*

Proof. Let $x \in D_k$. By Proposition 2.4 and the fact that X is downward closed, $x \in X^l$ iff $\text{Fin}_k(x) \subseteq X$. If $x \leq_I y \in D_k$, then $\text{Fin}_k(x) \subseteq \text{Fin}_k(y)$ and so $y \in X^l$ implies that $x \in X^l$. Thus, X^l is downward closed. Since X and X^l are downward closed, $x \in (X^l)^l$ iff $\text{Fin}_k(x) \subseteq X^l$ iff $\text{Fin}_k(x) \subseteq X$ iff $x \in X^l$. Thus $(X^l)^l = X^l$, so X^l is limit closed. $\square$

Finally, we want to show that X^{dl} is directed by proving that any two elements of X^{dl} have an upper bound in X^{dl} . We do this by proving a stronger result: not only does every pair of elements in X^{dl} have an upper bound in X^{dl} , but every pair of elements $x_1, x_2 \in X^{dl} \cap D_k$ has a *least upper bound with respect to $X^{dl} \cap D_k$* ; i.e., there is some $y \in X^{dl} \cap D_k$ such that $x_1, x_2 \leq_I y$ and if $x_1, x_2 \leq_I z, z \in X^{dl} \cap D_k$, then $y \leq_I z$.

Lemma 11.2. *Suppose that X_t is a directed subset of D_{k-1} for each $t \in D_{k-1}$, such that (a) every pair of elements of $X_t^{dl} \cap D_{k-1}$ has a least upper bound with respect to $X_t^{dl} \cap D_{k-1}$ and (b) $X_{t'} \subseteq X_t$ whenever $t' \leq_{k-1} t$. Let $f_1, f_2 \in D_k$. If $f_3(t)$ is a least upper bound of $f_1 : t, f_2 : t \in X_t^{dl}$ with respect to $X_t^{dl} \cap D_{k-1}$ for all $t \in D_{k-1}$, then $f_3 \in D_k$.*

Proof. First we show that f_3 is monotonic. Let $t' \leq_{k-1} t$. Let u be a least upper bound of $f_1 : t', f_2 : t'$ with respect to $X_{t'}^{dl} \cap D_{k-1}$. Thus, $u \leq_{k-1} f_3(t')$ since $f_3(t') \in (X_{t'})^{dl} \subseteq (X_t)^{dl}$. Thus, $u \in (X_t)^{dl}$ since $(X_t)^{dl}$ is downward closed. Therefore, $f_3(t') \leq_{k-1} u$ and so $u \equiv f_3(t')$. But $u \leq_{k-1} f_3(t)$ by the definition of u . Thus, f_3 is monotonic. Let Y be a directed subset of D_{k-1} and let y be a least upper bound of Y . Let $Z = \{f_3(y') \mid y' \in Y\}$. Since f_3 is monotonic, Z is a directed subset of D_{k-1} bounded by $f_3(y)$. Therefore, Z has a least upper bound $z \leq_{k-1} f_3(y)$ and so $z \in X_y^{dl}$. Thus, $f_3(y) \leq_{k-1} z$ since z is an upper bound of $f_1 : y$ and $f_2 : y$ by the continuity of $:$ on D_k . Therefore, $z \equiv f_3(y)$ and so f_3 is continuous and therefore an element of D_k . $\square$

Definition 11.3. For $X, Y \subseteq D$, let $X :: Y = \{x : y \mid x \in X, y \in Y\}$.

Lemma 11.4. *If X is directed, $x \in D_k$, and $x : t \in (X :: \{t\}^d)^{dl}$ for all $t \in D_{k-1}$, then $x \in X^{dl}$.*

Proof. Let $t \in \text{Fin}_k(x)$. Then $t \equiv [u_1 \rightarrow v_1, \dots, u_n \rightarrow v_n]$ for some $u_1, v_1, \dots, u_n, v_n \in \text{Fin}_{k-1}$. Note that $v_i = t : u_i \leq_I x : u_i \in (X :: \{u_i\}^d)^{dl}$. Thus, $v_i \in (X :: \{u_i\}^d)^{dl}$ since v_i is finite. Therefore, $v_i \leq_I y_i : u_i$ for some $y_i \in X$. Since X is directed, we can find $y \in X$ such that $y_1, \dots, y_n \leq_I y$. Therefore, $v_i \leq_I y : u_i$ and so $t \leq_I y \in X$. Therefore, $t \in X^d$. Thus, $\text{Fin}_k(x) \subseteq X^d$ and so $x \in X^{dl}$ by Proposition 2.4. $\square$

Lemma 11.5. *If X is a directed subset of D and $x_1, x_2 \in X^{dl} \cap D_k$, then there is a least upper bound of $\{x_1, x_2\}$ with respect to $X^{dl} \cap D_k$.*

Proof. We prove the result for all directed sets X by induction on k . If $k = 0$, the result easily follows since $D_0 \cap X^{dl} = D_0 \cap (\{\perp\} \cup X)$. For the general case, the proof splits into cases, depending on whether X contains an atom, a sequence item, or a function item. If X contains an atom, the result is trivial, while if it contains a sequence item, the result follows using the induction hypothesis. (In the case of sequence items, we rely on the fact that X is directed to choose among various minimal upper bounds. For example, if $x_1 = \langle a, * \rangle$ and $x_2 = \langle *, a \rangle$, then X must contain either $\langle a \rangle$ or $\langle a, *, a \rangle$.) If X contains a function item, then given $x_1, x_2 \in X^{dl} \cap D_k$, we can assume that they are both function items (since X is directed). For any $t \in D_{k-1}$, note that $X::\{t\}^d$ is directed (since X is) and if $x_i \in X^{dl}$, then $x_i:t \in (X::\{t\}^d)^{dl}$ (since $:$ is continuous on D_k). Thus, by the induction hypothesis, define $x_3(t)$ to be a least upper bound of $x_1:t, x_2:t$ with respect to $(X::\{t\}^d)^{dl}$. $x_3 \in D_k$ by Lemma 11.2. $x_3 \in X^{dl}$ by Lemma 11.4. x_3 is a least upper bound of x_1, x_2 with respect to $X^{dl} \cap D_k$ by construction. $\square$

This completes the proof of Proposition 2.6. We now consider two examples that show the necessity of some of the choices we made. The first shows that if X is a directed set, then in general X^{dl} is *not* an element of D^* . Thus the order in which we take the limit closure and downward closure is essential. The second example shows that if $X, Y \in D^*$ then $X::Y$ is not necessarily limit closed, and so may not be an element of D^* . Thus limit closure is needed in the definition of $X:Y$.

Example 11.6. Let $nest_0 = \perp$ and $nest_{i+1} = \langle nest_i \rangle$. Let $seq_j(x) = \langle x, \dots, x, * \rangle$ where there are j occurrences of x . For $j \geq 1$, let (using the notation of Appendix B) $f_j = [seq_1(a) \rightarrow seq_1(nest_1), \dots, seq_j(a) \rightarrow seq_j(nest_j)]$. Note that $f_j \in Fin_{j+2}$. Furthermore, if $f_j \leq_I g$, then $g \notin D_{j+1}$. Let $X = \{f_j \mid j \in \mathbb{N}\}$. Clearly $f_j \leq_I f_k$ if $j \leq k$, so X is directed. We claim that $X^l = \{g \mid \exists j(g \equiv_I f_j)\}$. For suppose Y is a directed subset of $X \cap D_k$. Then by the observation above, Y must in fact be a subset of $\{f_1, \dots, f_{k-2}\}$, and if $g \in \sqcup Y$, then $g \equiv_I f_j$, where j is the largest subscript that appears in Y .

We show that X^{dl} is not limit closed, from which it follows that $X^{dl} \notin D^*$. Let $g_k = [seq_1(a) \rightarrow seq_1(\perp), \dots, seq_k(a) \rightarrow seq_k(\perp)]$. Clearly $g_k \leq_I f_k$, so $g_k \in Fin_2 \cap X^{dl}$. Let $g = [seq_1(a) \rightarrow seq_1(\perp), \dots, seq_k(a) \rightarrow seq_k(\perp), \dots]$. Thus, $g \in D_2$ and $g \in \sqcup g_k$. Clearly $g \in X^{dl}$ but $g \notin X^{dl}$. Thus X^{dl} is not limit closed.

Example 11.7. We give an example where $X, Y \in D^*$ but $X::Y \notin D^*$. Take seq_i as in the previous example, and let $f_i = [seq_1(a) \rightarrow seq_1(a), \dots, seq_i(a) \rightarrow seq_i(a)]$. (Thus, f_i is essentially the identity function on items of the form $seq_j(a)$ for $j \leq i$.) Note that $f_i \in Fin_2$. Let $g_i = [seq_1(b) \rightarrow f_1, \dots, seq_i(b) \rightarrow f_i]$; clearly $g_i \in Fin_3$. Let $g = [seq_1(b) \rightarrow f_1, \dots, seq_i(b) \rightarrow f_i, \dots]$. Thus, $g \in D_3$. Let $X = \{g\}^d$ and $Y = \{seq_i(b) \mid i \in \mathbb{N}\}^d$. We leave it to the reader to check that $X, Y \in D^*$ and that $X::Y = \{f_i \mid i \in \mathbb{N}\}^d$. However $X:Y$ is not limit closed since $\{f_i \mid i \in \mathbb{N}\}^d$ has a least upper bound (namely, $[seq_1(a) \rightarrow seq_1(a), seq_2(a) \rightarrow seq_2(a), \dots]$) which is not in the set.

12. Appendix D: D^* Satisfies the Domain Equation

We want to prove Proposition 2.10. Our first step is to prove Lemma 2.9. We want to show that $X^{dl} : Y^{dl} = (X :: Y)^{dl}$. (Recall from Appendix C that $X :: Y = \{x : y \mid x \in X, y \in Y\}$.) This follows from the following chain of inequalities:

$$\begin{aligned}
 & X^{dl} : Y^{dl} \\
 &= (X^{dl} :: Y^{dl})^{dl} \\
 &\subseteq (X^d :: Y^d)^{dl} \quad (\text{since } : \text{ is continuous in each argument}) \\
 &\subseteq (X :: Y)^{dldl} \\
 &= (X :: Y)^{dl} \quad (\text{since } X :: Y \text{ is downward closed and limit closed by Proposition 2.6}) \\
 &\subseteq (X^{dl} :: Y^{dl})^{dl}
 \end{aligned}$$

Thus, equality holds throughout and the result follows. $\square$

We now turn our attention to Proposition 2.10. To prove part 1, we must show that $:$ (viewed as a function on $D^* \times D^*$) is continuous in each of its arguments. Suppose $\mathcal{Y}$ is a directed subset of D^* and $X \in D^*$. We have already shown that $\sqcup \mathcal{Y} = (\cup_{Y \in \mathcal{Y}} Y)^{dl}$. Thus, using Lemma 2.9 and the fact that $X = X^{dl}$, we get $X : (\sqcup \mathcal{Y}) = X : (\cup_{Y \in \mathcal{Y}} Y)^{dl} = (X :: (\cup_{Y \in \mathcal{Y}} Y))^{dl} = (\cup_{Y \in \mathcal{Y}} X :: Y)^{dl} = (\cup_{Y \in \mathcal{Y}} X : Y)^{dl} = \sqcup \{X : Y \mid Y \in \mathcal{Y}\}$. Thus $:$ is continuous in its second argument. The proof that $:$ is continuous in its first argument is completely analogous.

To prove part 2, we show that D^* is isomorphic to $\mathcal{A} + Seqs(D^*) + (D^* \rightarrow D^*)$. Following Scott (cf. [St77]), we will construct two monotonic functions, $\Phi : D^* \rightarrow (\mathcal{A} + Seqs(D^*) + (D^* \rightarrow D^*))$ and $\Psi : (\mathcal{A} + Seqs(D^*) + (D^* \rightarrow D^*)) \rightarrow D^*$ such that $\Phi \circ \Psi$ is the identity on $\mathcal{A} + Seqs(D^*) + (D^* \rightarrow D^*)$ and $\Psi \circ \Phi$ is the identity on D^* . That will suffice to prove part 2. The definitions of Ψ and Φ are obvious for $\perp$, the atomic items, and the sequence items. We define $\Psi(F) = \{f \in D \mid f \text{ is a function item and } \forall x \in D (f : x \in F(\{x\}^d))\} \cup \{\perp\}$. We must show that $\Psi(F) \in D^*$. If $f_1, f_2 \in \Psi(F) \cap D_k$, by Lemma 11.5 we can define $f_3(t)$ to be a least upper bound of $f_1 : t, f_2 : t$ with respect to $F(\{t\}^d) \cap D_{k-1}$ for all $t \in D_{k-1}$. By Lemma 11.2, $f_3 \in D_k$ and so $\Psi(F)$ is directed. $\Psi(F)$ is clearly downward closed. $\Psi(F)$ is limit closed since $:$ is continuous (on each D_k) in its first argument. Thus, $\Psi(F) \in D^*$. Clearly, Ψ is monotonic. If X is a function item in D^* , let $\Phi(X)(Y) = X : Y$ for all $Y \in D^*$. $\Phi(X) \in (D^* \rightarrow D^*)$ since $:$ is continuous (on D^*) in its second argument. Φ is clearly monotonic.

Let $F \in (D^* \rightarrow D^*)$. We show that $\Phi(\Psi(F)) = F$. Let X be an arbitrary element of D^* . By the definition of Ψ and the monotonicity of F , it is clear that $\Psi(F) : X \subseteq F(X)$. Thus, $\Psi(F) : X \subseteq F(X)$. Let $Y = X \cap Fin$; thus $X = \sqcup_{y \in Y} \{y\}^d$. Suppose $t \in F(X) \cap Fin$. By the continuity of F and the fact that t is finite, $t \in F(\{y\}^d)$ for some $y \in Y$. Hence, $[y \rightarrow t] \in \Psi(F)$. Thus, $t \in \Psi(F) : Y \subseteq \Psi(F) : X$. Therefore, $F(X) \cap Fin \subseteq \Psi(F) : X$. By Proposition 2.4 and the fact that $\Psi(F) : X$ is limit closed, $F(X) \subseteq \Psi(F) : X$. Therefore, we have that $\Psi(F) : X = F(X)$. By the defi-

dition of Φ , we have that $\Phi(\Psi(F))(X) = \Psi(F):X = F(X)$. As the choice of $X \in D^*$ was arbitrary, $\Phi(\Psi(F)) = F$.

We next show that $\Psi(\Phi(X)) = X$ for all $X \in D^*$. Let $X \in D^*$ be a function item and let $x \in X$. We clearly have that $\forall y \in D(x:y \in \Phi(X)(\{y\}^d))$. Thus, $x \in \Psi(\Phi(X))$ and so $X \subseteq \Psi(\Phi(X))$. If $x \in \Psi(\Phi(X))$, then $x:t \in \Phi(X)(\{t\}^d) = X:\{t\}^d = (X::\{t\}^d)^{dl}$. Thus, by Lemma 11.4, $x \in X^{dl} = X$ and so $\Psi(\Phi(X)) \subseteq X$. Therefore, $\Psi(\Phi(X)) = X$ for any arbitrary function item. This completes the proof of part 2.

To prove part 3, we show that $F = G$ provided $F, G \in D^*$ are function items and $F:X = G:X$ for all $X \in D^*$. By assumption and the definition of Φ in the proof of part 2, we have that $\Phi(F) = \Phi(G)$. Therefore, we get that $F = \Psi(\Phi(F)) = \Psi(\Phi(G)) = G$. $\square$

13. Appendix E: The FP-style rewrite rules

Note these rules apply only if the redex is well-formed.

$$\begin{aligned}
& \text{al}_m: \langle x, \langle y_1, \dots, y_n \rangle \rangle \rightarrow_a \langle x, y_1, \dots, y_n \rangle \\
& \text{al}_m: \langle x, \Omega(\langle y_1, \dots, y_n \rangle) \rangle \rightarrow_a \langle x, * (y_1, \dots, y_n) \rangle \\
& \text{ar}_m: \langle \langle y_1, \dots, y_n \rangle, x \rangle \rightarrow_a \langle y_1, \dots, y_n, x \rangle \\
& \text{ar}_m: \langle \Omega(\langle y_1, \dots, y_n \rangle), x \rangle \rightarrow_a \langle * (y_1, \dots, y_n), x \rangle \\
& \text{first}_m: \langle x_1, \dots, x_n \rangle \rightarrow_a x_1 \quad (n \geq 1) \\
& \text{first}_m: (\text{al}_m': \langle x, y \rangle) \rightarrow_a x \\
& \text{first}_m: (\text{ar}_m': \langle \text{al}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a x \\
& \text{first}_m: (\text{ar}_m': \langle \text{ar}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a \text{first}_m: (\text{ar}_m'': \langle x, y \rangle) \\
& \text{last}_m: \langle x_1, \dots, x_n \rangle \rightarrow_a x_n \quad (n \geq 1) \\
& \text{last}_m: (\text{ar}_m': \langle x, y \rangle) \rightarrow_a y \\
& \text{last}_m: (\text{al}_m': \langle x, \text{ar}_m'': \langle y, z \rangle \rangle) \rightarrow_a z \\
& \text{last}_m: (\text{al}_m': \langle x, \text{al}_m'': \langle y, z \rangle \rangle) \rightarrow_a \text{last}_m: (\text{al}_m'': \langle y, z \rangle) \\
& \text{tl}_m: \langle x_1, \dots, x_n \rangle \rightarrow_a \langle x_2, \dots, x_n \rangle \quad (n \geq 1, x_1 \neq * (y_1, \dots, y_k)) \\
& \text{tl}_m: \langle * (y_1, \dots, y_k), x_2, \dots, x_n \rangle \rightarrow_a \langle * (y_2, \dots, y_k, \rho(x_2)), x_3, \dots, x_n \rangle \quad (n \geq 3, k \geq 1) \\
& \text{tl}_m: \langle * (), x_2, \dots, x_n \rangle \rightarrow_a \langle * (), x_3, \dots, x_n \rangle, \quad (n \geq 3) \\
& \text{tl}_m: (\text{al}_m': \langle x, \text{al}_m'': \langle y, z \rangle \rangle) \rightarrow_a \text{al}_m'': \langle y, z \rangle \\
& \text{tl}_m: (\text{al}_m': \langle x, \text{ar}_m'': \langle y, z \rangle \rangle) \rightarrow_a \text{ar}_m'': \langle y, z \rangle \\
& \text{tl}_m: (\text{ar}_m': \langle \text{al}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a \text{ar}_m': \langle y, z \rangle \\
& \text{tl}_m: (\text{ar}_m': \langle \text{ar}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a \text{ar}_m': \langle \text{tl}_m: (\text{ar}_m'': \langle x, y \rangle), z \rangle \\
& \text{tr}_m: \langle x_1, \dots, x_n \rangle \rightarrow_a \langle x_1, \dots, x_{n-1} \rangle \quad (n \geq 1, x_n \neq * (y_1, \dots, y_k)) \\
& \text{tr}_m: \langle x_1, \dots, x_{n-1}, * (y_1, \dots, y_k) \rangle \rightarrow_a \langle x_1, \dots, x_{n-2}, * (\rho(x_{n-1}), y_1, \dots, y_{k-1}) \rangle \quad (n \geq 3, k \geq 1) \\
& \text{tr}_m: \langle x_1, \dots, x_{n-1}, * () \rangle \rightarrow_a \langle x_1, \dots, x_{n-2}, * () \rangle, \quad (n \geq 3) \\
& \text{tr}_m: (\text{ar}_m': \langle \text{ar}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a \text{ar}_m'': \langle x, y \rangle \\
& \text{tr}_m: (\text{ar}_m': \langle \text{al}_m'': \langle x, y \rangle, z \rangle) \rightarrow_a \text{al}_m'': \langle x, y \rangle \\
& \text{tr}_m: (\text{al}_m': \langle x, \text{ar}_m'': \langle y, z \rangle \rangle) \rightarrow_a \text{al}_m': \langle x, y \rangle \\
& \text{tr}_m: (\text{al}_m': \langle x, \text{al}_m'': \langle y, z \rangle \rangle) \rightarrow_a \text{al}_m': \langle x, \text{tr}_m: (\text{al}_m'': \langle y, z \rangle) \rangle \\
& \text{null}_m: \langle \rangle \rightarrow_a T \\
& \text{null}_m: a \rightarrow_a F \quad (\text{for any atom } a) \\
& \text{null}_m: \langle x_1, \dots, x_n \rangle \rightarrow_a F \quad (n \geq 1) \\
& \text{null}_m: (\text{al}_m': \langle x, y \rangle) \rightarrow_a F \\
& \text{null}_m: (\text{ar}_m': \langle x, y \rangle) \rightarrow_a F \\
& (\text{cons}_m: \langle f_1, \dots, f_n \rangle):x \rightarrow_a \langle f_1:x, \dots, f_n:x \rangle, \quad (f_i \neq * (g_1, \dots, g_k)) \\
& (\text{cons}_m: \langle f_1, \dots, * (g_1, \dots, g_k), \dots, f_n \rangle):x \rightarrow_a \langle f_1:x, \dots, * (g_1:\rho(x), \dots, g_k:\rho(x)), \dots, f_n:x \rangle, \quad n \geq 1 \\
& (\text{cons}_m: (\text{al}_m': \langle f, g \rangle)):x \rightarrow_a \text{al}_m': \langle f:x, (\text{cons}_m:g):x \rangle \\
& (\text{cons}_m: (\text{ar}_m': \langle f, g \rangle)):x \rightarrow_a \text{ar}_m': \langle (\text{cons}_m:f):x, g:x \rangle \\
& \text{cond}_m: \langle T, x, y \rangle \rightarrow_a x \\
& \text{cond}_m: \langle F, x, y \rangle \rightarrow_a y \\
& (\text{comp}_m: \langle f, g \rangle):x \rightarrow_a f:(g:x) \\
& \text{apply}_m: \langle f, x \rangle \rightarrow_a f:x \\
& (\text{K}_m:x):y \rightarrow_a P_{m-2}(x) \\
& \text{id}_m:x \rightarrow_a x
\end{aligned}$$

14. Appendix F: A proof that every FP' expression is essentially strongly normalizable

One might consider trying to show that every FP' expression is e.s.n. by induction on type (indeed, we imposed the notion of type exactly so that we could have something on which to induct!). However, the induction hypothesis that all expressions of type n are e.s.n. does not seem to be strong enough to enable this approach to work. An appropriate strengthening of the induction hypothesis is given by the notion of *stability* (cf [FLO83]). Before we can define stability, we need a more general notion of rewriting for FP' , which we denote $\rightarrow_b$. (We also use $\rightarrow_b$ in Section 7.)

Definition 14.1. The rewrite rules for $\rightarrow_b$ consist of all the rewrite rules for $\rightarrow_a$ together with:

$$x \rightarrow_b \Omega(\rho(x))$$

$$x \rightarrow_b P_m(x)$$

Note that $\rightarrow_b$ rewriting is in general unsound. But we do have

Lemma 14.2. *If $x' \rightarrow_b y'$, then $\mu'(y') \leq_I \mu'(x')$ and $\tau(y') \leq \tau(x')$.*

Proof. Straightforward. But again note that since $al_n: \langle x_1, x_2 \rangle \rightarrow_b al_n: P_m(\langle x_1, x_2 \rangle)$, the special cases of the definition of τ are essential here to show that type decreases. $\square$

Of course, we define *essentially strongly normalizable with respect to $\rightarrow_b$* in the obvious way: every $\rightarrow_b$ reduction sequence has only finitely many FP -style rules. We will show that every FP' expression is e.s.n. w.r.t. $\rightarrow_b$. Since $\rightarrow_b$ rewriting generalizes $\rightarrow_a$ rewriting, it will immediately follow that every FP' expression is e.s.n. w.r.t. $\rightarrow_a$.

Definition 14.3. An FP' expression y is a *component* of an FP' expression x iff there is a well-typed FP' expression z such that $x \rightarrow_b z$ and z has one the forms $al_i: \langle t, y \rangle$, $al_i: \langle y, t \rangle$, $ar_i: \langle y, t \rangle$, $ar_i: \langle t, y \rangle$, or $\langle \dots, y, \dots \rangle$.

Note that if y is a component of x then either $\tau(y) < \tau(x)$ or $\tau(y) = \tau(x)$ and $\mu'(y)$ is a sequence of shorter length than $\mu'(x)$. (This would not necessarily be true if we did not restrict our attention to *well-typed* expressions. For example, without this restriction, $\langle \langle a \rangle \rangle$ would be a component of $al_2: \langle a, \langle \langle a \rangle \rangle \rangle$, although its type is greater than that of $al_2: \langle a, \langle \langle a \rangle \rangle \rangle$.) This motivates the following definition.

Definition 14.4. The *weight* of an FP' expression x , written $w(x)$, is $\langle \tau(x), \ln(\mu'(x)) \rangle$, where $\ln(\langle x_1, \dots, x_k \rangle) = k$, $\ln(\langle x_1, \dots, x_i, *, x_{i+1}, \dots, x_k \rangle) = k$, and $\ln(x) = 0$ if x is not a sequence item.

We remark that since D has only finite sequences, we can do induction on weight, a fact which will be very useful in some of our proofs.

We are now ready to define stability. The precise definition is by induction on $w(x)$, where weights are well-ordered by the obvious lexicographic ordering.

Definition 14.5. An FP' expression x is *stable* iff it is e.s.n. w.r.t. $\rightarrow_b$ and

1. If $\mu'(x)$ is a sequence item, then every component of x is stable, and
2. If $\mu'(x)$ is a function item, then $x':y$ is stable for all x',y such that $x \rightarrow_b x'$, y is stable, and $x':y$ is well-typed.

Note that stability is well-defined, since by the remarks above, a component of x has lower weight than x , while if $x \rightarrow_b x'$ and $x':y$ is well-typed, then $\tau(x') \leq \tau(x)$, $\tau(y) < \tau(x')$ and $\tau(x':y) < \tau(x')$, hence the same inequalities hold with τ replaced by wt . It is also worth noting that the way we have defined stability guarantees

Lemma 14.6. *If x is stable and $x \rightarrow_b y$, then y is stable.*

Proof. Immediate from the definitions. $\square$

We remark that we defined stability as we did in order to make Lemma 14.6 easy to prove. We will make crucial use of this property in the sequel.

Proposition 14.7. *Every FP' expression is stable.*

Proof. The proof depends on the following five claims:

1. If x is an atom or of the form $\Omega(y)$, then x is stable.
2. If $x_1, \dots, x_n$ are stable, then $\langle x_1, \dots, x_n \rangle$ is stable and, if $n \geq 1$, so is $\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle$ for all FP expressions $z_1, \dots, z_k$ (see Lemma 14.9 below).
3. If x is stable, then so is $P_m(x)$ (see Lemma 14.10 below).
4. If y and z are stable, then so is $y:z$ (see Lemma 14.13 below).
5. Every primitive function symbol is stable (see Lemma 14.15 below).

Using the claims, we can show by an easy induction on structure that every FP' expression is stable. The first claim is trivial. In order to prove the second claim, we need a preliminary lemma. We write $Q(x)$ as an abbreviation for an expression of the form $P_{m_1}(P_{m_2} \dots (P_{m_k}(x)) \dots)$, $k \geq 0$.

Lemma 14.8. *If $\langle x_1, \dots, x_n \rangle \rightarrow_b y$, then y is of the form $Q(\langle y_1, \dots, y_n \rangle)$, where $x_i \rightarrow_b y_i$, $i = 1, \dots, n$.*

Proof. Straightforward. $\square$

Lemma 14.9. *If $x_1, \dots, x_n$ are stable, then $\langle x_1, \dots, x_n \rangle$ is stable and, if $n \geq 1$, so is $\langle x_1, \dots, x_i, *(z_1, \dots, z_k), x_{i+1}, \dots, x_n \rangle$ for all FP expressions $z_1, \dots, z_k$.*

Proof. We show $\langle x_1, \dots, x_n \rangle$ is stable; the other case is similar. Since x_i , $i = 1, \dots, n$, are all stable and hence e.s.n., it is easy to see using Lemma 14.8 that the sequence must be e.s.n. too. Moreover, if y is a component of $\langle x_1, \dots, x_n \rangle$, then it is easy to see that $x_i \rightarrow_b y$ for some i . Since x_i is stable, from Lemma 14.6 it follows that y is stable. $\square$

Lemma 14.10. *If x is stable, then so is $P_m(x)$.*

Proof. Immediate from Lemma 14.6, since $x \rightarrow_b P_m(x)$. (We remark that we defined $\rightarrow_b$ reduction precisely to facilitate this proof.) $\square$

Before we can prove that stability is closed under application, we again need a preliminary technical lemma.

Lemma 14.11. *If $x_0 \rightarrow_b x_1 \rightarrow_b x_2 \rightarrow_b \dots$, where $x_0 = y:z$, then at least one of the following cases must hold*

1. *for all k , x_k is of the form $Q(y_k:z_k)$, where $y \rightarrow_b^* y_k$, $z \rightarrow_b^* z_k$*
2. *for some k , $x_k = \Omega(\rho(x_{k-1}))$ and for all $j < k$, x_j is of the form $Q(y_j:z_j)$, where $y \rightarrow_b^* y_j$, $z \rightarrow_b^* z_j$.*
3. *for some k , x_k is a well-typed expression of the form $Q(y_k:z_k)$, where $y \rightarrow_b^* y_k$, $z \rightarrow_b^* z_k$.*

If $\mu'(x_0) = \perp$, then one of the first two cases holds.

Proof. If case 1 does not hold, consider the least k for which x_k is not of the form $Q(y_k:z_k)$, where $y \rightarrow_b^* y_k$, $z \rightarrow_b^* z_k$. Clearly we must have $k > 0$. By considering all the ways we could have $x_{k-1} \rightarrow_b x_k$, it is easy to see that either (a) x_k is of the form $\Omega(\rho(x_{k-1}))$, so we are in case 2, (b) x_{k-1} is of the form $Q(P_m(y_{k-1}):z_{k-1})$ where $y \rightarrow_b^* P_m(y_{k-1})$, $z \rightarrow_b^* z_{k-1}$, and x_k is of the form $Q(P_{m-1}(y_{k-1}):z_{k-1})$, or (c) x_{k-1} is of the form $Q(y_{k-1}:z_{k-1})$ and x_k is of the form $Q(x'_k)$, where $y_{k-1}:z_{k-1} \rightarrow_b x'_k$ is an instance of an FP-style rule. In both case (b) and (c), we must be in case 3, because the P -rule of case (b) and the FP-style rule of case (c) can only be applied if x_{k-1} is well-typed. $\square$

Lemma 14.12. *If y and z are both stable, $y:z \rightarrow_b^* x$, and x is well-typed, then x is stable.*

Proof. If $\mu'(y)$ is not a function item, then $\mu'(y:z) = \perp$. From Lemma 14.11 and the fact that both y and z are e.s.n. w.r.t. $\rightarrow_b$ (since they are stable), it is easy to see that in this case $y:z$ is e.s.n. w.r.t. $\rightarrow_b$ and hence stable. Since $y:z \rightarrow_b^* x$, by Lemma 14.6, x must also be stable.

If $\mu'(y)$ is a function item, suppose $x_0 \rightarrow_b \dots \rightarrow_b x_m$, where $x_0 = y:z$ and $x_m = x$. By Lemma 14.11 and the fact that x is well-typed, there must be some $k \leq m$ such that x_k is well-typed and either of the form $\Omega(\rho(x_k))$ or $Q(y_k:z_k)$, where $y \rightarrow_b^* y_k$, $z \rightarrow_b^* z_k$. In the first case x_k is clearly stable. In the second case, by Lemma 14.6, both y_k and z_k are stable. Since $y_k:z_k$ is well-typed, it follows from the definition of stability that $y_k:z_k$ is stable. By Lemma 14.10, any expression of the form $Q(y_k:z_k)$ is also stable, hence x_k is stable. Finally, by Lemma 14.6 it follows that x must be stable as well. $\square$

Lemma 14.13. *If y and z are stable, then so is $y:z$.*

Proof. We first show that $y:z$ is e.s.n. w.r.t. $\rightarrow_b$. Consider any infinite sequence of reductions $x_0 \rightarrow_b x_1 \rightarrow_b \dots$ with $x_0 = y:z$. We want to show that there can be at most finitely many FP-style rules in this sequence. If we are in case (1) or (2) of Lemma 14.11, this is clearly true, since both y and z are e.s.n. w.r.t. $\rightarrow_b$. If we are in case (3), then for some k , x_k is a well-typed FP' expression of the form $Q(y_k:z_k)$, where $y \rightarrow_b^* y_k$, $z \rightarrow_b^* z_k$. In this case, by Lemma 14.12 it follows that x_k is stable and hence e.s.n. w.r.t. $\rightarrow_b$. Thus it again follows that there are only finitely many FP-style rewrite rules in this sequence.

Now suppose that $\mu'(y:z)$ is a sequence item and y' is a component of $y:z$. Then there must be a well-typed FP' expression x such that $y:z \rightarrow_b x$ and x has one of the forms $al_i: \langle t, y' \rangle$, $al_i: \langle y', t \rangle$, $ar_i: \langle y', t \rangle$, $ar_i: \langle t, y' \rangle$, or $\langle \dots, y', \dots \rangle$. By Lemma 14.12, it follows that x must be stable. But y' is also a component of x and hence must also be stable. Thus it follows that $y:z$ is stable.

Finally, if $\mu'(y:z)$ is a function item, suppose that $y:z \rightarrow_b x$, z' is stable, and $x:z'$ is a well-typed expression. Since $x:z'$ is well-typed, then in particular x must be well-typed, and so by Lemma 14.12, x must be stable. Since $x:z' \rightarrow_b x:z'$, by another application of Lemma 14.12, it follows that $x:z'$ is stable. Thus, by the definition of stability, it follows that $y:z$ is stable. $\square$

Before we can prove that all primitive function symbols are stable, we need another preliminary technical lemma, somewhat analogous to Lemma 14.11.

Lemma 14.14. *Suppose $x_0 \rightarrow_b x_1 \rightarrow_b \dots$, where x_0 is of the form $Q_m(\dots Q_1(Q_0(f_n):y_1):\dots:y_m)$, $m \geq 1$, and f is a primitive function symbol. Then one of the following four situations holds:*

1. *for all k , x_k is of the form $Q'_m(\dots Q'_1(Q'_0(f_{n'}):y'_1):\dots:y'_m)$, where $n' \leq n$, $y_h \rightarrow_b y'_h$, $h = 1, \dots, m$,*
2. *for some k and i , x_k is of the form $Q'_m(\dots Q'_{i+1}(\Omega(\rho(f_{n'}:y'_1:\dots:y'_i)):y'_{i+1}):\dots:y'_m)$, where $y_h \rightarrow_b y'_h$, $h = 1, \dots, m$, $n' \leq n$,*
3. *$f \neq K, \text{comp}, \text{or cons}$, and for some k , x_k is of the form $Q'_m(\dots Q'_1(f_{n'}:y'_1):\dots:y'_m)$ and x_{k+1} is of the form $Q'_m(\dots (Q'_1(z):y'_2):\dots:y'_m)$, where $n' \leq n$, $y_h \rightarrow_b y'_h$, $h = 1, \dots, m$, and $f_{n'}:y'_1 \rightarrow_b z$ is an instance of an FP-style rewrite rule,*
4. *$f = K, \text{comp}, \text{or cons}$, $m \geq 2$, and for some k , x_k is of the form $Q'_m(\dots Q'_2(f_{n'}:y'_1:y'_2):\dots:y'_m)$ and x_{k+1} is of the form $Q'_m(\dots (Q'_2(z):y'_3):\dots:y'_m)$, where $n' \leq n$, $y_h \rightarrow_b y'_h$, $h = 1, \dots, m$, and $f_{n'}:y'_1:y'_2 \rightarrow_b z$ is an instance of an FP-style rewrite rule.*

Proof. The proof is similar in spirit to that of Lemma 14.11. If case 1 does not hold, consider the least k for which x_k is not of the appropriate form. Clearly we must have $k > 0$. By considering all the ways we could have $x_{k-1} \rightarrow_b x_k$, it is easy to see that one of cases 2, 3, or 4 must hold. $\square$

Lemma 14.15. *Every primitive function symbol is stable.*

Proof. We prove this by a case-by-case analysis of each primitive function symbol. Fortunately, the proof in most of the cases is similar.

We first show that al_n is stable for all n . Clearly no FP-style rewrite rules apply to al_n . Since $\mu'(al_n)$ is a function item, it suffices to show that $x:y$ is stable for all x, y such that $al_n \rightarrow_b x$, $\tau(y) < \tau(x)$, and y is stable. By Lemma 14.6, it actually suffices to show that $al_n:y$ is stable for all stable y such that $\tau(y) < n$.

Suppose we fix y such that $\tau(y) < n$ and y is stable. We first show $al_n:y$ is e.s.n. w.r.t. $\rightarrow_b$. Let $y_0 \rightarrow_b y_1 \rightarrow_b \dots$, where $y_0 = al_n:y$. We now apply Lemma 14.14. If we are in case (1) or case (2), then clearly there are only finitely many FP-style rules in this sequence since

y is e.s.n. If we are in case (3), then y_k must be of the form $Q(al_n': \langle z, \langle z_1, \dots, z_m \rangle \rangle)$ or $Q(al_n': \langle z, \Omega(\langle z_1, \dots, z_m \rangle) \rangle)$, where $y \rightarrow_b^* \langle z, \langle z_1, \dots, z_m \rangle \rangle$ (resp. $y \rightarrow_b^* \langle z, \Omega(\langle z_1, \dots, z_m \rangle) \rangle$) and y_{k+1} is of the form $\langle z, z_1, \dots, z_m \rangle$ (resp. $\langle z, *(z_1, \dots, z_m) \rangle$). In the former case, since y is stable, so are $z, z_1, \dots, z_m$ (since they are components of y or components of components of y) and hence, by Lemma 14.9, so is $\langle z, z_1, \dots, z_m \rangle$. Similarly we can show that $\langle z, *(z_1, \dots, z_m) \rangle$ is stable in the latter case. Thus y_{k+1} is stable and e.s.n. w.r.t. $\rightarrow_b$. So in this case too there can be only finitely many FP-style rewrite rules in the reduction sequence.

All that remains to show is that if $\mu'(al_n:y)$ is a sequence item (it clearly cannot be a function item, although it may be $\perp$), then all the components of $al_n:y$ are stable. By another application of Lemma 14.14, it is easy to see that every component of $al_n:y$ must also be a component of y or a component of a component of y . Since y is stable, all the components are stable. Thus $al_n:y$ is stable.

A similar proof shows that ar_n is stable for all n .

The proof that tl_n is stable for all n is very similar to that for al_n , but there are a few added subtleties which we point out here. It suffices to show that $tl_n:y$ is stable if y is stable and $\tau(y) < n$. We do this by induction on $wl(y)$. We first show that $tl_n:y$ is e.s.n. w.r.t. $\rightarrow_b$. So suppose that $y_0 \rightarrow_b y_1 \rightarrow_b \dots$, where $y_0 = tl_n:y$. The argument is straightforward if case (1) or case (2) of Lemma 14.14 applies, so assume that case (3) applies. As in the case of al_n , we show that y_{k+1} is stable. This is done by considering in turn all the possible tl rules that could apply to rewrite y_k to y_{k+1} . The only difficulty comes in the case that y_k is of the form $Q(tl_n': (ar_{n_1}: \langle ar_{n_2}: \langle z_1, z_2 \rangle, z_3 \rangle))$, where $n' \leq n$. In this case, y_{k+1} is of the form $Q(ar_{n_1}: \langle tl_{n'}: (ar_{n_2}: \langle z_1, z_2 \rangle), z_3 \rangle)$. Now z_1, z_2, z_3 must all be stable, since they are components of y or components of components of y , and y is stable. We have already shown that stability is closed under sequence formation and application, and that ar_m is stable for all m , hence in particular $ar_{n_2}: \langle z_1, z_2 \rangle$ is stable. Note that $wl(ar_{n_2}: \langle z_1, z_2 \rangle) < wl(y)$, so by the induction hypothesis we get that $tl_{n'}: (ar_{n_2}: \langle z_1, z_2 \rangle)$ is stable. Using Lemma 14.9, Lemma 14.10, and the fact that z_3 and ar_{n_2} are stable, it easily follows that y_{k+1} is stable, as desired. Thus there are only finitely many FP-style rewrite rules in this sequence.

It only remains to show that if $\mu'(tl_n:y)$ is a sequence item (it cannot be a function item), then all its components are stable. This follows by another application of Lemma 14.14. For example, if $tl_n:y \rightarrow_b^* \langle z_1, \dots, z_m \rangle$, then by the same arguments as used above, we can show that each z_i must be stable. Similar techniques work for all the other cases.

Of course, the proof that tr_n is stable for all n proceeds along the same lines as that of tl_n .

Finally, we show $cond_n$ is stable for all n . Indeed, we will show that if $y_1, y_2, \dots, y_m$ are all stable, then $x = Q_m(\dots Q_1(Q_0(cond_n): y_1) \dots y_m)$ is stable. The proof proceeds by induction on $\tau(x)$. We only sketch the details here, since the ideas are so similar to the previous proofs we have discussed.

The proof that x is e.s.n w.r.t. $\rightarrow_b$ uses Lemma 14.14 in the same way as all the previous proofs. Similarly, if $\mu'(x)$ is a sequence item, we can show that all components are stable. Finally, if $\mu'(x)$ is a function item, we must show that if $x \rightarrow_b x'$, y is stable, and $x':y$ is well-typed, then $x':y$ is stable. Using Lemma 14.14 in the same way as in previous arguments, we can either show that x' is stable, so that from Lemma 14.13 $x':y$ is stable, or show that $x':y$ is of the same form as x except of lower type, so by the induction hypothesis it is stable.

The proof that f_n is stable for all other primitive function symbols f proceeds along the same lines as that for cond_n ; in the case of first_n , last_n , and cons_n , we need to proceed by induction on $\tau(x)$ with a subinduction on $w(y_1)$, much like that in the proof for tl_n . We remark that we need the w function exactly for those primitive functions f that have rewrite rules in which $f:x$ (or $f:x:y$ in the case of cons) rewrites to an expression that also contains f . $\square$

We can now use a straightforward induction on structure to show that all FP' expressions are stable. Since stability implies e.s.n. w.r.t. $\rightarrow_b$, which in turn implies e.s.n. (w.r.t. $\rightarrow_a$), this completes the proof of Theorem 5.11. $\square$

15. Appendix G: Modified Leftmost is O-safe

In this appendix we show that the modified leftmost strategy is O-safe. We want to show that if $\mu(x)$ is an observable item, then for some n , $ML(x, n) = \mu(x)^e$. By Corollary 4.4, it suffices to show that ML reduces x to normal form. Since $\mu(x)$ is observable, we can find a well-formed FP' expression x' such that $\rho(x'_0) = x$ and $\mu'(x'_0) = \mu(x)$. Consider a tracking sequence (see Definition 6.7) $x'_0, x'_1, x'_2, \dots$ for x under the ML rewrite strategy. As noted, it suffices to show that $\rho(x'_n)$ is in normal form for some n . For this, it suffices to show that $x'_k \rightarrow_a^* x'_{k+1}$ uses an FP-style rule (as opposed to using a bottom rule) whenever $\rho(x'_k)$ is not in normal form. For then we have that either $\rho(x'_n)$ is in normal form for some n or else the tracking sequence has an infinite number of FP-style rules, contradicting Theorem 5.11. Since ML is history independent, it suffices to show the result for the first step, namely that $x'_0 \rightarrow_a^* x'_1$ uses an FP-style rule whenever $\rho(x'_0)$ is not in normal form and $ML(\rho(x'_0), 1) = \rho(x'_1)$, provided $\mu'(x'_0)$ is observable. We will find it more convenient to prove the contrapositive, namely that if $x'_0 \rightarrow_a x'_1$ using a bottom rule, then $\mu'(x'_0)$ is not observable. Before checking this, we need a bit of notation. Let x^{ml} be the FP expression obtained from the FP expression x by replacing the ML redex (if one exists) by $a:b$ where a and b are atoms. (There is nothing special about the choice of $a:b$; any expression whose meaning is $\perp$ would do.)

Lemma 15.1. *If $x'_0 \rightarrow_a x'_1$ by the use of a bottom rule and $\rho(x'_1) = ML(\rho(x'_0), 1)$, then $\mu'(x'_0) \leq_I \mu((\rho(x'_0))^{ml})$.*

Proof. This is easily proved by induction on the structure of x'_0 . For example, assume that x'_0 has the form $y':z'$. Then $y':z'$ is not the redex for the $\rightarrow_a$ rule used to rewrite x'_0 to x'_1 since we are assuming that a bottom rule was used. Therefore, the $\rightarrow_a$ redex must be entirely

contained inside of either y' or z' . Say, for example, that the redex is entirely contained in y' . Then we get $\mu'(x'_0) = \mu'(y':z') = \mu'(y') : \mu'(z') \leq_I \mu((\rho(y'))^{ml}) : \mu(\rho(z')) = \mu((\rho(y'))^{ml} : \rho(z')) = \mu((\rho(y') : \rho(z'))^{ml}) = \mu((\rho(y':z'))^{ml}) = \mu((\rho(x'_0))^{ml})$. $\square$

We next want to show that if x is an FP expression not in normal form, then $\mu(x^{ml})$ is not observable. Once we show this, it follows from Lemma 15.1 that no bottom rule could have been used in a tracking sequence for (ML, x) if $\mu(x)$ is observable. Coupled with the previous observation, this says that for some n , it must be the case that $ML(x, n)$ is in normal form. Thus the fact that ML is O -safe follows from part 3 of the following lemma (parts 1 and 2 are necessary only as induction hypotheses).

Lemma 15.2. *If x is not in normal form, then*

1. *If $\mu(x^{ml})$ is a sequence item, then x has one of the forms $al: \langle x_1, x_2 \rangle$, $ar: \langle x_1, x_2 \rangle$, or $\langle x_1, \dots, x_k \rangle$; if $\mu(x^{ml})$ is of definite length, then in fact x is of the form $\langle x_1, \dots, x_k \rangle$.*
2. *If $\mu(x^{ml})$ is a function item, then x has the form $f:y_1$ where f is `comp`, `cons`, or `K`.*
3. *$\mu(x^{ml})$ is not observable.*

Proof. We proceed by induction on the structure of x . If x is either an atom or a primitive function symbol, then it is in normal form, and hence the result is vacuously true. If x is of the form $\langle x_1, \dots, x_n \rangle$, then part 1 is trivial, part 2 is vacuous, and part 3 follows immediately from the induction hypothesis.

Finally, we consider the case that x is of the form $y:z$. First note that if $\mu(x^{ml}) = \perp$, then parts 1 and 2 are vacuous and part 3 is trivial. If x is a redex or $\mu(y^{ml})$ is not a function item, then $\mu(x^{ml}) = \perp$, so without loss of generality we can assume that neither of these is the case. Consider the case that y is not in normal form. By definition, $ML(y:z, 1) = ML(y, 1):z$ and $x^{ml} = y^{ml}:z$. Since we assume that $\mu(y^{ml})$ is a function item, by the induction hypothesis y must have the form $f:w$ where f is `comp`, `cons`, or `K`, and $y^{ml} = f:w^{ml}$. If $f = K$, then x is a redex, contradicting our assumption. Similarly, if $f = \text{comp}$, $\mu(w^{ml})$ must have length 2 (otherwise $\mu(x^{ml}) = \perp$), which means that $w = \langle w_1, w_2 \rangle$ by the induction hypothesis. In this case, once again, x is itself a redex. A similar argument holds if $f = \text{cons}$. So y must be in normal form. By Theorem 4.3, it follows that y must in fact be a primitive function symbol f or of the form $f:w$, where f is `comp`, `cons`, or `K`. A similar argument to that above shows that it cannot be in the latter form, so we are down to considering the case where y is a primitive function symbol.

First suppose y is `al`. By the induction hypothesis z must be of the form $\langle z_1, z_2 \rangle$ (otherwise $\mu(x^{ml}) = \perp$). All three parts easily follow if we show that $\mu(x^{ml})$ does not have definite length. Notice that z_2 does not have the form $\langle \dots \rangle$ (otherwise x is a redex). If z_2 is in normal form, then $\mu(z_2)$ cannot have definite length by Theorem 4.3 and thus $\mu(x^{ml}) = \mu(al: \langle z_1^{ml}, z_2 \rangle)$ does not have definite length. If z_2 is not in normal form, then

$\mu(z_2^{ml})$ cannot have definite length by the induction hypothesis and thus $\mu(x^{ml}) = \mu(al: \langle z_1, z_2^{ml} \rangle)$ cannot have definite length either.

A similar proof works in the case that $y = ar$; we omit details here (although it is interesting to note that we do not need a special clause in the definition of ML for ar , since in this case we have to work on z_1 rather than z_2).

To complete the proof, it suffices to show that if y is not al or ar , then it is one of K , $cons$, or $comp$. The idea is to show that if y is some other primitive function, then either x is a redex or $\mu(x^{ml}) = \perp$, both of which are contradictions. We check only the cases $cond$ and tl and leave the other cases to the reader.

In the case of $y = cond$, $\mu(z^{ml})$ must be a sequence of length 3 whose first component is T or F , or else $\mu'(x^{ml}) = \perp$. So, by the induction hypothesis, we may assume that $z = \langle z_1, z_2, z_3 \rangle$. If z_1 is in normal form, then by Theorem 4.3, z_1 must either be T or F (in which case x is itself a redex). If z_1 is not in normal form, then $\mu(z_1^{ml})$ is not observable (by the induction hypothesis), which implies that $\mu(x^{ml}) = \perp$. So $y \neq cond$.

Now suppose that $y = tl$. We may assume that $\mu(z^{ml})$ is a sequence item (otherwise $\mu(x^{ml}) = \perp$). $\mu(z^{ml})$ does not have definite length (otherwise x would be a redex by the induction hypothesis). Thus, by the induction hypothesis, z must have one of the forms $al: \langle z_1, z_2 \rangle$ or $ar: \langle z_1, z_2 \rangle$. Since they are similar, we consider only the al case. z_2 cannot have the form $\langle \dots \rangle$ (otherwise $\mu(z^{ml})$ has definite length). z_2 cannot have one of the forms $al: \langle z_3, z_4 \rangle$ or $ar: \langle z_3, z_4 \rangle$ (otherwise x is a redex). If z_2 is in normal form, then by Theorem 4.3 $\mu(z_2)$ is not a sequence item and hence $\mu(x) = \perp$; thus, $\mu(x^{ml}) = \perp$ which is a contradiction. If z_2 is not in normal form, then by the induction hypothesis, $\mu(z_2^{ml})$ is not a sequence item and hence $\mu(x^{ml}) = \perp$ which is also a contradiction. Thus, $y \neq tl$.

Similar proofs work in all other cases. $\square$

16. Appendix H: Characterizing S-safety

Recall that we are trying to show that (1) R is S-safe for x iff (2) every tracking sequence for (R, x) reaches e.n.f. We first show that (1) implies (2).

Definition 16.1. A rewrite strategy R is *steady* with respect to a well-formed FP' expression x' iff R is S-safe for $\rho(x')$ and there exists a tracking sequence for $(R, \rho(x'))$ starting with x' in which no FP -style rewrite rules are used.

Lemma 16.2. If x' is a well-formed FP' expression and R is a rewrite strategy that is steady with respect to x' , then x' is in essential normal form.

Proof. Suppose x' is well-formed and R is steady w.r.t x' . We first show that if $\mu'(x') \neq \perp$, then $x = \rho(x')$ must be transparent. For suppose not. Then, since R is S-safe for x , there must be some $n > 0$ such that $R(x, n)$ is transparent and $R(x, n')$ is not transparent for $n' < n$. Let $x'_0, x'_1, \dots$ be any tracking sequence for (R, x) with $x' = x'_0$. We claim there must be an FP -style rewrite rule applied in reducing $x'_{n-1} \rightarrow_a^* x'_n$. Since $\rho(x'_{n-1}) \neq \rho(x'_n)$, either a bottom

rule or an FP-style rule must be applied. If it is a bottom rule, then we leave it to the reader to check that we must have $\mu'(x_{n-1}) = \perp$. But this contradicts the assumption that $\mu'(x') \neq \perp$. Thus every tracking sequence for (R, x) starting with x' has an occurrence of at least one FP-style rule. But this contradicts the assumption that R is steady for x' .

The lemma now follows by induction on the structure of x' , using the observation above. If x' is an atom or a primitive function or of the form $\Omega(x)$, then it is clearly in e.n.f. If x' is of the form $\langle y'_1, \dots, y'_m \rangle$ and no y'_i has the form $\ast(\dots)$, let R'_i be the strategy defined on $\rho(y'_i)$, $i = 1, \dots, m$, by taking $R'_i(\rho(y'_i), n)$ to be the i^{th} component of $R(x, n)$. It is easy to see that R'_i is steady for y'_i , so by the inductive hypothesis, y'_i is in e.n.f. Hence $x' = \langle y'_1, \dots, y'_m \rangle$ is in e.n.f. as well. A similar proof works if one of the y'_i has the form $\ast(\dots)$. If x' is of the form $y':z'$ or $P_m(y':z')$ (the only remaining cases, since x' is well formed), then we proceed by cases on the form of $\mu'(x')$. Since x' does not have the form $\Omega(x)$ and x' is well-formed, we have that $\mu'(x') \neq \perp$. Thus, by the observation above, we know that $\rho(x')$ must be transparent. This means that $\rho(x)$ must be of the form $\text{al}:\langle z, y \rangle$ or $\text{ar}:\langle y, z \rangle$, where $\mu(y)$ is not of definite length, or of the form $f:y$, where f is one of comp , cons , or K . We consider only the case that $\rho(x')$ is of the form $\text{comp}:y$ since all the other cases are similar. In this case x' is of the form $\text{comp}_k:y'$ or $P_m(\text{comp}_k:y')$. Let R' be the rewrite strategy defined on y via $R'(y, n) = z$ if $R(x, n) = \text{comp}:z$. It is easy to see that R' is well defined and that R' is steady for y' . By the induction hypothesis, y' is in e.n.f., hence so is x' . $\square$

Now, to prove that (1) implies (2), suppose that R is S-safe for x , and let $x'_0, x'_1, \dots$ be a tracking sequence for (R, x) . Since FP' is essentially strongly normalizable (Theorem 5.11), there exists some n such that no FP-style reductions are performed in reducing $x'_k \rightarrow_a^* x'_{k+1}$ for $k \geq n$. Define the rewrite strategy R' on $x_n = R(x, n)$ by taking $R'(x_n, n') = R(x, n + n')$. Since R is S-safe for x , it is easy to see that R' is S-safe for x_n and moreover that $x'_n, x'_{n+1}, \dots$ is a tracking sequence for (R, x_n) in which no FP-style rewrite rules are used. Thus R' is steady for x'_n . By Lemma 16.2, x'_n is in e.n.f. Since the tracking sequence $x'_0, x'_1, \dots$ was chosen arbitrarily, it follows that every tracking sequence for (R, x) reaches e.n.f.

To prove that (2) implies (1), we again need some preliminary definitions and lemmas. As motivation, we will very briefly sketch part of the proof. Assume we are trying to show that R rewrites x to a 1-transparent expression. We pick an FP' expression x' with $\rho(x') = x$ such that $\mu'(x') \neq \perp$. By (2), we know that the tracking sequence starting with x' must reach e.s.n. When it does, the corresponding FP expression will be transparent by Proposition 5.13. Having gotten a 1-transparent expression, we then want to rewrite it to a 2-transparent expression. For example, suppose that $x \rightarrow^* \langle x_1, \dots, x_n \rangle$. We would like to continue inductively with each of the x_i . Unfortunately, in the tracking sequence the FP' expression that corresponds to x_1 might be of the form $\Omega(x_1)$, which means that we cannot apply the induction hypothesis. The solution is to choose the subscripts for the FP' expression x'_0 that starts the tracking sequence to be so big that even after rewriting x_0 to a transparent

expression x' , the subscripts are still big enough so that no “relevant” subexpression of x' has meaning $\perp$. To do this, we need an upper bound on how much a subscript can be reduced during the rewriting process. We do this by finding a bound on the number of rewriting steps and a bound on the amount that each rewrite rule can lower subscripts and then multiplying those two bounds.

Lemma 16.3. *If x' is well-formed and $x' \rightarrow_a y'$, then $|y'| \leq 5|\rho(y')|$.*

Proof. Let z' be an arbitrary FP' expression. We define $Plen(z')$ to be the maximum i such that z' has a subexpression of the form $P_{n_1}(\dots P_{n_i}(z'_0))$. It is easy to prove by induction on the structure of $\rho(z')$ that $|z'| \leq (2 + Plen(z')) \times |\rho(z')|$. Note that we need the constant 2 to handle the case where $z' = \Omega(z)$, so that we have $|z'| = 1 + |z| \leq 2|z|$. Thus to complete the proof it suffices to show that $Plen(y') \leq 3$.

Since x' is well-formed, the only possible rules that could apply are bottom rules and FP style rules. If $x' \rightarrow_a y'$ by a bottom rule, then y' is well-formed and hence $Plen(y') \leq 1$. If $x' \rightarrow_a y'$ by an FP -style rule, then $Plen(x') \leq 1$ since x' is well-formed. There are two ways that $Plen$ can grow, one by the FP -style rule introducing a P and the other by the FP -style rule taking away some structure and thereby allowing two P 's to come together. The first occurs only in the K rule $(K_m;x):y \rightarrow P_{m-2}(x)$. The second occurs only when the redex involves a K or id ; a case analysis shows that in all such cases we have $Plen(y') \leq 3$. $\square$

Definition 16.4. Given an FP' expression x' , let $\min(x')$ (resp. $\max(x')$) be the least (resp. greatest) subscript that appears in x' . If no subscripts occur in x' , then we take $\min(x') = \infty$ and $\max(x') = 0$. For example, if $x' = P_2(\text{comp}_7; \langle K_5, a \rangle)$, then $\min(x') = 2$ and $\max(x') = 7$.

Lemma 16.5. *If x' is an FP' expression such that $\mu'(x')$ is a function item, then $\min(x') - C(x') \leq \tau(x') \leq \max(x')$, where $C(x')$ is the number of colons ($:$) that appear in $\rho(x')$.*

Proof. By a straightforward induction on the structure of x' . $\square$

Remark: This lemma does not necessarily hold for expressions whose meanings are not function items; for example, $\tau(\langle a \rangle) = 1$ while $\max(\langle a \rangle) = 0$.

Lemma 16.6. *If $x'_1 \rightarrow_a^* x'_2$ using only P -rules and lowering rules, then $\min(x'_2) \geq \min(x'_1) - 8^{|x'_1|}$.*

Proof. By Proposition 5.7, we know at most $4^{|x'_1|}$ P -rules and lowering rules can be applied in rewriting x'_1 to x'_2 . Each application of a P -rule can reduce the minimal subscript by at most 1, while the rule $x' \rightarrow_a \Omega(\rho(x))$ cannot lower the minimal subscript. By Lemma 16.5, for every application of the lowering rule $x':y' \rightarrow_a x':P_{m-1}(y')$ we must have $m = \tau(x') \geq \min(x') - C(x')$, so it is easy to see that an application of this rule can lower the minimal subscript by at most $C(x') + 1 \leq C(x':y')$. Hence, $\min(x'_2) \geq \min(x'_1) - (1 + C(x'_1))4^{|x'_1|} \geq \min(x'_1) - (2^{|x'_1|} 4^{|x'_1|}) = \min(x'_1) - 8^{|x'_1|}$ as desired. $\square$

In general, we want to prevent an FP' expression from assuming the form $\Omega(x)$ whenever possible, since our induction hypothesis will not hold in this case. It is impossible to prevent

$x' \rightarrow \Omega(\rho(x))$ if $\mu(\rho(x)) = \perp$, but fortunately, this case does not cause us any problems. This motivates the following definition.

Definition 16.7. An FP' expression x' is *comfortably tight* iff it is star-free, and for all subexpressions of x' of the form $\Omega(z)$, $\mu(z) = \perp$. An FP' expression x' is *tight* iff it is comfortably tight and for all subexpressions y' of x' , if $\mu'(y') = \perp$, then $\mu(\rho(y')) = \perp$.

Note that the lowering rule $x' \rightarrow_a \Omega(\rho(x'))$ if $\mu'(x') = \perp$ guarantees that a well-formed expression is comfortably tight iff it is tight. Moreover, in a rewriting sequence such as $P_1(<< >>) \rightarrow_a <P_0(< >)> \rightarrow_a <\Omega(< >)>$, the first expression is tight, the second expression is comfortably tight but not tight, and the third expression is not comfortably tight. Thus, neither tight expressions nor comfortably tight expressions are closed under rewriting. However, we do have

Lemma 16.8. *If $x' \rightarrow_a y'$ and x' is tight, then y' is comfortably tight.*

Proof. Since x' is star-free, the only possible way for a $*$ to appear in y' is by applying the FP -style rule $al_k: <z', \Omega(<z_1, \dots, z_n>) > \rightarrow <z', * (z_1, \dots, z_n) >$ (or the corresponding rule for ar) to x' . But this rule could not have applied since x' is comfortably tight and thus does not have subexpressions of the form $\Omega(<\dots>)$. Thus, y' is star-free. And if y' has a subexpression of the form $\Omega(z)$ then it must be because (1) $\Omega(z)$ is a subexpression of x' , (2) a bottom rule of the form $\Omega(z_0) \rightarrow_a \Omega(z)$ was applied, or (3) a lowering rule of the form $z' \rightarrow_a \Omega(z)$ was applied. In case (1), we must have $\mu(z) = \perp$ since x' is tight, in case (2) we must have $\mu(z_0) = \perp$ (again, since x' is tight) so $\mu(z) = \perp$, and in case (3) we must have $\mu'(z') = \perp$, so since x' is tight we must also have $\mu(z) = \perp$. Thus y' is comfortably tight. $\square$

Proposition 16.9. *For every FP expression x and every finite $d \leq_I \mu(x)$ there exists an n such that if x' is a comfortably tight FP' expression, $\rho(x') = x$, and $\min(x') \geq n$, then $d \leq_I \mu'(x')$.*

Proof. This is easily proved by induction on the structure of x in a manner similar to the proof of part 2 of Proposition 5.3. $\square$

Corollary 16.10. *For every FP expression x there exists an n such that if $\rho(x') = x$, x' is comfortably tight, and $\min(x') \geq n$, then x' is tight.*

Proof. By induction on the structure of x , using Proposition 16.9. $\square$

Lemma 16.11. *For every FP expression x and every n , there exists a tight, well-formed FP' expression x' such that $\rho(x') = x$ and $\min(x') \geq n$.*

Proof. Let n_1 be the n that is guaranteed to exist for x by Corollary 16.10. Let $m = \max(n_1, n) + 8^{|x|}$. Let x'_0 be x with every primitive function symbol labelled by m . Thus, $|x'_0| = |x|$. x'_0 is comfortably tight by construction and so tight by Corollary 16.10. We show that if $x'_0 \rightarrow_a^* x''$ using only P -rules and lowering rules, then x'' is tight and $\min(x'') \geq n$. We proceed by induction on k , the number of P -rules and lowering rules used. For the general case, suppose that $x'_0 \rightarrow_a^* x''$ using $k+1$ rules. Thus there exists y such that $x'_0 \rightarrow_a^* y$

using k rules, and $y \rightarrow_a x''$. By induction hypothesis, y is tight. By Lemma 16.8, x'' is comfortably tight. From Lemma 16.6, it follows that $\min(x'') \geq \max(n_1, n)$. Thus by Corollary 16.10 again, x'' is tight. By Proposition 5.7, there is a well-formed expression z such that $x_0 \rightarrow_a^* z$ using only P -rules and lowering rules. By the argument we have just made, z satisfies the requirements of the lemma. $\square$

Lemma 16.12. *For every FP reduction sequence $x_0 \rightarrow \dots \rightarrow x_k$ and every $M \geq 0$, there exists N such that if $x'_0 \rightarrow_a^* \dots \rightarrow_a^* x'_k$ is a tracking sequence for the given FP reduction sequence, x'_0 is tight, and $\min(x'_0) \geq N$, then x'_i is tight and $\min(x'_i) \geq M$ for $i = 0, \dots, k$.*

Proof. This is easily proved by induction on k (the length of the reduction sequence) once we have established the base case. So assume $k = 1$ and that $x_0 \rightarrow x_1$. Let n_1 be the n that is guaranteed to exist by Corollary 16.10 for x_1 . Take $N = \max(M, n_1) + 2 + 8^{5|x_1|}$. To see that this works, let $x'_0 \rightarrow_a^* x'_1$ be a tracking sequence with $\min(x'_0) \geq N$ and x'_0 tight. There must exist some x'' such that $x'_0 \rightarrow_a x''$ using an FP-style rule or a bottom rule, and $x'' \rightarrow_a^* x'_1$ using only P -rules and lowering rules. By Lemma 16.8, x'' is comfortably tight. By Lemma 16.3, $|x''| \leq 5|\rho(x'')| = 5|x_1|$. Moreover, $\min(x'') \geq \max(M, n_1) + 8^{5|x_1|}$ since the only FP-style or bottom rule that can lower a subscript is the K rule and that can lower a subscript by at most 2. Thus, by Corollary 16.10, x'' is tight. By an argument analogous to that of the previous lemma, we can now show that for all t , if $x'' \rightarrow_a^* t$ using only P -rules and lowering rules, then t is tight. In particular, it follows that x'_1 is tight. $\square$

Lemma 16.13. *There exists a function BIG such that for all positive integers n , rewrite strategies R , and FP expressions x , $BIG(n, R, x)$ is a positive integer with the following property: for every FP' expression x' such that (a) x' is well-formed and tight, (b) $\rho(x') = x$, (c) $\min(x') \geq BIG(n, R, x)$, and (d) some tracking sequence for (R, x) starting with x' reaches e.n.f., there exists m such that $R(x, m)$ is n -transparent.*

Proof. First we define $BIG(n, R, x) = 0$ if $\mu(x) = \perp$. The result is trivially true in this case. For the rest of the proof, we assume that $\mu(x) \neq \perp$. We define BIG by induction on n for all R and x . For the base case, using Proposition 16.9, we take $BIG(1, R, x)$ sufficiently large so that if $\min(x') \geq BIG(1, R, x)$, then $\mu'(x') \neq \perp$, and if $\mu(x)$ is a sequence of definite length, then so is $\mu'(x')$. Suppose the m^{th} term in some tracking sequence for (R, x) starting with x' is in e.n.f. If this m^{th} term is x'_m , then by Proposition 5.13, it follows that $\rho(x'_m) = R(x, m)$ is transparent.

Inductively, suppose we have defined $BIG(i, R, x)$ for all R and x and $i \leq n$. We show how to define $BIG(n+1, R, x)$. Given R and x , there are two cases to consider: either for all m , $R(x, m)$ is not transparent, or for some m , $R(x, m)$ is transparent. In the former case, we take $BIG(n+1, R, x) = BIG(1, R, x)$. Given a tight, well-formed x' with $\min(x') \geq BIG(1, R, x)$ and $\rho(x') = x$, then no tracking sequence starting with x' ever reaches e.n.f (for otherwise, by the definition of $BIG(1, R, x)$, there would be some m such that $R(x, m)$ is transparent,

contradicting our assumption), so that the result is vacuously true. In the latter case, let m be such that $R(x, m)$ is transparent and $R(x, m')$ is not transparent for $m' < m$. $R(x, m)$ is either an atom, a primitive function symbol, or of the form $\langle y_1, \dots, y_k \rangle$, $al: \langle y, z \rangle$, $ar: \langle y, z \rangle$, or $f:y$, for $f = K, cons$, or $comp$. If $R(x, m)$ is either an atom or a primitive function symbol, then we can trivially choose $BIG(n+1, R, x) = 0$. Since the remaining cases are similar, we consider only the case that $R(x, m)$ is of the form $f:y$ here, leaving the other cases to the reader.

Suppose $R(x, m)$ is of the form $f:y$, where f is $K, comp$, or $cons$. Let R' be a rewrite strategy such that $R'(y, k) = z$ if $R(x, m+k) = f:z$. (Note that for all k , $R(x, m+k)$ is guaranteed to be of the form $f:z$, so R' is indeed well defined.) By Lemma 16.12, we can find N such that if x'_0 is tight, $\min(x'_0) \geq N$, and $x'_0 \rightarrow_a^* \dots \rightarrow_a^* x'_m$ is a tracking sequence for $R(x, 0) \rightarrow^* \dots \rightarrow^* R(x, m)$, then x'_m is tight and $\min(x'_m) \geq BIG(n, R', y)$. Take $BIG(n+1, R, x) = \max(N, BIG(1, R, x))$. To see that this choice works, suppose x' is such that $\rho(x') = x$, x' is tight and well-formed, $\min(x') \geq BIG(n+1, R, x)$, and some tracking sequence for x reaches e.n.f. The sequence must reach e.n.f. at some term $m' \geq m$, for if $m' < m$, the argument in the first paragraph of this proof shows that $R(x, m')$ is transparent, contradicting the choice of m . It is easy to see that the m^{th} term in the tracking sequence must be of the form $f_k':y'$ or $P_l(f_k':y')$, where y' is tight, well-formed, and $\min(y') \geq BIG(n, R', y)$. All the terms after the m^{th} term in the tracking sequence starting with x' must be of the form $f_k':z'$ or $P_l(f_k':z')$. Thus we can get a tracking sequence for y starting with y' that reaches e.n.f. By the inductive hypothesis, there exists an m'' such that $R'(y, m'')$ is n -transparent. Thus $R(x, m+m'')$ is $(n+1)$ -transparent. This completes the inductive step of our proof. $\square$

The proof that (2) implies (1) is now straightforward. Suppose every tracking sequence for (R, x) reaches e.n.f. Given n , we can choose (by Lemma 16.11) x' tight and well-formed such that $\rho(x') = x$ and $\min(x') \geq BIG(n, R, x)$. By Lemma 6.8, there is some tracking sequence for (R, x) starting with x' . By assumption, that tracking sequence reaches e.n.f. Then by Lemma 16.13, $R(x, m)$ is n -transparent for some m . Thus R is S -safe for x .

17. Appendix I: Characterizing L-safety

Recall that we are trying to show that (1) R is L -safe for x iff (2) for all $d \in Fin$ with $d \leq_I \mu(x)$, there exists a $\rightarrow_b$ tracking sequence $x'_0, x'_1, \dots$ for (R, x) such that for some n , x'_n is in e.n.f. and $d \leq_I \mu'(x'_n)$. In order to show that (1) implies (2), we need a preliminary lemma.

Lemma 17.1. *If y' is a well-formed star-free FP' expression such that $\rho(y') = R(x, n)$, then there exists a $\rightarrow_b$ tracking sequence $x'_0, x'_1, \dots$ for (R, x) with $x'_n = y'$.*

Proof. Let z' and z'' be FP' expressions. We show by induction on the structure of z' that if $\max(z') \leq \min(z'')$, $\rho(z') = \rho(z'')$, z'' is tight (see Definition 16.7), and z' is well-formed and star-free, then $z'' \rightarrow_b z'$ using only P -rules and lowering rules. The only non-trivial case is if z' is of the form $z'_1:z'_2$ while z'' is of the form $P_m(z''_1:z''_2)$. From the induction hypothesis,

it follows that $z''_i \rightarrow_b z'_i$, for $i = 1, 2$, so we also have $z'' \rightarrow_b P_m(z')$. Since z' is well-formed, we know that $\mu(z'_1)$ must be a function item (otherwise $\mu(z')$ would be $\perp$, so that z' would be of the form $\Omega(z)$). Now by Lemma 16.5, it follows that $\tau(z'_1) \leq \max(z'_1) \leq \max(z') \leq \min(z'') \leq m$. Thus $\tau(z') \leq m - 1 \leq m$, and hence $P_m(z') \rightarrow_a z'$. Thus $z'' \rightarrow_b P_m(z') \rightarrow_a z'$ using only P -rules and lowering rules as desired.

By Lemma 16.12 and Lemma 16.11, we can find a $(\rightarrow_a)$ tracking sequence $x'_0, \dots, x'_{n-1}, x''_n$ for the FP reduction sequence $R(x, 0), \dots, R(x, n)$ with x''_n well-formed, tight, and $\min(x'') \geq \max(y')$. Take $x'_n = y'$. Thus, by the result in the first paragraph, $x'_0, \dots, x'_n$ is a $\rightarrow_b$ tracking sequence for $R(x, 0), \dots, R(x, n)$. By Lemma 6.8, this $\rightarrow_b$ tracking sequence can be extended past n . It is interesting to note that the only $\rightarrow_b$ reductions (that are not also $\rightarrow_a$ reductions) were the ones used in rewriting x''_n to y' . $\square$

We can now show that (1) implies (2) in our characterization of L-safety. Suppose R is L-safe for x , $d \in \text{Fin}$, and $d \leq_I \mu(x)$. Since R is L-safe for x , there exists n such that $d \leq_I \mu_r(R(x, n))$. By part 4 of Lemma 7.1, there exists a star-free expression y' in e.n.f. such that $\rho(y') = R(x, n)$ and $d \leq_I \mu'(y')$. (2) easily follows from Lemma 17.1.

In order to show that (2) implies (1), assume (2) holds and we are given $d \in \text{Fin}$ with $d \leq_I \mu(x)$. We want to show that for some n , we have $d \leq_I \mu_r(R(x, n))$. By assumption, we can find a $\rightarrow_b$ tracking sequence $x'_0, x'_1, \dots$ for (R, x) such that for some n , x'_n is in e.n.f. and $d \leq_I \mu'(x'_n)$. By part 3 of Lemma 7.1, we have that $d \leq_I \mu'(x'_n) \leq_I \mu_r(R(x, n))$ which proves the result. $\square$

This characterization of L-safety was in terms in $\rightarrow_b$ tracking sequences. Does a similar characterization hold if we use $\rightarrow_a$ tracking sequences instead? As the following example shows, the answer is no!

Example 17.2. Let $x = \text{cons} : \langle \text{id} : \text{id}, \text{id} : \text{id} \rangle$, $y = \text{cons} : \langle \text{id}, \text{id} : \text{id} \rangle$, $z = \text{cons} : \langle \text{id} : \text{id}, \text{id} \rangle$, $w_0 = \text{cons} : \langle \text{comp} : \langle \text{comp}, \text{cons} : \langle K : \text{first}, \text{id} \rangle \rangle, \text{comp} : \langle \text{comp}, \text{cons} : \langle K : \text{last}, \text{id} \rangle \rangle \rangle : x$, $w_1 = \langle \text{comp} : \langle \text{first}, x \rangle, \text{comp} : \langle \text{last}, x \rangle \rangle$, and $w_2 = \langle \text{comp} : \langle \text{first}, y \rangle, \text{comp} : \langle \text{last}, z \rangle \rangle$. It is easy to check that $w_0 \rightarrow^* w_1 \rightarrow^* w_2$. Thus there exists a rewriting strategy R such that for some appropriately chosen m_1 and m_2 , we have $R(w_0, m_1) = w_1$ and $R(w_0, m_2) = w_2$. We further assume that $R(w_0, m) = w_2$ for $m \geq m_2$. It is easy to see that R is L-safe for w_0 . Moreover, if we take $d = \langle [a \rightarrow a], [a \rightarrow a] \rangle$, then $d \in \text{Fin}$ and $d \leq_I \mu(w_2)$. We now sketch a proof of the fact that if $w'_0, w'_1, \dots$ is a $\rightarrow_a$ tracking sequence for (R, w_0) and if, for some n , w'_n is in e.n.f., then it is *not* the case that $d \leq_I \mu'(w'_n)$.

So suppose we have such a tracking sequence for (R, w_0) and w'_n is in e.n.f. Assume the subexpression corresponding to x in w'_0 has the form $P'_{n_1}(\text{cons}_{m_2} : \langle P'_{n_3}(\text{id}_{n_4} : \text{id}_{n_5}), P'_{n_6}(\text{id}_{n_7} : \text{id}_{n_8}) \rangle)$, where we take the convention that the subscripts on P' range from 0 to ∞ , and $P'_0(t)$ is an abbreviation for $\Omega(\rho(t))$, $P'_m(t)$ is an abbreviation for $P_m(t)$ for $1 \leq m < \infty$, and $P_\infty(t)$ is an abbreviation for t . Let y' and z' be the subexpressions of w'_n that correspond to y and z

respectively. Since there is no way for any of the subscripts $n_2, \dots, n_8$ to change during the rewriting process, it is straightforward to check that y' has the form $P'_{m_1}(\text{cons}_{n_2}: \langle \text{id}_{\min(n_3, n_5)}, P'_{n_6}(\text{id}_{n_7}:\text{id}_{n_8}) \rangle)$ for some $m_1 \leq n_1$. Similarly, z' has the form $P'_{k_1}(\text{cons}_{n_2}: \langle P'_{n_3}(\text{id}_{n_4}:\text{id}_{n_5}), \text{id}_{\min(n_6, n_8)} \rangle)$ for some $k_1 \leq n_1$. We leave it to the reader to check that $\mu'(w'_n) = \langle \text{id}_{j_1}, \text{id}_{j_2} \rangle$ for some $j_1 \leq \min(m_1, n_2, n_3, n_4, n_5)$ and some $j_2 \leq \min(k_1, n_2, n_6, n_7, n_8)$. If $d \leq_I \mu'(w'_n)$, then we must have that $1 \leq j_1$ and $1 \leq j_2$. Thus, $1 \leq n_i$ for $i = 1, \dots, 8$, from which it follows that w'_n could not be in e.n.f., contradicting our initial assumption. (Of course, we could get it to e.n.f. by applying unsound $\rightarrow_b$ rules such as $P'_{n_6}(\text{id}_{n_7}:\text{id}_{n_8}) \rightarrow_b \Omega(\text{id}:\text{id}).$)

18. Appendix J: LL-completeness and LL-safety

In this section we define a semantic meaning function μ_{pr} that is just like μ_r except that it maps potential redexes to $\perp$, rather than just redexes. We then discuss analogues to limiting completeness and L-safety called LL-completeness and LL-safety. Since the details are very similar to those for μ_r , we just sketch them here.

We define μ_{pr} by induction on structure as follows:

$\mu_{pr}(x) = \mu(x)$ if x is an atom or primitive function symbol

$\mu_{pr}(\langle x_1, \dots, x_n \rangle) = \langle \mu_{pr}(x_1), \dots, \mu_{pr}(x_n) \rangle$

$\mu_{pr}(x:y) = \begin{cases} \mu_{pr}(x):\mu_{pr}(y) & \text{if } x:y \text{ is not a potential redex} \\ \perp & \text{if } x:y \text{ is a potential redex.} \end{cases}$

Not surprisingly, μ_{pr} has many of the same properties as μ_r . It does, however, have the additional property that if $x \rightarrow^* y$ then $\mu_{pr}(x) \leq_I \mu_{pr}(y)$. The following proposition describes the relationship between μ_r and μ_{pr} . In particular, part 3 says that the value given by μ_{pr} is in some sense the “lower limit” of the values given by μ_r .

Proposition 18.1.

1. $\mu_{pr}(x) \leq_I \mu_r(x)$.
2. $\forall x \exists y (x \rightarrow^* y \text{ and } \mu_{pr}(x) = \mu_r(y))$.
3. $\mu_{pr}(x) = \min \{ \mu_r(y) \mid x \rightarrow^* y \}$.

Proof. Part 1 is immediate from the definitions. Part 2 follows easily by induction on the structure of x . Part 3 follows from parts 1 and 2. $\square$

Using μ_{pr} , we can now define our last notion of having “enough” rules:

Definition 18.2. A set of rewrite rules is *LL-complete* iff for all FP expressions x and all $d \in \text{Fin}$ such that $d \leq_I \mu(x)$ there is an FP expression y such that $x \rightarrow^* y$ and $d \leq_I \mu_{pr}(y)$. R is *LL-safe for x* iff for all $d \in \text{Fin}$ such that $d \leq_I \mu(x)$, there exists n such that $d \leq_I \mu_{pr}(R(x, n))$. R is *LL-safe* iff it is LL-safe for all FP expressions x .

By part 2 of Proposition 18.1 and Theorem 7.4 it follows that FP has the LL-completeness property. By part 1 of Proposition 18.1, it follows that if R is LL-safe for x then it is also

L-safe for x , and so LL-safety implies L-safety. To see that the converse does not hold, let $x = \text{comp}: \langle \text{first}, \text{cons}:(\text{al}: \langle K:a, \text{id}: \langle \rangle \rangle) \rangle$. It is easy to check that $\mu_r(x) = \mu(x) = \mu(K:a)$, while $\mu_{pr}(x) = \perp$. So if R is a strategy that does nothing on x (i.e., $R(x, n) = x$ for all n), then R is L-safe for x but not LL-safe for x .

We would also like to show that S-safety implies LL-safety. To do this, we first give a characterization of LL-safety in terms of FP' tracking sequences. To understand the intuition behind this definition, recall that in Lemma 7.1 we showed that if x' is in e.n.f., then $\mu'(x') \leq_I \mu_r(\rho(x'))$. It is not necessarily the case that if x' is in e.n.f. then $\mu'(x') \leq_I \mu_{pr}(\rho(x'))$. For a counterexample, consider $x' = \text{al}_4: \langle a, \Omega(\text{id}: \langle a \rangle) \rangle$. It is easy to see that x' is in e.n.f., $\mu'(x') = \langle a, * \rangle$, and $\mu_{pr}(\rho(x')) = \perp$.

This expression x' is exactly the same one we used to show that it is possible for an FP' expression to be in e.n.f. and yet be rewritten to another expression which is *not* in e.n.f. We define an FP' expression to be in *almost normal form* (a.n.f.) if it is in e.n.f. and cannot be rewritten (using $\rightarrow_a$ rewriting) to another expression not in e.n.f. One can show that any well-formed FP' expression x' in e.n.f. can be rewritten to a.n.f. using at most a finite number of FP -style rules (indeed, the number is at most the number of occurrences of al and ar). We can now show that if x' is an FP' expression in a.n.f., then $\mu'(x') \leq_I \mu_{pr}(\rho(x'))$. Using a.n.f., we can now state our characterization for LL-safety.

Theorem 18.3. *Let R be a rewrite strategy. Then the following are equivalent:*

1. *R is LL-safe for x .*
2. *For all $d \in \text{Fin}$ with $d \leq_I \mu(x)$, there exists a $\rightarrow_b$ tracking sequence $x'_0, x'_1, \dots$ for (R, x) such that for some n , x'_n is in a.n.f. and $d \leq_I \mu'(x'_n)$.*

Proof. Just like that for Theorem 7.7, replacing e.n.f. by a.n.f. throughout. We omit details here. $\square$

It also turns out we can easily modify the proof of Theorem 6.9 to give a characterization of S-safety in terms of a.n.f.

Theorem 18.4. *Let R be a rewrite strategy. Then the following are equivalent:*

1. *R is S-safe for x .*
2. *Every tracking sequence for (R, x) reaches a.n.f.*

Corollary 18.5. *If R is S-safe for x then R is LL-safe for x . Thus, S-safety implies LL-safety.*

We remark that Example 7.9 can also be used to show that the converse does not hold.

References

- [Ba78] J.W. Backus, Can programming be liberated from the von Neumann style? A functional style and its algebra of programs, *Communications of the ACM* 21:8 (August, 1978), pp. 613-641.
- [Bo82] C. Böhm, Combinatory foundation of functional programming, *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming*, (August, 1982), pp. 29-36.
- [BMS80] R. Burstall, D. MacQueen, and D. Sannella, HOPE: An experimental applicative language, *Proceedings of the 1980 LISP Conference* (August, 1980), pp. 136-143.
- [DM84] W. Dosch and B. Moller, Busy and lazy FP with infinite objects, *Proceedings of the 1984 ACM Symposium on LISP and Functional Programming* (August, 1984), pp. 282-292.
- [FLO83] S. Fortune, D. Leivant, and M.J. O'Donnell, The expressiveness of simple and second-order type structures, *Journal of the ACM* 30:1 (January, 1983), pp. 151-185.
- [FW76] D. Friedman and D. Wise, Cons should not evaluate its arguments, *Proceedings of the Third International Colloquium on Automata, Languages, and Programming*, Springer-Verlag (1976), pp. 257-284.
- [GMP82] J. Goguen, J. Meseguer, and D. Plaisted, Programming with parameterized abstract objects in OBJ, in *Theory and Practice of Software Technology*, North-Holland (1982).
- [HW³85] J.Y. Halpern, J.H. Williams, E.L. Wimmers, and T.C. Winkler, Denotational semantics and rewrite rules for FP, *Proceedings of the Twelfth ACM Symposium on Principles of Programming Languages* (January, 1985), pp. 108-120.
- [HM76] P. Henderson and J.H. Morris, A lazy evaluator, *Proceedings of the Third ACM Symposium on Principles of Programming Languages* (January, 1976), pp. 95-103.
- [HO84] C.M. Hoffmann and M.J. O'Donnell, Implementation of an interpreter for abstract equations, *Proceedings of the Eleventh ACM Symposium on Principles of Programming Languages* (January, 1984), pp. 111-120.
- [MNV73] Z. Manna, S. Ness, and J. Vuillemin, Inductive methods for proving properties of programs, *Communications of the ACM* 16:8 (1973), pp. 491-502.
- [Me82] A.R. Meyer, What is a model of the λ -calculus?, *Information and Control* 52, (1982) pp. 87-122.
- [Ni85] F. Nickl, A denotational semantics for Backus' FP with infinite objects, Technical Report of the Fakultät für Mathematik und Informatik, Universität Passau (1985).
- [O'D77] M.J. O'Donnell, *Computing in Systems Described by Equations*, Lecture Notes in Computer Science No. 58, Springer-Verlag (1977).
- [PI77] G.D. Plotkin, LCF considered as a programming language, *Theoretical Computer Science* 5 (1977), pp. 223-255.
- [Ro67] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill (1967).

- [SP82] M.B. Smyth and G.D. Plotkin, The category-theoretic solution of recursive domain equations, *SIAM Journal of Computing* 11:4 (1982), pp. 761-783.
- [St77] J. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, MIT Press (1977).
- [Tu79] D.A. Turner, A new implementation technique for applicative languages, *Software Practice and Experience* 9 (1979), pp. 31-49.
- [Wa76] C.P. Wadsworth, The relation between computational and denotational properties of Scott's D_∞ -models of the lambda calculus, *SIAM Journal of Computing* 5 (1976), pp. 488-521.
- [Wa78] C.P. Wadsworth, Approximate reduction and lambda calculus models, *SIAM Journal of Computing* 7 (1978), pp. 337-356.
- [Wi81] J.H. Williams, Formal representations for recursively defined functional programs, in *Formalization of Programming Concepts*, Lecture Notes in Computer Science No. 107, Springer-Verlag (April, 1981), pp. 460-470.
- [Wi82] J.H. Williams, On the development of the algebra of functional programs, *ACM Trans. on Prog. Langs. and Sys.* 4:4 (October, 1982), pp. 733-757.