Minimal Backus FP is Turing Complete

Richard Statman[1]

1 **Department of Mathematical Sciences**
Pittsburgh, PA 15213
statman@cs.cmu.edu

Abstract

Cartesian monoids are rather simple algebraic structures of which you know many examples. They also travel under many pseudonyms such as Cantor algebras, Jonsson-Tarski algebras, and Frey-Heller monoids. John Backus's FP is just the theory of Cartesian monoids together with fixed points for all Cartesian monoid polynomials. Here we show that FP is a Turing complete programming system.

1998 ACM Subject Classification D.3.1, F.3.1

Keywords and phrases FP, Turing Complete

1 A minimal version of FP

In his 1977 Turing Award address, John Backus introduced the model of functional programming called "FP". FP is the decendant of Herbrand-Godel notion of recursive definablity and the ancestor of the programming language Haskell. One reason that FP is attractive is that it provides "an algebra of functional programs" ((b) pg 619 [1]). However, Backus did not believe that FP was powerful enough;

"FP systems have a number of limitations.....
If the set of primitive functions and functional
forms is weak, it may not be able to express every
computable function." (11.4.2 pg 623 [1])

and he moved on to stronger systems. It turns out that this is not correct (which is not to say that the stronger systems are not valuable). Here we shall show that is a system which is only a fragment of Backus FP can compute every partial recursive function.

In keeping with Backus'idea of an algebra of programs we can informally define our FP as the algebraic theory of Cartesian monoids with fixed points. We have written alot about Cartesian monoids. These monoids also have several aliases; Cantor algebras [10], Jonsson-Tarski algebras [5], and Freyd-Heller monoids ([4]). Much of this is due to the fact that they arise in the construction of groups with unsolvable word problems. However,in our connection Cartesian monoids were first defined by Lambek and Scott in 1980 ([6], [8]).
$C = (M, *, I, R, L, <>)$ is a Cartesian monoid if

(i) $I, L, R : M$
(ii) $<>: M \times M \to M$
(iii) $(M, *, I)$ is a monoid with identity I
(iv) $L* < a, b >= a$
(v) $R* < a, b >= b$
(vi) $I =< L, R >$
(vii) $< a, b > *c =< a * c, b * c >$.

The reader should compare this presentation with 11.2.3 and 11.2.4 of Backus' Turing lecture. There are many Cartesian monoids. All can be obtained by a Cayley type construction as follows. First take the monoid of all functions on an infinite set S and a surjective pairing function

$$(,) : S \times S \to S$$

with inverses $L, R : S \to S$. Lift $(,)$ to

$$<,>: ((S \to S) \times (S \to S)) \to (S \to S)$$

pointwise and take a substructure. However there is one example that we should single out; the free Cartesian monoid F defined as follows. The objects of the monoid F are the equivalence classes of the closed (variable free) monoid expressions determined by the congruence generated by the axioms above. That is, two expressions are equivalent if and only if they are equal in all Cartesian monoids (equivalently, are provably equal from the axioms above). The operations on the monoid are defined pointwise as follows - where, for expressions e, the congruence class of e is denoted e/F.

$I = I/F$
$f/F * g/F = (f * g)/F$
$L = L/F$
$R = R/F$, and
$< f/F, g/F >=< f, g > /F$.

It is a well known exercise of universal algebra to show that these definitions have all the desired properties. The Cartesian monoid polynomials are the Cartesian monoid expressions possibly containing free variables. We write $f = f(x_1, ..., x_n)$ if the polynomial f contains at most the variables (indeterminates) $x_1, ..., x_n$.

We now add to the axioms of Cartesian monoids the existence of fixed points. These are denoted by new constants 'p'
(viii) $p = f(p)$
for each Cartesian monoid polynomial $f(x)$. We will not use any general mechanism for introducing such constants such as a variable binder fix $(x, f(x))$. In particular the same monoid polynomial $f(x)$ can have multiple fixed points and the issue of whether or not they are equal will not arise. Here we note that such definitions as (viii) are generally weaker than recursive definitions since we do not have definition by cases. This should be compared to 11.2.5 of [1]. Of course, since we have pairing we automatically
get simultaneous fixed points
$p_1 = f_1(p_1, ..., p_n)$
$\vdots$
$p_n = f_n(p_1, ..., p_n)$

for polynomials $f_i(p_1, ..., p_n)$ with $i = 1, ..., n$. Thus we can also have fixed points of polynomials that have previously defined fixed points as parameters. One very interesting model for these axioms is the monoid of piecewise shift operators on Cantor space discussed in [11] and [12].

2 Rewriting FP terms

For axioms (i)-(viii) we have the equivalent rewrite system
(i) $L* < X, Y > \rightarrowtail X$
(ii) $R* < X, Y > \rightarrowtail Y$
(iii) $< X, Y > *Z \rightarrowtail < X * Z, Y * Z >$
(iv) $< L * X, R * X > \rightarrowtail X$

(v) $< L, R > \rightarrowtail I$
(vi) $I * X \rightarrowtail X$
(vii) $X * I \rightarrowtail X$
(viii) $p \rightarrowtail f(p)$

modulo the associativity axioms. This rewrite system is equivalent to the axioms in the sense that the smallest associative congruence containing the rules as identities is the congruence generated by the axioms. Note that the rewrite system is not left linear. It will be convenient to adopt the terminology of lambda calculus [2] for the rewrite system. So we will use the terminology of 'redex', 'reduct', and 'reduction' freely. In addition, we shall need to develope some of the "local structure" of lambda calculus reduction for our rewrite system. This includes much of Part III [2].

If \$ is a subset of the set of rules (i)-(viii) then the one step reduction relation generated by \$ is denoted

$$\overset{(\$)}{\rightarrowtail}$$

and the multi-step is denoted

$$\overset{(\$)}{\twoheadrightarrow}$$

Since (viii) redexes cannot overlap, there is the natural notion of the parallel reduction of a set of (viii) redexes in a term. This we denote by $\overset{(\text{viii})}{\Rightarrow}$ and the multi-step by $\overset{(\text{viii})}{\Rrightarrow}$

Examples of subsets of rules are

(1) \$1 = (i),(ii),(iii),(iv),(v),(vi),(vii),(viii)
(2) \$2 = \$1 - (viii)
(3) \$3 = (iv),(v),(vii)
(4) \$4 = \$1 - \$3
(5) \$5 = \$2 - \$3

Here we note that if Y is the result of a one step reduction $\overset{(\$5)}{\rightarrowtail}$ of X then each occurrence of each fixed point constant in Y has a unique ancestor in X defined in the obvious way.

We now interpert the expressions of the rewrite system in the non-negative integers. We define $o(X)$ by the following
$o(I) = o(L) = o(R) = 0(p) = 2$
$o(< U, V >) = o(U) + o(V) + 2$
$o(U * V) = o(U) * o(V)$.
Lemma: Suppose that the reduction

$$U \overset{(\$2)}{\rightarrowtail} V$$

Then $o(V) < o(U)$.
Proof; we illustrate with (iii)

$$< X, Y > *Z \rightarrowtail < X * Z, Y * Z >$$

where $o(< X, Y > *Z) =$
$(o(X) + o(Y) + 2) * o(Z) =$
$(o(X) * o(Z)) + (o(Y) * o(Z)) + (2 * o(Z)) >$
$(o(X) * o(Z)) + (o(Y) * o(Z)) + 2 =$
$o(< X * Z, Y * Z >)$

Since $o(Z) > 1$. End of proof.

Later we will generalize this interpertation to \$4 using ordinals less than epsilon 0.

Proposition 1; (1)$\overset{\$2}{\rightarrowtail}$ is strongly normalizing and Church-Rosser.
(2) $\overset{\$1}{\rightarrowtail}$ is Church-Rosser.

Proof. clearly (\$2) reductions satisfy the weak diamond property. Thus (1) follows immediately from lemma 1. Rewrites (viii), when executed as $\overset{(\text{viii})}{\Rrightarrow}$, are clearly Church-Rosser, and commute with $\overset{(\$2)}{\twoheadrightarrow}$ [7]. The diamond property for $\overset{(\$1)}{\rightarrowtail}$ follows from this. Thus we have (2). End of proof.

Thus w.l.o.g. we can assume that the polynomials $f(x)$ in (viii) rewrites are in $\overset{(\$2)}{\rightarrowtail}$ normal form. Next we have an analogue of eta postponement (section 15.1 [2]).

Proposition 2; (**\$3 postponement**) If $U \overset{(\$1)}{\twoheadrightarrow} V$ then there exists W such that $U \overset{(\$4)}{\twoheadrightarrow} W \overset{(\$3)}{\twoheadrightarrow} V$.

Proof; note first that if $X \overset{(\$3)}{\rightarrowtail} Y \overset{(\text{viii})}{\Rrightarrow} Z$ then there exists W such that $X \overset{(\text{viii})}{\Rrightarrow} W \overset{(\$3)}{\rightarrowtail} Z$.

Next we consider the case $U \overset{(\$2)}{\twoheadrightarrow} V$. We must show that there exists W such that $U \overset{(\$5)}{\twoheadrightarrow} W \overset{(\$3)}{\twoheadrightarrow} V$. This case is proved by induction on the entire $\overset{(\$2)}{\twoheadrightarrow}$ reduction tree of U which is finite by proposition 1 and Konig's lemma. In each subcase we consider the first rewrite in the reduction $U \overset{(\$2)}{\twoheadrightarrow} V$. In case this rewrite is not in \$3, the result follows from the induction hypothesis. In case the first rewrite belongs to \$3 we may apply the induction hypothesis to the result. Following this, we distinguish 12 sub-subcases. Almost all of these are routine. We give only a few to illustrate.

Case; (iv),(i)

$L* < L* < X, Y > *Z, R* < X, Y > *Z > \overset{(\text{iv})}{\rightarrowtail} L* < X, Y > *Z \overset{(\text{i})}{\rightarrowtail}$
$X * Z$

transformed to

$L* < L* < X, Y > *Z, R* < X, Y > *Z > \overset{(\text{i})}{\rightarrowtail} L* < X, Y > *Z \overset{(\text{i})}{\rightarrowtail}$
$X * Z$

Case; (v),(vi)

$< L, R > *Z \overset{(\text{v})}{\rightarrowtail} I * Z \overset{(\text{vi})}{\rightarrowtail} Z$

transformed to

$< L, R > *Z \overset{(\text{iii})}{\rightarrowtail} < L * Z, R * Z > \overset{(\text{v})}{\rightarrowtail} Z$

Case; (vii), (iii)

$< X, Y > *Z * I * W \overset{(\text{vii})}{\rightarrowtail} < X, Y > *Z * W \overset{(\text{iii})}{\rightarrowtail}$
$< X * Z * W, Y * Z * W >$

transformed to

$< X, Y > *Z * I* \overset{(\text{iii})}{\rightarrowtail} < X * Z * I * W, Y * Z * I * W > \overset{(\text{vii})}{\twoheadrightarrow}$
$< X * Z * W, Y * Z * W >$.

If the reader is puzzled by the difference between (vi) and (vii) she should consider the case (v),(vii). End of proof.

When V is normal there is an alternative proof almost identical to the proof of eta postponement in [2] 15.1.6. Now we can write each term U as $U_1 * ... * U_n$ where each U_i is I, L, R, p, or $< U_{i,1}, U_{i,2} >$. A top level \$1-(vii) redex is one whoes first symbol is one of the U_i. The entire redex is completely determined except in the cases (iii) and (vi). Now the analogue of the full standarization theorem of lambda calculus (section 11.4 [2]) fails for FP since we have reductions $< I, I > *p \rightarrowtail < I, I > *R * p < I * R, I * R > *p < R, R > *p$ for (viii) $p \rightarrowtail R * p$. However there is a version for \$4 reductions to \$5 normal forms which is enough for us. We now interpert the expressions of the rewrite system in the ordinal numbers less than epsilon 0 [9] We define

$w(0) = 2$
$w(1) =$ omega (the least infinite ordinal)
$w(n + 2) = w(0)^{w(n)}$ (ordinal exponentiation [9])
and we let # and @ denote "natural" sum and product resp. [9]. These have also been called "Hessenberg"sum and product by Sierpinski and "surreal" sum and product by Conway [3]. An ordinal valuation o is map from fixed-point constant occurrences in a term U into ordinals of the form $w(n)$. We extend o to a map from sub-expressions of U into ordinals ¡ epsilon 0 by the following
$o(I) = o(L) = o(R) = 2$
$o(< U, V >) = o(U)\#o(V)\#2$
$o(U * V) = o(U)@o(V)$.

An example is
o_n defined by$o_n(p) = w(n)$
for all occurrences of all constants in U. If we have a term U, with an ordinal valuation o defined on U, and

$$U \overset{(\$5)}{\rightarrowtail} V$$

then we induce an ordinal valuation on V by defining the value of a decendant to be the value of its ancestor. Since (iv) is not present this ancestor is unique. Par abus de notation we shall continue to refer to this as o. We now extend this to

$$U \overset{(\$4)}{\rightarrowtail} V$$

provided the instance of (viii) has $o(p) = w(n + 1) > 2$. In this case, we define $o(f(p)) = o_n(f(p))$, and we say that the reduction is o secured. Note that in this case the $l.h.s.p$ has no descendents and no fixed-point constant in the $r.h.s.f(p)$ has an ancestor. This gives a useful analogue for section 14 of [2].
Lemma 2; Suppose that the reduction

$$U \overset{(\$4)}{\rightarrowtail} V$$

is o secured. Then $o(V) < o(U)$.

Proof; this follows from the elementary properties of the natural sum and product of ordinals ¡ epsilon 0. It is not too different from the proof of lemma 1, as natural product distributes over natural sum. End of proof.

Lemma 3; Let $\mathcal{S}$ be a finite set of $\overset{(\$4)}{\twoheadrightarrow}$ reductions beginning with U. Then there exists an ordinal valuation o defined on U such that every member of $\mathcal{S}$ is o secured.

Proof: Just set $o = o_n$ for $n >$ the length of the longest member of $\mathcal{S}$. End of proof.

Proposition 3 (normal standardization); Suppose that $U \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} V$ with V in \$5 normal form. Then there is a \$4 reduction from U to V which contracts every left most top level \$4 redex first except those (viii) which have some descendants in V.

Proof; we suppose that we have a reduction $U \overset{(\$4)}{\twoheadrightarrow} V$ and apply lemma 3 to obtain an ordinal valuation o for which this reduction is o secured. By lemma 2, and Konig's lemma, the set of o secured reductions beginning with U is finite. This set of reductions satisfies the weak diamond property and is therefore Church-Rosser. We consider those U' such that there is an o secured reduction $U \overset{(\$4)}{\twoheadrightarrow} U'$
and an o secured reduction $U' \overset{(\$4)}{\twoheadrightarrow} V$ and prove the proposition by induction on $o(U')$. For readability purposes we omit the symbol' on U'. Consider, the leftmost top level \$4 redex in U. If this is a (vi) or an (viii), with no descendant in V, it is clear that this reduction can be done first. If the redex is (i) then

$$U = U_1 * ...U_{i-1} * L* < X, Y > *U_{i+2} * ... * U_n$$

where each U_j, for $j < i$, is L, R, or a p with a descendant in V. Since V is \$5 normal there exist X', Y', Z such that one step in the reduction has the form

$$U_1 * ...U_{i-1} * L* < X', Y' > *Z \rightarrowtail U_1 * ...U_{i-1} * X' * Z.$$

Here Z may be empty. Consider the first such step. Then there exist $Z_1, ..., Z_m$ such that

$U_{i+2} * ... * U_n \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} Z_1 * ... * Z_m * Z$
$X * Z_1 * ... * Z_m \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} X'$, and
$Y * Z_1 * ... * Z_m \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} Y'$.

Thus $U_1 * ...U_{i-1} * X * U_{i+2} * ... * U_n \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} V$ and the induction hypothesis applies to this term. The case for (ii) is symmetrical. The case for (iii) is similar. End of proof.

A term U with no top level \$5 redex has the form

$$U_1 * ... * U_n$$

where each U_i is L, R, or p except possibly $i = n$ where

(a) $U_n = I$, or
(b) $U_n =< X, Y >$ and if $n > 1$ then $U_{n-1} = p$.
In case (b), under the hypothesis of Proposition 3, V has the form

$$U_1 * ... * U_{n-1}* < W, Z >$$

with $X \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} W$ and $Y \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} Z$. Here, Proposition 3 can be applied recursively and a further \$4 reduction of U by reducing a redex in X or Y is said to be "lower level".

3 Simulation of Turing machines

We shall now give a simulation of any deterministic 1 tape Turing machine K, which reads from right to left. We assume that K has the finite state set $\{s, s', s", ...\}$ and alphabet $\{a, a', a", ...\}$. We assume that there is a designated initial state j and a designated blank symbol b. We shall need some extra marker symbols as well

@ = left end of tape
$\#$ = right end of tape
$+$ = signals the tape head to move right
! = signals termination
? = error message.

These will all be encoded into blocks of bits, where L encodes 1, R encodes 0 and * stands for concatenation. So the number of bits in each block should be at least log(|States|+|Alphabet| +5), say k. If the machine K is started in state j with the word w of length l printed on its tape then we will start with the term $W * T$ where W is a string of length $k(l+2)$ encoding $@w\#$ and T is a term, defined below by the fixed point axioms, which simulates the action of K. More generally, we will encode the word u (lowercase) by the term U (uppercase). We will maintain this notational convention below.

In addition, we shall need to define certain other terms M, N, P, Q, E by the fixed point axioms. In order to simplify the definitions we note that it is sufficient to give a list of conditions of the form

$$X * p = g(p)$$

for polynomials $g(p)$ and terms X, such that the X's are $*$'s of L's and R's, all are of the same length, and all are distinct. For example, we could write instead of $p = < p * R, p * L >$; $L * p = p * R$, and $R * p = p * L$. We may omit one or more of the X's with default being $X * p = X$.

The intuition behind the simulation is this. M controls the action from the right end of the tape except for inception, error, and termination. The current state of the machine will be encoded immediately to the right of the letter currently being read by the machine. This is initiated by T and T does nothing else. M sends out a drone N to find the current state and simulate the move made by the Turing machine. The drone can move from right to left but not from left to right since movement is achieved by reductions (i) and (ii). A simple example is the drone defined by $p = < p * R, p * L >$ which reverses bits (forever)

$$L * R * R * L * L * R * p \twoheadrightarrow p * R * L * L * R * R * L.$$

The timing of these drones is irrelevant because we have normal standardization. To move right, N must leave a marker which is sought by another drone and moved successively to the right until M is reached. Then, new specialized drones P are dispatched to execute the move of the state to the right. In each case care must be exercised at the boundaries of the tape; this is the job of Q.

We now proceed to the general definition;

$\# * T$	$=$	$J\# * M$	J is the code of the initial state
$A * \# * M$	$=$	$A * N * \# * M$	for each letter a
$S * \# * M$	$=$	$S * Q * \# * M$	for each state s
$+ * \# * M$	$=$	$P * \# * M$	
$! * \# * M$	$=$	$\#$	
$? * \# * M$	$=$	$E * ? * \#$	
$X * E$	$=$	E	for any symbol X not @
$@ * E$	$=$	$@$	
$X * A * N$	$=$	$X * N * A$	for any symbol X except + or !
$+ * A * N$	$=$	$A * +$	for any letter a
$! * A * N$	$=$	$A * !$	for any letter a
$X * S * N$	$=$	$X * !$	for any symbol X if s is final
$A * S * N$	$=$	$S' * A$	if K in state s reading a prints a' goes into state s' and moves left
$A * S * N$	$=$	$A' * S' * +$	if K in state s reading a prints a' goes into state s' and moves right

$@ * S * N$	$=$	$@ * B * S$	if s is not final add a blank B
$A * A' * P$	$=$	$A * P * A'$	for any letters a, a'
$S * A * P$	$=$	$A * S$	for any state s and letter a
$A * S * Q$	$=$	$S' * A'$	if K in state s reading a prints a' goes into state s' and moves left
$A * S * Q$	$=$	$?$	if K in state s reading a moves right
$A * S * Q$	$=$	$A*!$	if s is final

Lemma 4; If the machine K is started in it initial state j with the word w printed on its tape and halts with the word u printed on its tape then

$$@ * W * \# * T = @ * U * \#.$$

Proof; indeed, $@ * W * \# * T \overset{\$4-\text{(iii)}}{\twoheadrightarrow} @ * U * \#.$
End of proof.

Lemma 5; If $@ * W * \# * T = @ * U * \#$ then K started in its initial state j with the word w printed on its tape will halt with the word u printed on its tape.

Proof; since $@ * U * \#$ is normal we have $@ * W * \# * T \overset{(\$1)}{\rightarrowtail\!\!\!\rightarrow} @ * U * \#.$

By Proposition 2 there exists V such that $@ * W * \# * T \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} V$

and $V \overset{(\$3)}{\rightarrowtail\!\!\!\rightarrow} @ * U * \#$. Now V is (viii) normal therefore it is \$4 normal. Thus by Proposition 3 there is a \$4 reduction from $@ * W * \# * T$ to V which contracts every left most top level \$4 redex first (there are no (viii) redexes which have descendants in V). This can easily be seen to be the simulation of a K computation until a fixed-point constant free term is obtained by

$$W*! * \# * MW * \#$$

(or $@ * E@$ which is impossible by choice of K). But then $V = @ * U * \#$ because any \$4 normal such reduct must begin with @. End of proof.

Lemmas 1-5 establish the following.

Theorem 1; FP is a Turing complete formal system.

4 Extensions of Theorem 1

In this section we present several extensions of theorem 1. Theorem 2 addresses the connection to the model of partial piewise shift operators. Theorem 3 addresses the connection to the theory of semigroups.

A binary relation % on finite binary sequences is said to be SLC (strongly left compatible) if the following conditions holds

$$u\%v\text{iff0u\%0vand1u\%1v}.$$

A partial function G on finite binary strings is said to be FP representable if there exists a term T such that

$$G(u) = v\text{iffU} * \text{T} = \text{V}$$

An equivalence relation % on finite binary sequences is said to be FP algebraic if there exists a polynomial $f(x)$ such that

$$u\%v\text{ifff(U)} = \text{f(V)}.$$

Here, we assume that the binary sequences u and v have been encoded by possibly longer strings of L's and R's, U resp. V, as in Theorem 1, but not for SLC.
Theorem 2; The following are equivalent
(1) G is partial recursive and its graph is SLC
(2) G is FP representable.

Proof; that (2) $\Rightarrow$ (1) is obvious. Now suppose (1). First we need to make some modifications in the notion of a Turing machine K (without input) with an oracle O. Here, when K's program queries O, it only asks whether the smallest yet unqueried integer is in the set represented by O. K can keep track of the initial segment of integers queried so far, and calls to the oracle are only exchanges of one bit of information. It is clear that every ordinary Turing machine with oracle can be simulated by such a modified machine. In addition, we require that K has an end of tape marker @ and that it only queries O when it is currently scanning @. The result of the query should be printed at the end of the tape and @ moved one square left. Now we take a specific machine K which enumerates the graph of G in some order and queries the oracle O establishing an initial segment of the characteristic function of the set represented by O until one is reached which is an extension wu of the first coordinate of an already enumerated pair (u, v). Then wv is output. Of course the output of K depends on the enumeration, and so it is very possible that $G(u) = v$ but only wv is output after wu is returned by O. If $G(u) = v$ then for every infinite f extending u, when the oracle O represents f, there is an initial segment wu such that wv is output. Thus by Konig's lemma, there exists an integer n such that for every finite binary seq. w of length n, K outputs wv for O returning wu after queries. It is this property which we exploit for the FP representation. Now we construct the term T very much like the term T doing the simulation in (III). The main difference is an auxilliary term C which, when the current state S is a query state, searches for @ and
$L * @ * S * C = @ * 0 * S'$ if K transitions into state s' from the query response 0
$R * @ * S * C = @ * 1 * S$" if K transitions into state s" from the query response 1.
Clearly if K outputs v from u then $U * T = V$. The problem occurs when $G(u) = v$ but K does not output v from u. In this case we have $U{*}T = @{*}S{*}C{*}X$ for some X. But then by the property above, there exists an integer n such that for every binary sequence w of length n, we have that K outputs wv for O returning wu. Thus for every word w of length $n, W * U * T = W * @ * S * C * X = W * V$. Thus by axiom (vi), $U * T = V$. Conversely suppose that $U * T = V$.

Then $U * T \overset{(\$4)}{\twoheadrightarrow} V$ by a reduction which proceeds as follows. By Proposition 2 there exists a W such that $U * T \overset{(\$4)}{\twoheadrightarrow} W \overset{(\$3)}{\twoheadrightarrow} V$. Since V is \$1 normal, W is fixed-point constant free. If Z is the \$4 normal form of W then $Z \overset{(\$3)}{\twoheadrightarrow} V$ by Proposition 1 (1). By Proposition 2 there is a \$4 reduction from $U * T$ to Z which contracts every left most top level \$4 redex first and then proceeds by similar low level reductions if they exist. We shall prove that $u \% v$ by induction on the length of Z. Now K can simulate this reduction until a term $@ * S * C * X$ is obtained. This term top level reduces to

$$< @ * 0 * S' * X, @ * 1 * S" * X >$$

with a (iii) and Z has the form $< Z', Z" >$. So, we have the reductions $L{*}U{*}T \overset{(\$4)}{\twoheadrightarrow} @{*}0{*}S'{*}X \overset{(\$4)}{\twoheadrightarrow} Z'^{(\$3)} L * V$ and $R * U * T \overset{(\$4)}{\twoheadrightarrow} @ * 1 * S' * X^{(\$4)} Z" \overset{(\$3)}{\twoheadrightarrow} R * V$, to which the induction hypothesis applies. Thus $u\%v$ since % is SLC. End of proof.

Let $\mathcal{S}$ be a set of fixed-point constant occurrences in U. A "development" of U is a \$4 reduction of U where (viii) reductions are only made on descendants of members of $\mathcal{S}$. The development is "complete" if the result contains no descendants of members of $\mathcal{S}$ (see 11.2.11 [2]). An analogue of the finiteness of developments theorem 11..2 [2] is

Fact 1 (FD!); Every complete development of U eventually terminates in the same \$5 normal term.

Proof; Use the ordinal valuation o_1 on members of $\mathcal{S}$ and o_0 on the other fixed-point constant occurrences. End of proof.

An analogue of the Strip Lemma (1.1.9 [2]) is

Fact 2 (parallel moves); If $\mathcal{S}$ is a subset of the fixed-point constant occurrences in U and both
$U \overset{(\$4)}{\twoheadrightarrow} Z$ by a complete development of \$, and
$U \overset{(\$4)}{\twoheadrightarrow} V \overset{\text{(viii)}}{\Longrightarrow} W$,

where the second reduction is a complete development of the all descendents of $\mathcal{S}$ in V, then
$Z \overset{(\$4)}{\twoheadrightarrow} W$.

Proof; by induction on the length of the reduction $U \overset{(\$4)}{\twoheadrightarrow} V$. The basis case is subsumed under Fact 1. For the induction step assume that $U \overset{(\$4)}{\twoheadrightarrow} V' \overset{(\$4)}{\rightarrowtail} V$". Applying the induction hypothesis to

$$U \overset{(\$4)}{\twoheadrightarrow} V' \overset{(\$4)}{\twoheadrightarrow} W'$$

where the second reduction is a complete development of the all descendents of $\mathcal{S}$ in V' we obtain $Z \overset{(\$4)}{\twoheadrightarrow} W'$. Now if the reduction $V' \overset{(\$4)}{\twoheadrightarrow} V$" is in \$5 then, by Fact 1, the complete developement of all the descendants of $\mathcal{S}$ in V" ends in W and we are done. Otherwise the reduction
$V' \overset{(\$4)}{\rightarrowtail} V$" is an (viii)

$$p \rightarrowtail f(p).$$

Now let \$ be all the descendants of members of $\mathcal{S}$ in V' plus the occurrence of p contracted to obtain V" and apply fact 1. End of proof.

An analogue of Klop's cofinality theorem 13.6.14 [2] is

Proposition 4 (cofinality); Suppose that we have a sequence of terms U_n such that $U_n{}^{(\$4)}U_{n+1}$ and for each n

(a) U_n is \$5 normal
(b) no fixed-point constant occurrence in U_{n+1} is the descendant of one in U_n

then if U_0V there is some n such that $V \overset{(\$4)}{\twoheadrightarrow} U_n$.

Proof; The proof is by induction on the length of the reduction $U_0 \overset{(\$4)}{\twoheadrightarrow} V$. For the induction step we suppose that

$$U_0 \overset{(\$4)}{\twoheadrightarrow} W \overset{(\$4)}{\rightarrowtail} V$$

and by induction hypothesis there exists n such that

$$W \overset{(\$4)}{\twoheadrightarrow} U_n.$$

Now for any subset $\mathcal{S}$ of the fixed-point constants in W there are no descendants in U_{n+1} by the reduction

$$W \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} U_n \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} U_{n+1}.$$

Thus, by Fact 2, $V \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} U_{n+1}$. End of proof.
Theorem 3; The following are equivalent

(1) % is recursively enumerable
(2) % is FP algebraic.

Proof; that (2) implies (1) is immediate. Now suppose (1). Consider a machine K which takes an an input an integer n and a word u. K then does n steps in the enumeration of those v such that $u\%v$ and prints the integer $n+1$ and the shortest such v on its tape. We now construct a term T as in the above simulation with certain modifications. The modifications are necessary since K is not meant to halt on any of the considered inputs.

First, we assume that the integer k in the encoding is sufficiently large so that R^k and L^k are not codes of any symbols so far considered. We encode the integer n by $R^k * (L^k)^n$. Thus we need not consider separately any K symbol used to separate the pair of inputs. Next we replace the equation $! * \# * M = \#$ by

$$! * \# * M = \# * T.$$

Now set $f(x) = @ * x * R^k * \# * T$. Suppose that $f(U) = f(V)$ for the codes of u and v. The construction in Lemma 3 gives us reductions

$@ * U_0 * R^k * \# * T \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow} \ldots \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow}$
$@ * U_n * R^k * (L^k)^n \# * T \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow} \ldots$

$@ * V_0 * R^k * \# * T \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow} \ldots \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow}$
$@ * V_n * R^k * (L^k)^n \# * T \overset{(\$4-\text{viii})}{\rightarrowtail\!\!\!\rightarrow} \ldots$

for $U_0 = U$ and $V_0 = V$. By Proposition 4 both of these reduction sequences are cofinal. By Church-Rosser there exists Z such that

$@ * V_0 * R^k * \# * T \overset{(\$1)}{\rightarrowtail\!\!\!\rightarrow} Z \overset{(\$1)}{\leftarrow\!\!\!\leftarrowtail} @ * U_0 * R^k * \# * T.$
Moreover by Proposition 2 exists term W, W' such that

$@ * V_0 * R^k * \# * T \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} W \overset{(\$3)}{\rightarrowtail\!\!\!\rightarrow} Z$
$@ * U_0 * R^k * \# * T \overset{(\$4)}{\rightarrowtail\!\!\!\rightarrow} W' \overset{(\$3)}{\rightarrowtail\!\!\!\rightarrow} Z$

Moreover, we may assume that Z, W, and W' are \$5 normal. Thus by Proposition 3 there is a \$4 reduction from $@{*}V_0 * R^k * \# * T$ to W which contracts every left most top level \$4 redex first except those T which are ancestors of the last T. Now the first part of the reduction to W is just an initial segment of the cofinal reduction of

$$@ * V_0 * R^k * \# * T.$$

This does not achieve an \$5 normal term until we have one of the $@ * V_m * R^k * (L^k)^m \# * T$. But this term is \$3 normal so W coincides with Z. Similarly for W'. Thus for some n, $@ * U_n * R^k * (L^k)^n \# * T$ is identical to

$@ * V_m * R^k * (L^k)^m \# * T$; so $n = m and U_n = V_n$. Hence $u\%v$. End of proof.

For a small application we suppose that we have a finitely generated semigroup with recursively e numerable word problem. We assume that the generators are encoded as strings of L's and R's and words u on these generators are encoded as strings U. Now, by the Church-Rosser theorem the semigroup generated by L and R is the same semigroup as the one generated by L and R in F and this is free. Thus there is a homomorphism h onto . The fibers of h form an RE partition to which we can apply Theorem 3. Thus there is a polynomial f such that $u = v$ in if and only if $f(U) = f(V)$. So we can define a copy of on the terms $f(U)$ by defining the product $\times$

$$f(U) \times f(V) = df f(U * V).$$

It would be very interesting to know when $\times$ can be defined by a polynomial. It is clear from Theorem's 2 and 3 that this can be done when has a recursive word problem.

References

[1] John Backus. Can programming be liberated from the von neumann style?: A functional style and its algebra of programs. *Commun. ACM*, 21:613641, 1978.

[2] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland Pub- lishing Company, 1981.

[3] J.H. Conway. On numbers and games. *CRC Press*, 1976.

[4] Peter Freyd and Alex Heller. Splitting homotopy idempotents II. *Journal of Pure and Applied Algebra*, 89:93106, 1993.

[5] B. & Tarski A. Jonsson. On two properties of free algebras. *Math. Scand.*, 9:95101, 1961.

[6] J. Lambek. From lambda calculus to cartesian closed categories. To H.B. Curry; Essays in Combinatory Logic..., 1980.

[7] B.K. Rosen. Tree manipulation systems and church-rosser theorems. Journal of the A.C.M., 20:160187, 1973.

[8] D. Scott. Relating theories of the lambda calculus. *To H.B. Curry; Essays in Combinatory Logic...*, 1980.

[9] W. Sierpinski. *Cardinal and Ordinal Numbers*, volume Polska Akademia Nauk Monografie Matematyczne 334. Panstwowe Wydawnictwo Naukowe, Warsaw, 1958.

[10] D.M. Smirnov. Cantor algebras with one generator. *Algebra and Logic*, 10:4049, 1971.

[11] R. Statman. Cartesian monoids. *Electronic Notes in Theoretical Computer Science*, 265:437451, 2010.

[12] R. Statman. Representation of cartesian monoids in and around the piecewise shift opera- tors. *Math. Struct. in Comp. Science*, forthcoming.