stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 130/80 JANUARI

J.V. TUCKER

COMPUTING IN ALGEBRAIC SYSTEMS

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

1980 Mathematics subject classification: 03D01, 03D35, 03D75, 03D80, 68B10, 68C30, 68C40

ACM-Computing Review-categories: 5.24, 5.26, 5.27

Computing in algebraic systems[*)]

by

J.V. Tucker

ABSTRACT

We describe the basic recursion theory of functions computable by different kinds of register machines designed to operate within an arbitrarily chosen algebraic system. Especial emphasis is placed upon computation in natural algebraic systems such as groups, rings and fields. A useful theorem about the topological structure of computable subsets of a Hausdorff topological algebra is proved.

KEY WORDS & PHRASES: *algebraic register machines, finite algorithmic procedures, algorithmically decidable and undecidable problems in Algebra, computable functions on algebraic data types*

[*)] This report will be published elsewhere and is not for review.

INTRODUCTION

Given a relational structure A, imagine an A-*register machine* which can hold in its registers a fixed, finite number of elements of A, perform the basic operations and decide the basic relations on these elements, and manage some simple manipulations and decisions such as to replace the contents of one register by those of another and to tell when two registers carry the same element. Next, picture an A-*register machine with counting* by adding a finite number of *counting registers* to an A-register machine; these carry natural numbers and the device is able to put zero into a counting register, add or subtract one from the contents of any counting register, tell if two registers contain the same number, and so on. Thirdly, there is the A-*register machine with stacking* which augments an A-register machine with a single *stack register* into which the entire contents of the ordinary algebraic registers of the basic machine can be temporarily placed at various points in the course of a calculation. Thus, the combinatorial operations of the A-register machine are extended in the first instance by permitting subcomputations on the natural numbers ω and in the second by prolonging the number and complexity of entirely algebraic subcomputations. On arranging both we have an A-*register machine with counting and stacking.*

To use one of these machines to compute a partial function on A is to write down the familiar finite program of instructions referring to whatever activities of the machine are available and containing information to stop in certain circumstances. Such a program is called a *finite algorithmic procedure*, a *finite algorithmic procedure with counting*, a *finite algorithmic procedure with stacking*, or a *finite algorithmic procedure with counting and stacking* accordingly as it includes or ignores instructions involving the counting and stack facilities. These are abbreviated in turn by *fap*, *fapC*, *fapS* and *fapCS*. The classes of all functions over A they compute, FAP(A), FAPC(A), FAPS(A) and FAPCS(A), are the subject of this paper.

The idea of the A-register machine first appeared in G.T. Herman & S.D. Isard [13] and in H. Friedman [7], there together with its counting facility. The stack mechanism belongs to

[22,23] in which J. Moldestad, V. Stoltenberg-Hansen and I investigated recursion-theoretic properties of the four families of fap-computable functions. This third article is a set of notes which introduces such models of computing into an algebraic *milieu* to settle decision problems: *The membership question for finitely generated ideals of polynomial rings over fields is fapCS-computable* (Theorem 3.2). *The membership question for single generator subgroups of the torus groups is fap-semicomputable, but not fapCS-computable* (Example 6.4). And to settle theoretical issues about computation: *For* A *an "everyday algebraic system of Mathematics",* FAPC(A) = FAPCS(A) (Theorem 5.2). *But, for example, if* A *is the algebraic closure of some finite field* $\mathbb{Z}_p$ *then* FAPS(A) $\subsetneqq$ FAPC(A) (Theorem 5.9).

The intention is to map out *Elementary* Recursion Theory in an algebraic setting: enough to show what peculiarities arise in lifting the contents of, say, Machtey & Young's book [20], stopping short of the Turing degree theory of the various fap-semicomputable sets which seems mysterious outside the finitely generated algebras (c.f. Section 4 and Theorem 5.10). The first two sections summarise formal definitions and certain relevant properties of the functions from [22,23,24]. The next two sections establish the local algebraic character of machine computations while Section 5 is about counting and stacking. Section 6 contains a useful theorem about fapCS-computable subsets of topological algebras.

Although the emphasis is on the classes of functions computed over natural algebraic systems, and not on the structure of programs and their semantics, the material here does usefully connect with the literature of Theoretical Computer Science: it provides the sort of basic technical information needed to fit de Bakker's mathematical semantics of program control structures based on ω [2] to the ADJ Group's algebraic theory of data types [8], for example. A specific technical application to program semantics is [3] by J.A. Bergstra, J. Tiuryn and myself.

My interest in Generalised Recursion Theory I owe to my time in Oslo, to J.E. Fenstad, D. Normann, S.S. Wainer and, especially, J. Moldestad and V. Stoltenberg-Hansen; in this connection I gratefully acknowledge the support of a fellowship from the European Programme of The Royal Society, London. These notes have also profited from information and advice from P.R.J. Asveld, J.A. Bergstra, B. Birkeland, H. Rolletschek, and J.I. Zucker.

NOTATION Throughout we are concerned with relational structures of finite signature. The word function, unqualified, will mean partial function and typically these will be (n,m)-ary maps either $A^n \times \omega^m \to A$ or $A^n \times \omega^m \to \omega$, a distinction we preserve in the abbreviation $A^n \times \omega^m \to A/\omega$. For $(a,x) \in A^n \times \omega^m$, $f(a,x) \simeq g(a,x)$ means f(a,x), g(a,x) are both defined and equal or are both undefined. On fixing part of the argument $a \in A^n$ of a function $f:A^{n+m} \to A$ we write $f(a):A^m \to A$. On extending a unary function $f:A \to A$ to an n-fold product $f\times\ldots\times f:A^n \to A^n$, we write for $a = (a_1,\dots,a_n) \in A^n$ the value $f(a)$ for $(fa_1,\dots,fa_n)$. Relations may be identified with their characteristic functions using 0 for *true*, 1 for *false*. The complement of a set A is denoted $\neg A$.

$\mathbb{Z}$, $\mathbb{Z}p$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ are the integers, the integers modulo p, the rationals, reals and complex numbers respectively.

For those ideas and facts of Algebra left unexplained in the text consult the books of A.G. Kurosh [18] and Mal'cev [21] on Universal Algebra, that of Kurosh [16,17] on Group Theory and B.L. van der Waerden [25] for Field Theory.

1. FINITE ALGORITHMIC PROCEDURES

The program language for A-register machines has variables $r_0, r_1, r_2, \dots$ for *algebra registers*; its constants, function and relation symbols are those of the signature of A together with new constants T for *true*, F for *false*, H for *halt*, 1,2, ... for *instruction labels* or *markers* and new relation = for *algebraic equality*.

A *finite algorithmic procedure* is a finite ordered list of labelled instructions $I_1,\dots,I_\ell$ which are of these types. The *algebraic operational instructions* manipulate elements of A and are

$r_\mu := \underline{a}$ meaning "*replace the contents of register* r_μ *by the element* $a \in A$ *named by* $\underline{a}$".

$r_\mu := \sigma(r_{\lambda_1},\dots,r_{\lambda_k})$ meaning "*apply the* k-*ary operation* σ *of* A *to the contents of registers* $r_{\lambda_1},\dots,r_{\lambda_k}$ *and replace the contents of* r_μ *by this value*".

$r_\mu := r_\lambda$ meaning "*replace the contents of register* r_μ *with that of* r_λ".

The *algebraic conditional instructions* determine the order of executing instructions and are

<u>if</u> $R(r_{\lambda_1},\dots,r_{\lambda_k})$ <u>then</u> i <u>else</u> j meaning "*if the* k-*ary relation* R *is true of the contents of* $r_{\lambda_1},\dots,r_{\lambda_k}$ *then the next instruction is* I_i *otherwise it is* I_j".

<u>if</u> $r_\mu = r_\lambda$ <u>then</u> i <u>else</u> j which takes its obvious meaning and, incidentally, gives us <u>goto</u> i.

Finally, $r_\mu := T$, $r_\mu := F$ marking true and false, and H meaning "*stop*" are included.

For the A-register machine with counting the language is extended by variables $c_0, c_1, c_2, \dots$ for *counting registers*, a constant 0 for *zero*, function symbols +1 for *successor*, $\dot{-}1$ for *predecessor*; and relational symbol = for *numerical equality*. The *counting instructions*, which when mixed with fap instructions make *finite algorithmic procedures with counting*, are simply the usual instructions for (natural number) register machines:

$c_\mu := 0$ $c_\mu := c_\lambda + 1$ $c_\mu := c_\lambda \dot{-} 1$ <u>if</u> $c_\mu = c_\lambda$ <u>then</u> i <u>else</u> j

and take their obvious meanings.

For the A-register machine with stacking we add to the A-register machine language a variable s for *stack register* and $\emptyset$ for *empty*. The new instructions are

<u>stack</u>$(z, r_0, \dots, r_m)$ meaning "*along with instruction*

marker z, *copy the contents of* $r_0,\ldots,r_m$ *as an* (m+1)-*tuple into the stack*".

<u>restore</u>$(r_0,\ldots,r_{j-1},r_{j+1},\ldots,r_m)$ meaning "*remove the last or topmost vector in the stack and replace the contents of the* r_i $(i\neq j)$ *by the corresponding entries of the vector*".

<u>if</u> $s=\emptyset$ <u>then</u> i <u>else</u> <u>marker</u> meaning "*if the stack is empty then the next instruction is* I_i *otherwise it is the instruction labelled by the marker in the topmost element in the stack*".

A *finite algorithmic procedure with stacking* is an ordered list of fap instructions and stack instructions with the convention that all the ordinary algebraic register variables are included in every <u>stack</u>-instruction, and all but one of them are included in every <u>restore</u>-instruction; the rôle of the marker is to remember at which points in the program the basic A-register machine is cleared for an independent subcomputation. For a proper account of the stack and its operation the reader should consult [22], but the description given is adequate for what follows. The definition of a *finite algorithmic procedure with counting and stacking* is immediate though we note that the stack allows only algebra elements to be stored.

To compute $A^n \times \omega^m \to A/\omega$ with program α choose n of its algebraic variables, and m of its counting variables, as *input variables*, and name a suitable variable as *output variable*. On fixing a machine appropriate for α load input argument $(a,x) \in A^n \times \omega^m$ into the registers α reserves as inputs, and make the remaining registers empty. The instructions of α are executed on the machine in the order they are given except where a conditional instruction directs otherwise. If at some stage an instruction cannot be carried out, such as happens when one applies a fap conditional to empty registers, then the computation is said to *hang* in that state and no value is computed. If the machine halts then the output value of the computation $\alpha(a,x)$ is the element contained in the output register named by α, if this is not empty; in all other circumstances, no output value is obtained. Converging and diverging computations are distinguished by $\alpha(a,x)\downarrow$ and $\alpha(a,x)\uparrow$, respectively, and, as usual, $\alpha(a,x)$ also denotes the (defined or undefined) value of a computation.

The four types of fap-computable function, relation or set can now be formally defined in the obvious way. A set or relation $S \subset A^n \times \omega^m$ is, say, *fapCS-semicomputable* if it is the domain of a fapCS-computable function; fap/fapC/fapS-semicomputability is defined similarly, of course. The different sets of computable functions $A^n \to A$ are denoted ${}^n\mathrm{FAP}(A)$, ${}^n\mathrm{FAPC}(A)$, ${}^n\mathrm{FAPS}(A)$, ${}^n\mathrm{FAPCS}(A)$.

For those acquainted with Theoretical Computer Science it ought to be pointed out that the hierarchy of low to high-level languages characteristic of computing *praxis* is also an essential feature of *theoria*: to prove a relation on an algebra A decidable, the richer the programming language the better whereas to prove it undecidable the opposite is true. Here our needs are best met by defining the various fap-computable functions through a crude "assembler code" for A-register machines, but features like the so called *structured control statements, recursion,* and *data type creation* can be encorporated into the definitions of FAP(A), FAPS(A) and FAPCS(A) respectively. Thus, to prove the set $FO(G)$ of all elements of finite order in a group G is fap-semicomputable one may use this program wherein r_I and r_0 are input and output variables:

$r:=r_I$; while $r \neq 1$ do $r:=r.r_I$ od; $r_0:=T;H$.

In conclusion, here are some technical ideas. A *state description* in a machine computation under fapCS α is typically a list

$(k;a_1,\ldots,a_p,x_1,\ldots,x_q;(z_1;a_{11},\ldots,a_{1p}),\ldots,(z_s;a_{s1},\ldots,a_{sp}))$

where $a_1,\ldots,a_p$ are the contents of the algebra registers named by α; $x_1,\ldots,x_q$ are those of the counting registers; there are s vectors piled in the stack register, z_i is the marker of the i-th element and a_{ij} the element in the j-th register of the i-th stored vector. And, finally, k is the number of the instruction in α which is to be applied to these elements. Each state description represents a *step* in a computation. Often we need to *unfold* a computation $\alpha(a,x)$ into all its stages and then we use

$D_i(\alpha,a,x) = (m_i,a_{ij},x_{ij};\ (z_{ij},a^i_{jk}))$

to denote the i-th step in the computation $\alpha(a,x)$. The length $|\alpha,a,x|$ of the computation $\alpha(a,x)$ is by definition the ordinal number of steps in the unfolding of $\alpha(a,x)$.

All four sets of programs we assume gödel numbered in the usual way with $\Omega_0,\Omega_C,\Omega_S,\Omega_{CS}$ denoting the code sets for fap, fapC, fapS and fapCS programs respectively. We also assume the *term* or *polynomial algebra* $T[X_1,\ldots,X_n]$ of any signature to be coded uniformly in n, ${}^n\gamma_*: {}^n\Omega \to T[X_1,\ldots,X_n]$ with the abbreviation ${}^n\gamma_*(i) = [i]$. Each term $t(X_1,\ldots,X_n)$ defines a function $A^n \to A$ by substituting algebra elements for indeterminates and we define n-ary term evaluation ${}^nTE:{}^n\Omega \times A^n \to A$ by ${}^nTE(i,a) = [i](a)$. Finally, define for $t,t' \in T[X_1,\ldots,X_n]$ $t \equiv_A t'$ if, and only if, for all $a \in A^n$, $t(a) = t'(a)$.

2. THE FAP-COMPUTABLE FUNCTIONS IN THE LARGE

To sketch essential background information from [22,23,24] about the recursion-theoretic properties of fap-computable functions, we begin with an axiomatisation of the large-scale structure of the partial recursive functions on ω.

In summary, a set of functions Θ over A is a *computation theory over* A *with code set* $C \subset A$ if associated with Θ is a surjection $\varepsilon: C \to \Theta$, called a *coding* and abbreviated by $\varepsilon(e) = \{e\}$ for $e \in C$, and an ordinal valued length of computation function $|.|$ such that $|e,a|\downarrow \iff \{e\}(a)\downarrow$, for which the following properties hold.

(1) C contains a copy of ω and Θ contains copies of zero, successor, predecessor on ω.

(2) Θ contains the projection functions, the operations of A and the relations of A in the form of definition-by-cases functions.

(3) Θ is closed under composition, the permuting of arguments, and the addition of dummy arguments. And, in particular,

(4) Θ contains *universal functions* ${}^nU$ such that for $e \in C, a \in A^n$

$${}^nU(e,a) \simeq \{e\}(a).$$

(5) Θ enjoys this *iteration property*: for each n,m there is a map $S^n_m \in \Theta$ such that for $e \in C$, $a \in C^n$, $b \in A^m$

$$\{S^n_m(e,a)\}(b) \simeq \{e\}(a,b).$$

Moreover, it is required that certain uniformity hypotheses are satisfied and that the length function respect the efficiency of the functions mentioned in the definition; for full details see [23] or J.E. Fenstad's monograph [6] from which this axiomatisation is taken.

The coding of programs, mentioned in the previous section, extends to a coding of the functions they compute: choose $C=\omega$ and $\{e\}$ to be the function computed by fapC e, if $e \in \Omega_C$, or to be the everywhere undefined function otherwise. Define length of computation $|e,a,x|$ to be the number of steps in computation $\{e\}(a,x)$. To be faithful to the definition of a computation theory, the code set C is adjoined to A to make the structure A_ω whose domain is $A \dot{\cup} \omega$ and whose operations and relations are those of A together with zero, successor, predecessor, and equality on ω. It turns out that the fapC-computable functions can be identified with FAP(A_ω).

2.1 THEOREM *The fapC-computable functions over* A *constitute a computation theory over* A *with code set* $C = \omega$ *if, and only if, term evaluation* ^{n}TE *is fapC-computable over* A *uniformly in* n.

Now term evaluation is always fapCS-computable and so on repeating the coding constructions with Ω_{CS} it follows that

2.2 THEOREM *The fapCS-computable functions over* A *constitute a computation theory over* A *with code set* $C = \omega$.

The next step in [23] was to define a computation theory Θ over A with code set C to be *minimal* if $\hat{\Theta}$ is contained in any other computation theory Φ over A with code set C. And then to prove

2.3 THEOREM *The fapCS-computable functions over* A *constitute the minimal computation theory over* A *with code set* $C = \omega$.

2.4 THEOREM *There exists a structure* A *where the following inclusions are strict*

$$\begin{tikzcd} & \mathrm{FAPC(A)} \arrow[dr] & \\ \mathrm{FAP(A)} \arrow[ur] \arrow[dr] & & \mathrm{FAPCS(A)} \\ & \mathrm{FAPS(A)} \arrow[ur] & \end{tikzcd}$$

Theorem 2.3 coupled to Theorem 2.1, and Theorem 2.4 are the point of departure for Section 5. Two other basic facts are needed later on. The first comes from [23], the second, a corollary, we leave as an exercise for the reader.

2.5 THEOREM *The relation* ${}^{n,m}\mathrm{STEP} \subset C \times A^n \times \omega^m \times \omega$ *defined by*

$${}^{n,m}\mathrm{STEP}(e,a,x,k) \equiv |e,a,x| \leq k$$

is fapCS-computable uniformly in n,m.

2.6 THEOREM $S \subset A^n \times \omega^m$ *is fapCS-computable if, and only if,* S *and* $\neg$ S *are fapCS-semicomputable.*

In [22] it was shown that the functions *inductively definable* over A, in the sense of Platek, are precisely the fapS-computable functions. To establish a wider context for the algebraic study of computation, J. Moldestad and I have attempted to systematically classify, in terms of the fap formalism, the many disparate approaches to defining computability in an abstract setting. Thus in [24] it is shown that Normann's *set recursion* is equivalent to fapCS-computability, that a natural generalisation of Herbrand-Gödel-Kleene *equational definability* is equivalent to fapS-computability, as is computability by *flowcharts with* LISP-*like recursive procedures*. See [24] for a complete survey and a discussion of a *Generalised Church-Turing Thesis* nominating the fapCS-computable functions as *the* class of functions effectively calculable by finite, deterministic algorithms in Algebra.

3. ALGEBRAIC INFLUENCES ON FAP-COMPUTATION

3.1 LOCALITY OF COMPUTATION LEMMA *Let* $\alpha:A^n \times \omega^m \to A/\omega$ *be a fapCS and* $(a,x) \in A^n \times \omega^m$. *Then each state of the computation* $\alpha(a,x)$ *lies within* ω *and the subalgebra* $<a>$ *of* A *generated by* $a \in A^n$. *In particular, if* $\alpha:A^n \times \omega^m \to A$ *and* $\alpha(a,x){\downarrow}$ *then* $\alpha(a,x) \in <a>$.

PROOF. Let $D_i(\alpha,a,x)$ be a state description in the unfolding of $\alpha(a,x)$. We must prove by induction on i that any algebraic element of $D_i(\alpha,a,x)$ lies in $<a>$. This is immediate for the basis $i = 1$ by convention. The induction step from $D_i(\alpha,a,x)$ to $D_{i+1}(\alpha,a,x)$ depends upon the type of instruction numbered n_i in $D_i(\alpha,a,x)$. There are 15 cases. The conditional instructions do not alter algebraic registers, nor do the operational instructions for counting. And the algebraic operational instructions either relocate elements of $D_i(\alpha,a,x)$, assumed in $<a>$, or apply a basic operation of A to them to create possibly new elements of $<a>$. Q.E.D.

Let F be a field. *The function* $f(a) = \sqrt{a}$ *which picks out some square root of* a, *if it exists, is not in general fapCS-computable.* Take $F = \mathbb{R}$: if f were fapCS-computable then $f(2) = \pm\sqrt{2}$ would lie in the subfield $<2> = \mathbb{Q}$ which is not the case.

Let R be an Euclidean domain with degree function $\partial:R \to \omega$. *The algorithm which computes for* $a \neq 0$, $b \in R$ *elements* q,r *with* $b = qa + r$ *and either* $r = 0$ *or* $\partial(r) < \partial(a)$ *is not in general fapCS-computable over* (R,∂). Take $R = \mathbb{Z}[X]$ with ∂ the usual polynomial degree. Let $a = X^2$, $b = X^7$ so that necessarily $q = X^5$, $r = 0$. But $X^5 \notin <X^7,X^2>$ since this subring involves only powers of the form X^{2n+7m} for $n,m \in \omega$.

In addition to treating search mechanisms with caution when formalising algorithms, one must also be prepared to sometimes dispense with pairing and unpairing functions:

A is *locally* n-*finite* if every n-generator subalgebra of A is finite. A is *locally finite* if it is locally n-finite for each n. If A is locally n-finite then the number of algebraic values for all the fapCS-computable functions at argument $a \in A^n$ is finite being bounded by the order function ${}^n\mathrm{ord}(a) = |<a>|$. In [9], E.S.

Golod shows that for each n there is a group G which is locally n-finite but not locally (n+1)-finite. And for these groups the theory of fapCS-computable functions of $\leq n$ arguments is considerably removed from that of functions of $> n$ arguments.

The reinstatement of search and pairing (*of a local character*) is the subject of Section 4, but fapCS-computation is better thought of as "sensitive" rather than "weak":

3.2 THEOREM *Let* F *be a field. Then the membership relation for finitely generated ideals of* $F[X_1,\dots,X_n]$, *defined*

$${}^{n,k}M(q,p_1,\dots,p_k) \equiv q \in (p_1,\dots,p_k)$$

is fapCS-computable over F *uniformly in* k,n.

PROOF. We describe the algorithm informally for (any) fixed n,k and sketch the reasons why it stays within the realm of fapCS-computation over F, freely referencing principles which belong to the next section. The algorithm refers to the data types F, $F[X_1,\dots,X_n]$, the ring M(s,t,F) of $s \times t$ matrices over F, ω, and a second polynomial ring $F[t_{ij}: 1\leq i\leq k, 1\leq j\leq \ell]$, but in fact operates in those parts defined by the subfield L of F generated by the coefficients of the input polynomials: $L[X_1,\dots,X_n]$, M(s,t,L) and so on. These localisations, and what the algorithm requires of them, can be fapCS-computably simulated over F, ω from the input coefficients; this claim we ask the reader to check possibly guided by the principles of Local Enumeration, Search and Pairing from Section 4.

From Satz 2 of G. Hermann [14] one can obtain this fact: *Consider equations of the form*

$$r_1p_1+\dots+r_kp_k = q \qquad (*)$$

where $q,p_1,\dots,p_k$ *are given and* $r_1,\dots,r_k$ *are to be found. There exists a primitive recursive function* $f: \omega^3 \to \omega$ *such that if* (*) *has a solution in* $F[X_1,\dots,X_n]$ *then it does so with* $\deg(r_i) \leq f(a,b,n)$ *where* $a = \deg(q)$ *and* $b = \max\{\deg(p_i): 1\leq i\leq k\}$. To decide $q \in (p_1,\dots,p_k)$ is to decide whether or not (*) has a solution and this is done by setting up a system of linear equations over F.

Construct formal polynomials $r_1,\dots,r_k$ of degree $d = f(a,b,n)$ with coefficients treated as indeterminates over F,

$$r_i = \sum_{|j|\leq d} t_{ij}X^j$$

where $j = (j_1,\dots,j_n)$, $|j| = j_1+\dots+j_n$ and $X^j = (X_1^{j_1}\dots X_n^{j_n})$. Substituting these into equation(*) produces a polynomial identity whose LHS has degree $\leq f(a,b,n) + b$ and whose RHS has degree $= a$. Comparing coefficients leads to a set of linear equations in t_{ij} over F. Thus $q \in (p_1,\dots,p_k)$ iff this set of equations has a solution in F; the latter point is covered by this lemma:

3.3 LEMMA *The relation* ${}^{m,n}R \subset M(m,n,F) \times F^m$ *defined*

$${}^{m,n}R(A,b) \equiv (\exists x\in F^n)(Ax=b)$$

is fapCS-computable over F *uniformly in* n,m.

PROOF. First consider the rank function ${}^{m,n}r: M(m,n,F) \to \omega$ defined ${}^{m,n}r(A)$ = rank of matrix A. This is fapCS-computable over F uniformly in m,n: calculate $r_0 \in \omega$ such that A has at least one non-singular $r_0 \times r_0$ minor and, for $s > r_0$, every $s \times s$ minor of A is singular. Then ${}^{m,n}r(A) = r_0$ and ${}^{m,n}r$ is fapCS-computable (by Local Search 4.4 and the fact that determinants are polynomials over the prime subfield of F!).

Now the lemma follows from a well known theorem of Linear Algebra: given $A \in M(m,n,F)$ and $b \in F^m$, let $[A,b]$ be A augmented by b as an (n+1)-th column. Then ${}^{m,n}R(A,b)$ iff ${}^{m,n}r(A) = {}^{m,n+1}r([A,b])$. Q.E.D.

What is striking about Hermann's bound is that it holds for *all* fields. To properly exploit the constructive implications of this uniformity one must dispense with traditional hypotheses, such as that F be computable, designed to make the problem ${}^{n,k}M$ constructively well posed, but irrelevant to its algorithmic solution which is field-theoretic in the fullest, *abstract*, sense of the term. And it is with precisely this sort of example in mind that we advocate the study of effective computability in a completely abstract algebraic setting.

3.4 INVARIANCE THEOREM *Let* A *and* B *be relational systems and* $\phi:A \to B$ *a relational homomorphism which is injective. Let* α *be a fapCS over their signature. Then*

$$\alpha(\phi a,x) \simeq \phi\alpha(a,x) \qquad \text{if } \alpha:A^n \times \omega^m \to A;$$
$$\alpha(\phi a,x) \simeq \alpha(a,x) \qquad \text{if } \alpha:A^n \times \omega^m \to \omega.$$

PROOF. This is best proved via a stronger, but technical, fact about state descriptions of $\alpha(a,x)$ and $\alpha(\phi a,x)$: if $D_i(\alpha,a,x) = (m_i,a_{ij},x_{ij},(z_j,a^i_{jk}))$ then $D_i(\alpha,\phi a,x) = (m_i,\phi a_{ij},x_{ij},(z_j,\phi a^i_{jk}))$. The argument is by induction on i. The basis i=1 is true by convention. Assume the identity true at step i and compare how $D_{i+1}(\alpha,a,x)$ and $D_{i+1}(\alpha,\phi a,x)$ arise from $D_i(\alpha,a,x)$ and $D_i(\alpha,\phi a,x)$, respectively; this depends on the 15 types of instruction numbered by m_i at stage i.

For example, let m_i name <u>if</u> $R(r_{\lambda_1},\ldots,r_{\lambda_k})$ <u>then</u> u <u>else</u> v. In both cases the new state descriptions differ from those at stage i only in their instruction numbers say m_{i+1} and n_{i+1}. Thanks to the induction hypothesis, the transitions are determined by these formulae

$$m_{i+1} = \begin{cases} u \text{ if } R(a_{i\lambda_1},\ldots,a_{i\lambda_k}); \\ v \text{ otherwise;} \end{cases} \qquad n_{i+1} = \begin{cases} u \text{ if } R(\phi a_{i\lambda_1},\ldots,\phi a_{i\lambda_k}); \\ v \text{ otherwise.} \end{cases}$$

Since ϕ is a relational homomorphism $m_{i+1} = n_{i+1}$.

We leave to the reader the task of checking the other cases noting only that it is the algebraic conditional with equality which requires ϕ to be a monomorphism. Q.E.D.

3.5 COROLLARY *Each fapCS-semicomputable set* $S \subset A^n$ *is invariant under the action of the automorphism group* Aut(A) *of* A.

PROOF. Let α be a fapCS such that $S = \mathrm{dom}(\alpha)$ and let $\phi \in \mathrm{Aut}(A)$. For $a \in A^n$,

$$\begin{aligned}\phi(a) \in S &\Longleftrightarrow \alpha(\phi a)\downarrow \\ &\Longleftrightarrow \phi\alpha(a)\downarrow \quad \text{by Theorem 3.4;} \\ &\Longleftrightarrow \alpha(a)\downarrow \quad \text{since } \phi \text{ is total.}\end{aligned}$$

Thus $\phi(S) = S$. Q.E.D.

For example, fapCS-semicomputable subgroups must be normal. And since complex conjugation $z \mapsto \bar{z}$ is an automorphism of the field $\mathbb{C}$, the set $\{i\}$ where $i = \sqrt{-1}$ is not fapCS-semicomputable.

3.6 COROLLARY *The group of all fapCS-computable automorphisms* $\mathrm{Aut}_{\mathrm{fapCS}}(A)$ *is a subgroup of the centre of* $\mathrm{Aut}(A)$. *Thus* $\mathrm{Aut}_{\mathrm{fapCS}}(A) \lhd \mathrm{Aut}(A)$.

If G is a group with trivial centre then Aut(G) has trivial centre, see Kurosh [16] p. 89. Therefore, *if group* G *has trivial centre, as is the case when* G *is non-abelian and simple or when* G *is a free group of rank* > 1, *then* $\mathrm{Aut}_{\mathrm{fapCS}}(G)$ *is trivial.* On the other hand, if G is a torsion-free divisible abelian group then Aut(G) contains the infinite family $\exp_n(g) = g^n$ for $0 \neq n \in \omega$.

4. LOCAL FAPCS-ENUMERATION, SEARCH AND PAIRING

In fapCS-computation, algebraic enumerations assume a strictly local character: *given* $a_1,\ldots,a_n \in A$ one can always fapCS-computably list the subalgebra $\langle a_1,\ldots,a_n\rangle$. And from this comes *local* fapCS-computable search and *local* fapCS-computable pairing. Here we state results which formulate these mechanisms precisely, leaving their proofs to the reader as straightforward, though instructive, exercises in manipulating the term evaluation functions by means of recursive calculations on ω. The theorems are quite fundamental, however. They strongly suggest that most results in Degree Theory and Axiomatic Complexity Theory on ω can be reproved *in a local form* for fapCS-computation, and that, in particular, Classical Recursion Theory can be reconstructed for FAPCS(A) provided A is finitely generated *by elements named as constants in its signature.*

(A study of the regularities and singularities involved in lifting the principal facts of Degree Theory and Axiomatic Complexity Theory has been undertaken by V. Stoltenberg-Hansen and myself and will appear in due course.)

4.1 LOCAL ENUMERATION THEOREM *There is a family of functions* ${}^nL: A^n \times \omega \to A$ *fapCS-computable uniformly in* n *such that for each* $a \in A^n$, ${}^nL(a): \omega \to \langle a\rangle$ *is a surjection; moreover* ${}^nL$ *can be so chosen as to make* ${}^nL(a)$ *a bijection* $\omega \to \langle a\rangle$, *if* $\langle a\rangle$ *is infinite, or a bijection* $\{0,\ldots,m-1\} \to \langle a\rangle$, if $\langle a\rangle$ *is finite of order* m.

Two corollaries of this enumeration facility which we use later on are

4.2 LEMMA *The order function* ${}^n\mathrm{ord}: A^n \to \omega$ *defined* ${}^n\mathrm{ord}(a) = |\langle a\rangle|$ *is fapCS-computable uniformly in* n.

4.3 LEMMA *The membership relation for finitely generated subalgebras of* A, *defined* ${}^nM(b,a) \equiv b \in \langle a\rangle$ *for* $b \in A$ *and* $a \in A^n$, *is fapCS-semicomputable uniformly in* n.

4.4 LOCAL SEARCH THEOREM *There is a family of functions* ${}^{n,m,k}v: C \times A^n \times \omega^m \to A^k$ *fapCS-computable uniformly in* n,m,k *such that if* $S \subset A^n \times \omega^m \times A^k$ *is fapCS-semicomputable by fapCS with code* $e \in C$ *and there is* $y \in \langle a \rangle^k$ *such that* $S(a,x,y)$ *then* ${}^{n,m,k}v(e,a,x)\downarrow$ *and* $S(a,x,{}^{n,m,k}v(e,a,x))$.

4.5 LOCAL PAIRING THEOREM *There is a family of functions* ${}^{n,m}L_*: A^n \times \omega \to A^m$ *fapCS-computable uniformly in* n,m *such that for each* $a \in A^n$, ${}^{n,m}L_*(a): \omega \to \langle a \rangle^m$ *is a surjection; moreover* ${}^{n,m}L_*$ *can be so chosen as to make a listing of* $\langle a \rangle^m$ *without repetitions.*

5. COUNTING AND STACKING: VARIETIES AND LOCAL FINITENESS

While the four kinds of fap-computable functions on algebra A are distinct (Theorem 2.4), the typical situation in Algebra is

$$FAP(A) \hookrightarrow FAPS(A) \hookrightarrow FAPC(A) = FAPCS(A)$$

Defining algebra A to be *regular* if A has uniform fapC-computable term evaluation, with Theorems 2.1 and 2.4 in mind, we substantiate this claim with a sufficient condition for A to be regular based on the proofs that groups and rings are regular. Notice regularity is an isomorphism invariant (Theorem 3.4).

First, we introduce *term width* which is used to measure the number of algebra registers required to fapC-evaluate a term on any input; c.f. Friedman [7], pp. 376-377.

An n-*ary syntactic development of width* m is a sequence $T_1,\ldots,T_\ell$ such that

(i) each T_1 is a list of m terms from $T[X_1,\ldots,X_n]$ $1 \le i \le \ell$;

(ii) T_1 contains only indeterminates

(iii) for $1 \le i \le \ell - 1$, either T_{i+1} arises from T_i by applying some k-ary operation symbol σ to some of the terms $t_1,\ldots,t_k$ in T_i and replacing one of the terms of T_i by $\sigma(t_1,\ldots,t_k)$ or,

(iv) T_{i+1} arises from T_i by replacing one of the terms of T_i by an indeterminate or by another of its terms;

(v) T_{i+1} differs from T_i in at most one of its terms $1 \le i \le \ell$.

A term $t(X_1,\ldots,X_n)$ is of *width at most* m if it belongs to some n-ary syntactic development of width m. Given a code for a term of width m and any $a \in A^n$ one can recursively calculate a code for some development to construct its values at each stage in m algebraic registers and output its value t(a).

Next, we adapt the notion of a *varietal normal form* as in H. Lausch & W. Nöbauer [19], p. 23.

A *recursive and complete set of term representatives of width* m(n) for an algebra A is a family $J = \{J_n : n \in \omega\}$ with $J_n \subset T[X_1,\ldots,X_n]$ such that

(i) the terms of J_n are of width at most m(n);

(ii) for each $t \in T[X_1,\ldots,X_n]$ one can recursively calculate $t' \in J_n$ for which $t \equiv_A t'$;

(iii) m is recursive.

5.1 THEOREM *Let* A *have a recursive and complete set of term representatives of width bounded by* m(n). *Let* M *be the largest arity of the operations of* A. *Then there is a fapC involving* (M+1). m(n) *algebraic work registers which computes term evaluation. In particular,* A *and* all its homomorphic images *are regular.*

The proof of this is routine: use Friedman's Lemma 1.6.2 and straightforward properties of homomorphisms. Theorem 5.1 is designed with those algebras which make up categories possessing free objects of all finite ranks in mind; for example, varieties and quasivarieties (among the first-order axiomatisable classes). Not only is Algebra replete with such categories but, invariably, the sets of *normal forms* constructed for the syntactic copies of their free objects are recursive and are of bounded width: from Lausch & Nöbauer [19] and P. Hall [11] we can "read off"

5.2 THEOREM *Let* A *be an algebra belonging to one of these varieties: semigroups, groups, associative rings with or without unity, Lie rings, semilattices, lattices, boolean algebras. Then* FAPC(A) = FAPCS(A).

Of course, the hypothesis of recursive sets of representatives for a variety is much weaker than that of recursive normal forms as the latter entails the free algebras of a variety have a decidable word problem. Notice, too, that fields do *not* form a variety, but recursive and complete sets of term representatives are easily made from the construction of rational functions, $\mathbb{Q}(X_1,\ldots,X_n)$ or $\mathbb{Z}_p(X_1,\ldots,X_n)$, from polynomials, $\mathbb{Z}[X_1,\ldots,X_n]$ or $\mathbb{Z}_p[X_1,\ldots,X_n]$.

Keeping within the regular algebras A, we now digress to collapse FAPCS(A) to FAP(A) by counting inside A.

An algebra A is said to have *fap counting on an element* $a \in A$ if there are functions $S,P:A \to A$ such that (i) $\{S^n(a):n\in\omega\}$ is infinite, (ii) $PS^n(a) = S^{n-1}(a)$ for each n and (iii) S,P are fap computable over (A,a).

5.3 THEOREM *Let* A *be regular. Then* A *has fap-counting on an element* $a \in A$ *if, and only if,* A *is not locally* 1-*finite and* FAP(A,a) = FAPCS(A,a).

PROOF. Assuming A to have fap-counting on $a \in A$, the Locality Lemma 3.1 shows A is not locally 1-finite. To simulate a fapC over (A,a) by a fap one rewrites α replacing each counting variable c in α by a new algebra variable u and then replacing each instruction of the left hand column by the corresponding fap instructions abbreviated in the right hand column:

c: = 0	u: = a
c: = c'+1	u: = S(u')
c: = c'$\dot{-}$1	u: = P(u')
<u>if</u> c=c' <u>then</u> * <u>else</u> **	<u>if</u> u=u' <u>then</u> * <u>else</u> **

The converse implication is an exercise in the use of the Local Enumeration Theorem 4.1.   Q.E.D.

5.4 EXAMPLE *A group* G *has fap-counting on an element if, and only if,* G *is not periodic.* Since periodicity and local 1-finiteness coincide in groups one implication follows from Theorem 5.3. Conversely, if G is non-periodic with $g \in G$ of finite order then define $S(x) = gx$ and $P(x) = g^{-1}x$ which are fap-computable over (G,g).

It is not enough to assume G is not locally finite to obtain counting on an element: take any of Golod's groups [9] which are periodic but not locally finite. This is not the case for fields however:

5.5 EXAMPLE *A field* F *has fap counting on an element if, and only if,* F *is of characteristic* 0 *or is of prime characteristic but not algebraic over its prime subfield.* Let F have fap-counting on $a \in F$ and assume F has characteristic p; denote the prime subfield of F by $\mathbb{Z}_p$. The subfield $\langle a \rangle = \mathbb{Z}_p(a)$ is infinite so cannot be algebraic over $\mathbb{Z}_p$ as otherwise $\mathbb{Z}_p(a)$ would have order p^n where n is the degree of a over $\mathbb{Z}_p$. The converse is obvious in case F has characteristic 0 and if $t \in F$ is transcendental over $\mathbb{Z}_p$ then F as a multiplicative group is not periodic since t is of infinite order and we can use Example 5.4. It is easy to check F is locally finite if, and only if, it has prime characteristic and is algebraic over its prime subfield.

We now turn to the study of exclusively algebraic computation: the class FAPS(A). The reader might care to keep in mind the equivalences of fapS-computability with inductive definability, equational definability and recursive procedures. For example, the facts which follow have a bearing on the debate about inductive definability as *the* generalisation of Recursion Theory to an abstract setting, see J.E. Fenstad [5,6].

5.6 THEOREM *If* A *is locally* n*-finite then the halting problem for* ^{n}FAPS(A) *is fapCS-decidable: the relation* ${}^nH(e,a)$ *iff* $\{e\}(a)\downarrow$ *is fapCS-computable on* $\Omega_S \times A^n$.

PROOF. Let R(e), I(e), and M(e) recursively calculate the number of algebraic register variables, instructions, and markers for stack blocks appearing in the fapS coded by e. A state description in a fapS computation is a list of the form

$$(k;a_1,\ldots,a_m;\ (z_1;a_{11},\ldots,a_{1m}),\ldots,(z_s;a_{s1},\ldots,a_{sm}))$$

and if this belongs to a computation by e then $k \le I(e)$, $m = R(e)$ and each $z_i \le M(e)$.

If in unfolding a computation $\{e\}(a)$ *there are two identical ordinary algebraic state descriptions or two identical stack state descriptions then* $\{e\}(a)\uparrow$. This is clear for if either of two identical states D_i, D_j arise then the program must regenerate after D_j exactly those states intermediate between D_i and D_j to produce an infinite, but periodic, set of state descriptions. When A is locally n-finite we can bound the number of distinct state descriptions in a convergent computation $\{e\}(a)$:

There are at most $B_0(e,a) = (I(e)+1)({}^n\text{ord}(a)+1)^{R(e)}$ different states for the ordinary algebraic registers and instructions. Due to the fact the stack mechanism copies ordinary algebraic state descriptions, it is easy to calculate that there can be at most

$$B_S(e,a) = \sum_{s=0}^{B_0(e,a)} \frac{(M(e).B_0(e,a))!}{(M(e).B_0(e,a)-s)!}$$

distinct stack states. Therefore, the total number of state descriptions available for $\{e\}(a)\downarrow$, and so the computation's run-time, is fapCS-computably bounded by (the total function) $B(e,a) = B_0(e,a).B_S(e,a)$.

Claim: For $e \in \Omega_S$, $a \in A^n$, ${}^nH(e,a) \Longleftrightarrow {}^nSTEP(e,a,B(e,a))$. Obviously, if ${}^nSTEP(e,a,B(e,a))$ is true then $\{e\}(a)\downarrow$. Conversely, if $\{e\}(a)\downarrow$ in ℓ steps and $\ell > B(e,a)$ then some duplication of state descriptions must have appeared and so $\{e\}(a)\uparrow$. Thus $\ell \le B(e,a)$. Q.E.D

5.7 COROLLARY *If* A *is locally finite then the halting problem for fapS computations is fapCS-decidable.*

An algebra A is said to be *fapCS formally valued* when there is a total function $v:A \to \omega$ which is a fapCS-computable surjection.

5.8 PROPOSITION *Let* A *be* 1-*finite and fapCS formally valued by* v. *Then the set* $K_v = \{a \in A: v(a) \in \Omega_S \;\&\; \{v(a)\}(a)\downarrow\}$ *is fapCS-computable but not fapS-computable.*

PROOF. A 1-finite implies ${}^1H(e,a)$ is fapCS-decidable on $\Omega_S \times A$ by Theorem 5.6. Thus, K_v is fapCS-decidable as $a \in K_v \Longleftrightarrow v(a) \in \Omega_S \;\&\; {}^1H(v(a),a) = 0$, and v is total. Assume for a contradiction that K_v is fapS-computable. Then $\neg K_v$ is fapS-semi-computable and there exists a fapS α such that $\mathrm{dom}(\alpha) = \neg K_v$. Choose $b \in A$ so that $v(b)$ codes α.

Then $b \in \neg K_v \Longleftrightarrow \alpha(b)$
$\Longleftrightarrow \{v(b)\}(b)\downarrow$
$\Longleftrightarrow b \in K_v$.

Thus there is no such α and K_v is not fapS-computable. Q.E.D.

For illustrations we look for valuations of regular algebras A which are 1-finite where $FAPS(A) \subsetneq FAPC(A) = FAPCS(A)$.

If A contains 1-generator subalgebras of every finite order then ${}^1ord:A \to \omega$ is a fapCS formal valuation *provided* A is 1-finite. For example, take the locally finite group $\mathbb{Z}_\infty$ consisting of all the complex roots of unity. More generally, if algebra A is 1-finite and $\pi(A) = im({}^1ord)$ contains a recursive set S then choose a recursive bijection $f:S \to \omega$ and define

$$v(a) = \begin{cases} f({}^1ord(a)) & \text{if } {}^1ord(a) \in S, \\ {}^1ord(a) & \text{otherwise.} \end{cases}$$

For an example among groups we must look for a periodic group of infinite exponent. Let p be a prime and let $\mathbb{Z}p^\infty$ be the locally finite abelian group of all complex roots of unity which are of order some power of p: for these groups $\pi(\mathbb{Z}p^\infty) = \{p^n : n \in \omega\}$, a recursive set.

Turning to fields, let F be a locally finite field of characteristic p and observe that ${}^1ord(a) = p^{d(a)}$ where $d:F \to \omega$

calculates the degree of $a \in F$, that is $d(a) = \dim[\mathbb{Z}_p(a) : \mathbb{Z}_p]$. If F contains elements of every degree belonging to a recursive set then d is a fapCS formal valuation. Such an F can be chosen by taking the splitting field of the polynomials $\{X^{p^n} - X \in \mathbb{Z}_p[X] : n \in \omega\}$ but the best example is the algebraic closure K of $\mathbb{Z}_p$. In summary,

5.9 THEOREM *Over the groups* $A = \mathbb{Z}_\infty$, $\mathbb{Z}p^\infty$ *and the field* $A = K$,

$$\mathrm{FAPS}(A) \subsetneqq \mathrm{FAPC}(A) = \mathrm{FAPCS}(A).$$

In conclusion, we mention in connection with Corollary 5.7, that fapS computability and fapS semicomputability may actually coincide.

An algebra A is said to be *uniformly locally finite*, *ulf* for short, if there is a function $\lambda : \omega \to \omega$ such that for any $a_1, \ldots, a_n \in A$, ${}^n\mathrm{ord}(a_1, \ldots, a_n) < \lambda(n)$.

If A is a ulf algebra then we can replace $B(e,a)$ in the argument of Theorem 5.6 by a function $B'(e,n)$. One consequence of this is that for *fixed* e, ${}^nH(e,a)$ is fapS-decidable:

5.10 THEOREM *Let* A *be uniformly locally finite. Then* $S \subset A^n$ *is fapS-semicomputable if, and only if, S is fapS-computable.*

For groups, *Kostrikin's Theorem* implies any locally finite group of prime exponent is uniformly locally finite. More generally, calling a class of algebras k uniformly locally finite if there is $\lambda : \omega \to \omega$ such that for $A \in k$ and $a_1, \ldots, a_n \in A$, ${}^n\mathrm{ord}(a_1, \ldots, a_n) < \lambda(n)$, it is a corollary of the *Ryll-Nardzewski Theorem* that if k is an ω-categorical first-order axiomatisable class then k is a uniformly locally finite class. See also *Mal'cev's Theorem* [21] p. 285 that a variety consisting of locally finite algebras is uniformly locally finite.

6. TOPOLOGICAL ALGEBRAS

A is a *topological algebra* if its domain is a non-trivial topological space on which its operations are continuous.

6.1 THEOREM *Let* A *be a Hausdorff topological algebra in the quasivariety* V. *If* A *contains a* V*-free* n*-generator subalgebra then for any fapCS-decidable relation* $S \subset A^n \times \omega^m$ *and* $x \in \omega^m$ *the sets* $\{a \in A^n : S(a,x)\}$ *and* $\{a \in A^n : \neg S(a,x)\}$ *cannot both be dense in* A^n.

This fact we obtain as a useful, palatable corollary of a more general, technical result about topological relational systems. Both theorems are suggested by remarks of Herman and Isard, in [13], concerning $\mathbb{R}$; we use them on fields, abelian groups and differential rings.

A relation $R \subset A^n$ is *continuous* at $a \in A^n$ if its characteristic function $R : A^n \to \{0,1\}$ is continuous at a between the product topology on A and the discrete topology on $\{0,1\}$.

Recalling the congruence relation $\equiv_A$ on $T[X_1, \ldots, X_n]$ from Section 1, a point $a \in A^n$ is said to be *transcendental* if for any terms $t, t' \in T[X_1, \ldots, X_n]$

$t \equiv_A t'$ if, and only if, $t(a) = t'(a)$ in A.

6.2 THEOREM *Let* A *be a relational structure which is a Hausdorff topological algebra, and let* $S \subset A^n \times \omega^m$ *be a fapCS-decidable relation. If* A *contains a transcendental point* $a \in A^n$ *on whose subalgebra* $\langle a \rangle$ *the basic relations of* A *are continuous then for any* $x \in \omega^m$ *there is an open subset of* A^n *containing* a *upon which* S *holds or fails accordingly as it holds or fails on* a.

To deduce Theorem 6.1, let V be a quasivariety and let $T_V[X_1,\dots,X_n] = T[X_1,\dots,X_n]/\equiv_V$ the V-free polynomial algebra of rank n. If $A \in V$ and $a \in A^n$ V-freely generates the V-free algebra $\langle a \rangle$ then $v_a : T_V[X_1,\dots,X_n] \to A$, defined $v_a[t] = t(a)$, is an embedding: $t(a) = t'(a)$ in A iff $[t] = [t']$ in $T_V[X_1,\dots,X_n]$. But $t \equiv_V t'$ implies $t \equiv_A t'$ and so a is transcendental. With this hypothesis of Theorem 6.2 satisfied, the remainder of the deduction of 6.1 is straightforward.

The Locality Lemma 3.1 prompts us to make this definition. A *syntactic state description* is a list of the form

$$(k,t_1(X),\dots,t_p(X),\ x_1,\dots,x_q;(z_1,t_{11}(X),\dots,t_{1p}(X)),\\ \dots,(z_s,t_{s1}(X),\dots,t_{sp}(X)))$$

where k and the z_i are instruction labels, $x_1,\dots,x_q \in \omega$, and what remains are terms in $X = (X_1,\dots,X_n)$. When we unfold a computation $\alpha(a,x)$ syntactically we obtain the i-th syntactic state description $T_i(\alpha,a,x) = (m_i,t_{ij},x_{ij},(z_{ij},t^i_{jk}))$ which is a mapping A^n to state descriptions.

PROOF OF THEOREM 6.2 Let α fapCS-decide S over A. Let $a \in A^n$ be transcendental and $x \in \omega^m$ be arbitrarily chosen and fixed. Since S is fapCS-decidable iff $\neg S$ is a fapCS-decidable, we assume, without loss of generality, that $S(a,x)$ is true and consider a computation $\alpha(a,x)$ of this fact. Let $\alpha(a,x)$ have length ℓ and syntactic state descriptions $T_i(\alpha,a,x)$ for $1 \le i \le \ell$ which we take as functions,

$$T_i(X) = (m_i,t_{ij}(X),\ x_{ij},(z_{ij},t^i_{jk}(X))),$$

of $X = (X_1,\dots,X_n)$ only. For $b \in A^n$ we denote the i-th state of the computation $\alpha(b,x)$ by

$$D_i(\alpha,b,x) = (n_i,b_{ij},y_{ij},(w_{ij},b^i_{jk})).$$

We prove there is a basic open set $B(a)$ containing a such that for all $b \in B(a)$ and each $1 \le i \le \ell$, $D_i(\alpha,b,x) = T_i(b)$. Obviously, this entails $\alpha(b,x) = \alpha(a,x)$ for $b \in B(a)$.

Claim. For each $1 \le i \le \ell$ there is a basic open set $B_i(a)$ containing a so that for $b \in B_i(a)$, $D_i(\alpha,b,x) = T_i(b)$.

On proving the claim by induction on i, we may take

$$B(a) = \cap_{i=1}^{\ell} B_i(a).$$

The basis i=1 is true for $B_1(a) = A^n$ by convention. Assume the claim true at stage i of the computation $\alpha(a,x)$: $B_i(a)$ is constructed and for any chosen $b \in B_i(a)$, $D_i(\alpha,b,x) = T_i(b)$. Consider the passage from $D_i(\alpha,b,x)$ to $D_{i+1}(\alpha,b,x)$ which depends on the nature of the instruction n_i. We give just 3 of the 15 cases:

Let $n_i = m_i$ be $r_\mu := \sigma(r_{\lambda 1},\dots,r_{\lambda k})$. In applying this instruction only algebra registers are changed, the transitions of $D_i(\alpha,b,x)$ to $D_{i+1}(\alpha,b,x)$, and $T_i(X)$ to $T_{i+1}(X)$, being determined by the formulae,

$$b_{i+1,j} = \begin{cases} \sigma(b_{i\lambda_1},\ldots,b_{i\lambda_k}) & \text{if } j = \mu \\ b_{ij} & \text{if } j \neq \mu \end{cases}$$

$$t_{i+1,j}(X) = \begin{cases} \sigma(t_{1\lambda_1}(X),\ldots,t_{i\lambda_k}(X)) & \text{if } j = \mu \\ t_{ij}(X) & \text{if } j \neq \mu. \end{cases}$$

Substituting $X = b$ and applying the induction hypothesis $b_{i\lambda_j} = t_{i\lambda_j}(b)$ we get $D_{i+1}(\alpha,b,x) = T_{i+1}(b)$ for *any* $b \in B_i(a)$. So we can set $B_{i+1}(a) = B_i(a)$.

The other operational instructions also make state transitions "independently" of a, but conditional instructions do not:

Let $n_i = m_i$ be <u>if</u> $R(r_{\lambda_1},\ldots,r_{\lambda_k})$ <u>then</u> u <u>else</u> v. By the induction hypothesis $b_{ij} = t_{ij}(b)$ so the state transitions are determined by

$$n_{i+1} = \begin{cases} u \text{ if } R(t_{i\lambda_1}(b),\ldots,t_{i\lambda_k}(b)) \\ v \text{ otherwise.} \end{cases}$$

$$m_{i+1} = \begin{cases} u \text{ if } R(t_{i\lambda_1}(a),\ldots,t_{i\lambda_k}(a)) \\ v \text{ otherwise.} \end{cases}$$

By the continuity hypothesis on relations, the map $R \circ p: A^n \to \{0,1\}$ is continuous at a where $p(X) = (t_{i\lambda_1}(X),\ldots,t_{i\lambda_k}(X))$. So there is a basic open set $V_i(a)$ containing a such that for any $b \in V_i(a) \cap B_i(a)$, $R \circ p(b) = R \circ p(a)$ and $n_{i+1} = m_{i+1}$. Thus take $B_{i+1}(a) = V_i(a) \cap B_i(a)$.

Let $n_i = m_i$ be <u>if</u> $r_\mu = r_\lambda$ <u>then</u> u <u>else</u> v. Again since $b_{ij} = t_{ij}(b)$ we have

$$n_{i+1} = \begin{cases} u \text{ if } t_{i\mu}(b) = t_{i\lambda}(b) \\ v \text{ otherwise.} \end{cases}$$

$$m_{i+1} = \begin{cases} u \text{ if } t_{i\mu}(a) = t_{i\lambda}(a) \\ v \text{ otherwise.} \end{cases}$$

Consider the two cases for m_{i+1}. If $t_{i\mu}(a) = t_{i\lambda}(a)$ then, since a is transcendental, $t_{i\mu} \equiv_A t_{i\lambda}$ and $t_{i\mu}(b) = t_{i\lambda}(b)$ for any $b \in A^n$, and the passage of $T_i(X)$ to $T_{i+1}(X)$ is independent of a. Thus, set $B_{i+1}(a) = B_i(a)$ on which $n_{i+1} = m_{i+1}$.

However, if $t_{i\mu}(a) \neq t_{i\lambda}(a)$ then $\{b \in A^n: t_{i\mu}(b) \neq t_{i\lambda}(b)\}$ is an open set containing a - because A is Hausdorff - and we can choose a neighbourhood $V_i(a)$ about a within it. For $b \in V_i(a) \cap B_i(a)$, $n_{i+1} = m_{i+1}$ so here we set $B_{i+1}(a) = V_i(a) \cap B_i(a)$. Q.E.D.

FIELDS. In the field of reals $\mathbb{R}$ any transcendental number is a transcendental element in our special sense. Therefore, by Theorem 6.2, sets such as the rationals $\mathbb{Q}$ and the algebraic numbers A which are dense and codense in $\mathbb{R}$ cannot be fapCS-decidable. These examples, cited for fap-decidability, are in Herman & Isard [13].

ABELIAN GROUPS Let T^n be the n dimensional torus group, the n-fold direct product of the circle group S^1.

6.3 EXAMPLE *The set $FO(T^n)$ of all elements of T^n of finite order, also known as the torsion subgroup of T^n, is fap-semicomputable but not fapCS-computable.*

PROOF It is known from Section 1 that $FO(T^n)$ is fap-semicomputable. To apply Theorem 6.1 we have to show T^n contains a 1-generator free-abelian subgroup and that $FO(T^n)$ is dense and codense in T^n. We begin by examining S^1.

Let $f:[0,1) \to S^1$ be the continuous parameterisation $f(t) = (\sin 2\pi t, \cos 2\pi t)$. It is easy to check $f(t)$ is of finite order iff t is a rational number. Thus S^1 has a free-abelian subgroup and, indeed, since the rationals are dense and codense $[0,1)$, $FO(S^1)$ is dense and codense in S^1 because the image of a dense subset of a space is dense in the image of a continuous map. Now observe that for any group G, $FO(G)^n = FO(G^n)$ and $IO(G)^n = IO(G^n)$ where $IO(G) = \neg FO(G)$, and also that in any topological space X, D dense in X entails D^n dense in X^n. Applying these facts to $FO(S^1)$ and $IO(S^1)$ we are done. Q.E.D.

6.4 EXAMPLE *The 1-generator subgroup membership relation M in T^n is fap-semicomputable but not fapCS-computable.*

PROOF. First we show $M = \{(g,t): g \in \langle t\rangle\}$ is dense in $T^n \times T^n$. Let $B(a,b)$ be a basic open set containing $(a,b) \in T^n \times T^n$ which we can take, without loss of generality, to be of the form $B_1(a) \times B_2(b)$ where $B_1(a)$, $B_2(b)$ are basic open sets about a,b respectively. Now the set of those $t \in T^n$ such that $\langle t\rangle$ is dense in T^n is itself dense in T^n, see J.F. Adams [1], p. 79. So we can choose $t \in B_2(a)$ so that $\langle t\rangle$ meets all neighbourhoods and in particular $B_1(b)$. This means there is $(g,t) \in B_1(a) \times B_2(b)$ such that $g \in \langle t\rangle$ and M is dense.

Consider $\neg M$. Let $B(a,b)$ be as before. From the argument of Example 6.3, we can choose an element $t \in B_2(b)$ of finite order and $g \in B_1(a)$ of infinite order and so a pair $(g,t) \in B(a,b)$ for which $g \notin \langle t\rangle$.

From the observations of 6.3, it is easy to show T^n contains a 2-generator free-abelian subgroup and complete the argument with Theorem 6.1. Q.E.D.

INTEGRATION Let $C^\omega(\mathbb{R},\mathbb{R})$ be the set of all analytic functions $\mathbb{R}\to\mathbb{R}$ which is a differential ring under pointwise addition and multiplication, and differentiation. Let E be the differential subring generated by e^x, $\sin x$, the polynomial functions $\mathbb{R}[X]$, and all their compositions, a subring of the so-called elementary functions. Let $I(f)$ be the integration relation in E, $I(f) \equiv (\exists g \in E)(Dg = f)$. For example, it is well known from a theorem of Liouville that $e^{-x^2} \notin I$, see G.H. Hardy's book [12].

6.5 EXAMPLE I *is a fapCS-semicomputable subset of E which is not* *fapCS-computable.*

PROOF. Equip $C^\omega(\mathbb{R},\mathbb{R})$ with the C^∞-topology prescribed thus: a sequence $f_n \to 0$, as $n \to 0$, iff for each k, the sequence $D^k f_n \to 0$, as $n \to 0$, in the topology of uniform convergence on compact subsets on $C^\omega(\mathbb{R},\mathbb{R})$; this means that for each k and each real $R > 0$, the sequence $\sup_{|x|<R} |D^k f_n(x)| \to 0$ in $\mathbb{R}$. With the C^∞-topology $C^\omega(\mathbb{R},\mathbb{R})$ is a topological differential ring (which is not the case with the usual topology of uniform convergence on compacts where D fails to be continuous). See M. Golubitsky & V. Guillemin [10], pp. 42-50. Consider the hypotheses of 6.2. By induction on term height it is straightforward to prove that (say) e^x is a transcendental point in E: this is omitted. That I is dense in E in the C^∞-topology follows from the fact that the polynomials $\mathbb{R}[X] < I < E$ and that the sequence of Taylor polynomials of an analytic function converge to the function in the topology of uniform convergence. That I is codense is more involved.

Let $f \in I$, we shall approximate f by the sequence of non-integrable functions $f_n(x) = f(x) + 1/n\ e^{-x^2/n}$. It is easy to see that the f_n are not integrable and that the approximation property follows from the claim that $1/n\ e^{-x^2/n} \to 0$ as $n \to \infty$ in the C^∞-topology.

On calculating the k-th derivative $D^k(e^{-x^2/n})$ we find it to be of the form $P_k(x,1/n)e^{x^2/n}$, where $P_k(x,1/n) = \Sigma_{i=1}^n a_i X^i/n^{f_i(k)}$ where $f_i(k) \geq [k/2]$ = largest natural number $\geq k/2$. Whence it is easy to check that $D^k(e^{-x^2/n}) \to 0$ in the topology of uniform convergence on compact sets. Q.E.D.

Two papers the reader might care to study, in the light of these notes, are A. Kreczmar [15] and E. Engeler [4], though these deal with a class of functions slightly smaller than FAP(A).

REFERENCES

1. J.F. ADAMS, *Lectures on Lie groups*, W.A. Benjamin, New York, 1969.

2. J.W. de BAKKER, *Mathematical theory of program correctness*, Prentice Hall International, London, 1980.

3. J.A. BERGSTRA, J. TIURYN & J.V. TUCKER, 'Correctness theories and program equivalence', *Mathematical Centre, Computer Science Department Research Report*, IW 119, Amsterdam, 1979.

4. E. ENGELER, 'Generalized galois theory and its application to complexity'. *ETH-Zürich, Computer Science Institute Report* 24, Zürich, 1978.

5. J.E. FENSTAD, 'On the foundation of general recursion theory: computations versus inductive definability' pp. 99-111 of J.E. FENSTAD, R.O. GANDY & G.E. SACKS (eds.) *Generalized recursion theory II*, North Holland, Amsterdam, 1978.

6. _______ , *Recursion theory: an axiomatic approach*, Springer-Verlag, Berlin, 1980.

7. H. FRIEDMAN, 'Algorithmic procedures, generalised Turing algorithms, and elementary recursion theory', pp. 316-389 of R.O. GANDY & C.E.M. YATES (eds.), *Logic Colloquim, '69*, North-Holland, Amsterdam, 1971.

8. J.A. GOGUEN, J.W. THATCHER & E.G. WAGNER, 'An initial algebra approach to the specification, correctness and implementation of abstract data types' pp. 80-149 of R.T. YEH (ed.) *Current trends in programming methodology IV. Data structuring*, Prentice-Hall, Engelwood Cliffs, New Jersey, 1978.

9. E.S. GOLOD, 'On nil-algebras and residually finite p-groups'. *American Math. Soc. Translations*, (2) 48 (1965) 103-106.

10. M. GOLUBITSKY and V. GUILLEMIN, *Stable mappings and their singularieies*, Springer-Verlag, New York, 1973.

11. P. HALL, 'Some word problems', *J. London Math. Soc.*, 33 (1958) 482-496.

12. G.H. HARDY, *The integration of functions of a single variable*, Second edition, Cambridge University Press, London, 1916.

13. G.T. HERMAN & S.D. ISARD, 'Computability over arbitrary fields', *J. London Math. Soc.* 2 (1970) 73-79.

14. G. HERMANN, 'Die Frage der endlich vielen Schritte in der Theorie der Polynomideale', *Mathematische Annalen* 95 (1926) 736-788.

15. A. KRECZMAR, 'Programmability in fields', *Fundamenta Informaticae* 1 (1977) 195-230.

16. A.G. KUROSH, *The theory of groups I*, Chelsea, New York, 1955.

17. ___________, *The theory of groups II*, Chelsea, New York, 1956.

18. ___________, *General algebra*, Chelsea, New York, 1963.

19. H. LAUSCH and W. NÖBAUER, *Algebra of polynomials*, North-Holland, Amsterdam, 1973.

20. M. MACHTEY & P. YOUNG, *An introduction to the general theory of algorithms*, North-Holland, New York, 1978.

21. A.I. MAL'CEV, *Algebraic systems*, Springer-Verlag, Berlin, 1973.

22. J. MOLDESTAD, V. STOLTENBERG-HANSEN & J.V. TUCKER, 'Finite algorithmic procedures and inductive definability', to appear in *Mathematica Scandinavica*.

23. ______ , 'Finite algorithmic procedures and computation theories' to appear in *Mathematica Scandinavica*.

24. J. MOLDESTAD & J.V. TUCKER, 'On the classification of computable functions in an abstract setting', in

preparation.

25. B.L. VAN DER WAERDEN, *Algebra I*, Ungar, New York, 1970.