INITIAL ALGEBRAIC SEMANTICS FOR NON CONTEXT-FREE LANGUAGES

H.Kaphengst and H.Reichel
VEB Robotron, Zentrum für
Forschung und Technik
DDR-801 Dresden,Leningrader Str. 15

Introduction

J.A.Goguen has introduced the concept of'Initial Algebraic Semantics' in his papers [1] and [2], and he has shown that this concept yields an elegant algebraic presentation and a simple mathematical foundation of Knuth's approach in 'Semantics of Context-free Languages',[3].

The basic idea is the fact that with every context-free grammar G there can be associated in a natural way a heterogeneous operator domain τ_G in such a way that the set of derivation trees of G forms an initial algebra in the class $M(\tau_G)$ of all τ_G-algebras. On that account a context-free programming language can be fully defined by giving a syntactical τ_G-algebra whose elements are strings over the set of terminals of G and by giving a semantical τ_G-algebra.

Both Goguen and Gladki & Dikovski [4] say that this logical simple model for defining syntax and semantics of a programming language is only applicable to context-free languages.

In the paper we will outline how the basic idea in the approach of Goguen and Knuth can be extended to any recursively enumerable language and that in this extension the usually in natural languages expressed context conditions of programming languages can be expressed in a formalized way and in a legible form. The algebraic basis for this extension is the existence of freely generated equational-partial heterogeneous algebras [5],[6].

Algebraic Basic Notions

In the following we will summarize the basic concepts of equational-partial heterogeneous algebras.

A heterogeneous operator domain is a quadruple $\tau=(B,\Omega,\delta,\sigma)$ consisting of a finite set B of sorts, of a finite set Ω of operators, of the 'input mapping' $\delta:\Omega \to B^*$(B^* denotes the set of strings over B, including the empty string) and the 'output mapping' $\sigma:\Omega \to B$.
A heterogeneous algebra $A=((A[b])_{b\in B},(A[\omega])_{\omega\in\Omega})$ is an ordered pair of

a family $(A[b])_{b\in B}$ of sets, and of a family $(A[\omega])_{\omega\in\Omega}$ of mappings $A[\omega]: A[\delta(\omega)] \longrightarrow A[\sigma(\omega)]$ with $A[\delta(\omega)] = A[b_1] \times A[b_2] \times \ldots \times A[b_n]$ for $\delta(\omega)=b_1b_2\ldots b_n\in B^*$.

The <u>set</u> $T(\tau)$ <u>of operator terms over</u> τ is given by:

(1) For every $w\in B^*$ and every integer $i\leq l(w)$ ($l(w)$ denotes the length of w) "^iw" is an operator term with $w=\delta(^iw)$ as input domain and $\sigma(^iw)=w(i)$ (the i-th componente of w) as output domain.

(2) For operator terms $t_1,\ldots,t_n$ with $\delta(t_1)=\delta(t_2)=\ldots=\delta(t_n)$ and for an operator $\omega\in\Omega$ with $\delta(\omega)=b_1b_2\ldots b_n$ and $b_j=\sigma(t_j)$ for $j=1,2,\ldots,n$ "$\omega(t_1,t_2,\ldots,t_n)$" is an operator term with $\delta(t_1)=\delta(t_2)=\ldots=\delta(t_n)=\delta(\omega(t_1,\ldots,t_n))$ as input domain and $\sigma(\omega(t_1,\ldots,t_n))=\sigma(\omega)$ as output domain.

An unordered pair $t_1\equiv t_2$ of operator terms with $\delta(t_1)=\delta(t_2)=w$ and $\sigma(t_1)=\sigma(t_2)$ is an <u>equation in</u> w <u>over</u> τ and a set $g=\{t_j\equiv r_j \mid j\in J\}$ is an <u>equation system in</u> w <u>over</u> τ if $\delta(t_j)=\delta(r_j)=w$ for every $j\in J$. With $\emptyset_w$ we denote for $w\in B^*$ the empty equation system in w over τ. $G(\tau)$ denotes the set of all equation systems over τ and $G_w(\tau)\subseteq G(\tau)$ denotes the subset of all equation systems in w over τ.

A <u>partial</u> τ-<u>interpretation</u> $P=((P[b])_{b\in B},(P[\omega])_{\omega\in\Omega})$ of a heterogeneous operator domain $\tau=(B,\Omega,\delta,\sigma)$ is given by a family $(P[b])_{b\in B}$ of sets and a family $(P[\omega])_{\omega\in\Omega}$ of partial mappings

$$P[\omega] : P[\delta(\omega)] \longrightarrow\!\!\!\!\circ\; P[\sigma(\omega)]$$

with $P[w]= P[b_1] \times \ldots \times P[b_n]$ for every $w=b_1\ldots b_n\in B^*$.

By induction we can define for every $t\in T(\tau)$ and every partial τ-interpretation P a partial mapping $P[t]: P[\delta(t)] \longrightarrow\!\!\!\!\circ\; P[\sigma(t)]$ in such a way that $P[t](p)$ is defined for $p\in P[\delta(t)]$ and for $t=\omega(t_1,\ldots,t_n)$ iff $P[t_j](p)$ for $j=1,\ldots,n$ and $P[\omega](P[t_1](p),\ldots,P[t_n](p))$ are defined.

For every partial τ-interpretation P and every equation system g in w over τ there can be defined a set

$$P[g] = \bigcap_{r_j\equiv s_j\in g} P[r_j\equiv s_j]$$

with $P[r\equiv s] = \{p\in P[\delta(r)] \mid P[r](p)$ and $P[s](p)$ are both defined and equal $\}$.

An <u>equational-partial operator domain</u>

$$\Theta = ((B,\Omega,\delta,\sigma),\pi:\Omega\longrightarrow \mathrm{Nat},\beta:\Omega\longrightarrow G(\tau))$$

is given by a heterogeneous operator domain τ, by a mapping $\pi:\Omega\longrightarrow$ Nat, where $\pi(\omega)$ is called the degree of partiality of ω, by

a mapping $\beta:\Omega \to G(\tau)$, where $\beta(\omega)$ is called the definition-condition of ω, such that:

(a) $\beta(\omega)$ is an equation system in $\delta(\omega)$ over τ for every $\omega \in \Omega$;

(b) $\pi(\omega) = 1$ if and only if $\beta(\omega) = \emptyset_{\delta(\omega)}$;

(c) $\pi(\omega) = 1 + \max\{\pi(\omega') \mid \omega' \in \Omega(\beta(\omega))\}$

($\Omega(g)$ denotes the set of operators that are used in g).

A θ-<u>interpretation</u> is a partial τ-interpretation such that for every $\omega \in \Omega$ and every $p \in P[\delta(\omega)]$ holds

$P[\omega](p)$ is defined if and only if $p \in P[\beta(\omega)]$.

An ordered pair (g_1,g_2) of equation systems in w over τ is called an <u>implication in</u> w <u>over</u> θ. Let P be a θ-interpretation for $\theta=(\tau,\pi,\beta)$ then we say that the implication (g_1,g_2) in $w \in B^*$ over θ holds in P if $P[g_1] \subseteq P[g_2]$.

The implications are the natural extensions of equations in universal algebra for the above defined kind of partial heterogeneous algebras. By $(\emptyset_w,g)$ we can express the usual equations in algebraic structures, because $P[\emptyset_w] = P[w]$ is valid for every $w \in B^*$ and every partial τ-interpretation P.

An ordered pair $\underline{S} = (\theta,\Gamma)$, where θ is an equational-partial operator domain and Γ is a set of implications over θ, will be called a <u>signature</u>. By $M(\underline{S})$ we denote the class of all θ-interpretations P such that every implication $(g_1,g_2) \in \Gamma$ holds in P. This kind of heterogeneous partial algebras will be called <u>equational-partial heterogeneous algebras</u> or "<u>equoids</u>". For $P \in M(\underline{S})$ we write also $\underline{S}$-model or $\underline{S}$-equoid.

Let P and M be $\underline{S}$-equoids. An $\underline{S}$-homomorphism $f:P \to M$ is given by a family $f=(f[b])_{b \in B}$ of mappings $f[b]: P[b] \to M[b]$, $b \in B$, such that the following condition (H) is satisfied for every $\omega \in \Omega$:

(H) If $p \in P[\beta(\omega)]$ then $f_{\delta(\omega)}(p)=(f[b_1](p_1),\ldots,f[b_n](p_n))$ for $\delta(\omega)=b_1b_2\ldots b_n \in B^*$ is an element of $M[\beta(\omega)]$ and

$$f_{\sigma(\omega)}(P[\omega](p)) = M[\omega](f_{\delta(\omega)}(p)) .$$

It can be shown that for every $\underline{S}$ in $M(\underline{S})$ freely generated equoids exist. The domains of the defining operations of freely generated equoids are in general not empty. Some of the best known examples of equoids are small categories.

A signature $\underline{S}_2$ is called an extension of $\underline{S}_1$, $\underline{S}_1 \subseteq \underline{S}_2$, if every sort, every operator and every implication of $\underline{S}_1$ is also contained in $\underline{S}_2$, and if the mappings $\delta_1,\sigma_1,\pi_1,\beta_1$ are the corresponding restrictions of

$\delta_2, \sigma_2, \pi_2, \beta_2$.

If $\underline{S}_1 \subseteq \underline{S}_2$ is a signature extension then we have a forgetful functor $U_{\underline{S}_2,\underline{S}_1}: M(\underline{S}_2) \rightarrow M(\underline{S}_1)$ which associates with every $\underline{S}_2$-equoid M the underlying $\underline{S}_1$-equoid $M\downarrow\underline{S}_1$ and with $f: M \rightarrow P$ the underlying $\underline{S}_1$-homomorphism $f\downarrow\underline{S}_1: M\downarrow\underline{S}_1 \rightarrow M\downarrow\underline{S}_1$.

It can be shown that for every signature extension $\underline{S}_1 \subseteq \underline{S}_2$ the forgetful functor has a left adjoint functor $F_{\underline{S}_1,\underline{S}_2}: M(\underline{S}_1) \rightarrow M(\underline{S}_2)$. This means that relatively free equoids exist. The free category of paths over a directed graph is a good example for such a relatively free equoid.

Let $\underline{S}_1 \subseteq \underline{S}_2$ be a signature extension and let M be an $\underline{S}_1$-equoid. The $\underline{S}_2$-equoid $F(M)$ that is freely generated by M with respect to the canonical $\underline{S}_1$-homomorphism $i_M: M \rightarrow F(M)\downarrow\underline{S}_1$ is called a <u>free extension</u> of M, if $i_M: M \rightarrow F(M)\downarrow\underline{S}_1$ is an isomorphism.

By this notion we can inductively define the concepts of a <u>signature chain</u> $\underline{\underline{S}}$ and of the model class $M(\underline{\underline{S}})$ of a signature chain $\underline{\underline{S}}$:

(I_1) Every signature $\underline{S}$ forms a signature chain $\underline{\underline{S}}$ (of length one) with $\underline{S}$ as associated signature. Every $\underline{S}$-model is also an $\underline{\underline{S}}$-model.

(I_2) Let $\underline{\underline{S}}$ be a signature chain with $\underline{S}$ as associated signature and let $\underline{S} \subseteq \underline{R}$ be a signature extension. Then $\underline{\underline{R}} = \underline{\underline{S}} \subseteq \underline{R}$ is a signature chain with $\underline{R}$ as associated signature. Every $\underline{R}$-model M, so that $M\downarrow\underline{S}$ is an $\underline{\underline{S}}$-model, is an $\underline{\underline{R}}$-model. Such an extension of $\underline{\underline{S}}$ to $\underline{\underline{R}} = \underline{\underline{S}} \subset \underline{R}$ is called a <u>proper extension</u> of $\underline{\underline{S}}$ by $\underline{R}$.

(I_3) Let $\underline{\underline{S}}$ be a signature chain with $\underline{S}$ as associated signature and let $\underline{S} \subseteq \underline{R}$ be a signature extension. Then we get a new signature chain $\underline{\underline{R}} = \underline{\underline{S}} \sqsubseteq \underline{R}$ if for every $\underline{\underline{S}}$-model M a free extension $F(M)$ of M to an $\underline{R}$-equoid exists. The associated signature of $\underline{\underline{R}} = \underline{\underline{S}} \sqsubseteq \underline{R}$ is $\underline{R}$, and we say that $\underline{\underline{R}} = \underline{\underline{S}} \sqsubseteq \underline{R}$ arises from $\underline{\underline{S}}$ through a <u>weak extension</u> by $\underline{R}$. The $\underline{\underline{R}}$-models are exactly those $\underline{R}$-equoids that are free extensions of $\underline{\underline{S}}$-models.

It is evident that not for every signature chain $\underline{\underline{S}}$ with $\underline{S}$ as associated signature and for every signature extension $\underline{S} \subseteq \underline{R}$ the weak extension $\underline{\underline{R}} = \underline{\underline{S}} \sqsubseteq \underline{R}$ exists. In general it is recursively undecidable wether or not the weak extension of a signature chain exists.

The weakly added sorts and operators in a signature chain $\underline{\underline{S}}$ with $\underline{S}$ as associated signature have to be interpreted as minimal solutions of the weakly added implications. In the case of total operations, $\pi(\omega)=1$ for all $\omega \in \Omega$, we get exactly the definition of abstract data types by

Goguen and others [7]. The more general concept of abstract data types in the language of signature chains can now be used to describe the context conditions of the production rules in the definition of a programming language.

Equation Grammars

If we convert the construction of a heterogeneous operator domain τ_G from a context-free grammar G, as described in [1] and [2], and if we start from the more general concept of signature chains, we get some kind of grammars, which we will call "equation grammars".

An equation grammar $G=(T,N,W,(\alpha_i)_{i=1,\ldots,k})$ consists of a finite set $T=\{x_1,\ldots,x_m\}$ of so-called terminals, of a finite set $N=\{X_1,\ldots,X_n\}$ of so-called non-terminals with $N\cap T=\emptyset$, of a so-called root $W\in N$, and of a finite sequence $(\alpha_i)_{i=1,\ldots,k}$ of production rules or attribute definitions or definitions of attribute functions.

A production rule is a triple consisting of a non-terminal, of a finite word over $T\cup N$, and of a domain condition, which is a possibly empty conjunction of term equations, where the terms are built up from attribute functions. For production rules we use the denotation

$$X_{j_o} \leftarrow w_{i_o} X_{j_1} w_{i_1} X_{j_2} \ldots X_{j_k} w_{i_k} \text{ iff } \tau_1 = \tau_1' \,\&\ldots\&\, \tau_r = \tau_r' \text{ ffi}$$

$$\text{where } X_{j_s} \in N,\ w_{i_s} \in T^*.$$

If the domain condition of a production rule is the empty conjunction then we write nothing, and this means that the applicability of the concerned production rule is not restricted.

An attribute definition is a triple consisting of a finite set of attribute sorts, of a finite set of defining operators and of a finite set of defining implications. It is possible to define in one definition of attribute functions simultaneously some attribute functions by a recursive system of implications. A definition of attribute functions contains the names of attribute functions that will be defined, contains the descriptions of the domains and ranges of the defined attribute functions, and contains a finite set of implications. The range of an attribute function is either an attribute sort, defined before, or a non-terminal.

For want of room we can't give here a complete definition of equation grammars. The informal description of equation grammars will be explained by an example which describes the set of all such finite trees, whose nodes are marked with finite subsets of $\{a,b,\ldots,z\}$ in such a way that for every path starting from the root of the tree the subsets corresponding to the nodes of the path are disjoint.

The equation grammar G is given by $T = \{ a,b,\ldots,z,[,],\},\{,;\}$,
$N = \{$char, char-sequ, tree, tree-sequ$\}$, W = tree and the following sequence:

α_1 : char ← a
α_2 : ← b
. . . .
α_{26}: ← z
α_{27}: attribute: boole | t,f: ( ) → boole end
α_{28}: a-functions: equal:(char,char) → boole
with equal(a,a)=equal(b,b)=...=equal(z,z)= t ,
equal(a,b)= ... = equal(y,z) = f end
α_{29}: attribute: set | λ : ( ) → set, k:(set,char) → set,
$\cup$:(set,set) → set, $\in$:(char,set) → boole,
$\cap$:(set,set) → set
with k(k(set,char),char)=k(set,char)in(set,char),
k(k(set,$char_1$),$char_2$)=k(k(set,$char_2$),$char_1$)
in(set,char,char),
$\cup$(set,λ)=set, $\cup$(set_1,set_2)= $\cup$(set_2,set_1),
$\cup$(set_1,k(set_2,char))=k($\cup$(set_1,set_2),char)in(set,set,char),
$\in$(char,λ)= f,
if$\in$(char,set)=t then$\in$(char,k(set,$char_2$))=t in(char,set,char)fi,
if$\in$(char,set)=f then$\in$(char,k(set,$char_2$))=equal($char_1$,$char_2$)
in(char,set,char)fi,
$\cap$(set,λ)= λ , $\cap$(set_1,set_2)= $\cap$(set_2,set_1),
if$\in$(char,set_1)=t then $\cap$(set_1,k(set_2,char))=k($\cap$(set_1,set_2),char)
in(set,set,char)fi,
if$\in$(char,set_1)=f then $\cap$(set_1,k(set_2,char))= $\cap$(set_1,set_2)
in(set,set,char)fi end
α_{30}: char-sequ ← char
α_{31}: ← char-sequ ; char iff
$\in$(char,members(char-sequ))=f ffi
α_{32}: a-functions: members:(char-sequ) → set
with members(α_{30}(char))=k(λ,char),
members(α_{31}(char-sequ,char))=k(members(char-sequ),char) end
α_{33}: tree ← [{char-sequ};]
α_{34}: ← [{char-sequ} ; tree-sequ] iff
$\cap$(members(char-sequ),members(tree-sequ))= λ ffi
α_{35}: tree-sequ ← tree
α_{36}: ← tree-sequ ; tree

α_{37}: a-functions: members:(tree-sequ) $\longrightarrow$ set,
members:(tree) $\longrightarrow$ set
with members([{char-sequ};])=members(char-sequ),
members([{char-sequ}; tree-sequ])=
$= \cup$(members(char-sequ),members(tree-sequ)),
members(tree-sequ ; tree) =
$= \cup$(members(tree-sequ),members(tree)),
members(α_{35}(tree))=members(tree) end

An example for a tree defined by G is given by

[{a;b};[{c;d;e};];[{c;f};[{g;h};];[{g;r;t};];[{d;e};]].

The example demonstrates that in analogy to BNF-grammars the production rules contain their syntactical interpretation as string functions over the set T of terminals of G.

In analogy to the approach of Knuth we can connect the semantical interpretation with every production rule of an equation grammar, and we get a complete description of syntax and semantics.
The class of formal languages definable by equation grammars is the class of typ 0 languages, i.e. recursively enumerable sets of strings.

References

[1] Goguen,J.A.,Initial Algebraic Semantics, IEEE Conf. Rec. SWAT, 15 (1974), 63-77.

[2] Goguen,J.A.,Semantics of Computation, Lecture Notes in Computer Science, vol. 25, 151-163.

[3] Knuth,D.E.,Semantics of Context-free Languages, Math.Syst.Theory, 1968, 2 , No 2, 127-145.

[4] Gladki,A.V.,Dikovski,A.J.,Theory of Formal Grammars,(Russian), Probability Theory, Mathematical Statistics, Theoretical Cybernetics, vol. 10, 107-142, Moscow, 1972.

[5] Kaphengst,H.,Reichel,H.,Operative Theorien und Kategorien von operativen Systemen, Studien zur Algebra und ihren Anwendungen, Berlin 1972, Akademie-Verlag.

[6] Kaphengst,H.,Reichel,H., Algebraische Algorithmentheorie, VEB Robotron, Zentrum für Forschung und Technik, WIB Nr. 1, 1971.

[7] Goguen,J.A.,Thatcher,J.W.,Wagner,E.G.,Wright,J.B., Abstract Data Types as Initial Algebras and Correctness of Data Representations, Proceedings, Conference on Computer Graphics, Pattern Recognition and Data Structures, May, 1975.