SOCIEDADE BRASILEIRA DE LÓGICA

PROCEEDINGS OF THE THIRD BRAZILIAN CONFERENCE ON MATHEMATICAL LOGIC

(UFPe - RECIFE, DECEMBER 17-22, 1979)

Edited by

A. I. ARRUDA, N. C. A. DA COSTA, A. M. SETTE

PROCEEDINGS OF THE THIRD BRAZILIAN CONFERENCE ON MATHEMATICAL LOGIC

Edited by

AYDA I. ARRUDA
Universidade Estadual de Campinas

NEWTON C. A. da COSTA
Universidade de São Paulo

ANTONIO MARIO SETTE
Universidade Estadual de Campinas

PROCEEDINGS OF THE THIRD BRAZILIAN
CONFERENCE ON MATHEMATICAL LOGIC
A. I. ARRUDA, N. C. A. da Costa and
A. M. Sette (eds.)

TOWARDS A LOGIC OF LIMITED PERCEPTION

ROBERTO LINS DE CARVALHO and PAULO AUGUSTO S. VELOSO

ABSTRACT. This paper introduces a semantical system more adequate to computer science than the classical ones. The key point is the usage of namable models (akin to ω-models) with minimal interpretations for the relation symbols. This formalizes the notion of class of data structures, and axiomatic specification, besides allowing a initiality property. The introduction of some new logical symbols is also suggested.

1. INTRODUCTION.

The development of semantical systems, such as the usual one for classical first-order logic and its extensions (modal, intensional, etc), has been directed mainly by the needs of classical mathematics, philosophy and linguistics. For some time now the field of computer science is emerging as another province in which logic is being increasingly used. Automatic theorem-proving and program verification are only two examples. However the world of the computer scientist is quite different from those of the mathematician, philosopher, linguist, etc., thus creating different needs.

In a very schematic way one can say that the art of computer programming has two main aspects:

a- The design of the program text itself.

b- The design of the data organization.

Those aspects are not independent of one another. Rather, their natural dependencies can be exploited to obtain better programs, this interweaving being an important tool in the solution of complex problems.

It is important to bear in mind that in both aspects above we are faced with problems of specification, namely

a'- Program specification,

b'- Data structure specification.

One way to specify a computer program P is to give input assertions ϕ, which give properties assumed about the input data, and output assertions $\psi(x,y)$, which state the required relations between input and output data.

To guarantee that P works as specified, one show that for all input data satisfying $\phi(x)$: (i) P halts (termination); (ii) whenever P does halt, then $\psi(x,y)$ is satisfied (partial correctness).

The latter proof reduces to proving $\phi'(x) \rightarrow \psi'(x,y)$ (transformed assertions) within the theory of the data.

For instance, a program to compute the gcd of two integers is proven partially correct by using some properties of the integers (cf.e.g. Manna [10], p. 71; Lucchesi et al. [9], p. 28). When the program manipulates structured data their properties must be formally specified for a proof to be possible.

Now, data are generally stored in the computer by means of memory organization know as *data structures* or *information structures*. Basically, data structures are (finite) sets of memory cells organized so as to be accessible from a small subset, the *entry points*.

For a simple example consider

$$c \xrightarrow{f} a_1 \xrightarrow{f} \cdots a_{n-1} \xrightarrow{f} d \circlearrowright f$$

Imagine that information is stored at each point a_i of this structure, the entry point of which is c. To gain access to the information at a

particular point one has to traverse the structure, starting at c and following the f-links. If c is denoted by a constant symbol then the accessible points are those that have a name $f \ldots f(c)$.

Such structures of arbitrary length, possibly together with the infinite "limit"

$$c \xrightarrow{f} a_1 \xrightarrow{f} \ldots \xrightarrow{f} a_n \xrightarrow{f} \ldots \; d \circlearrowright f$$

form the class of singly-linked linear lists.

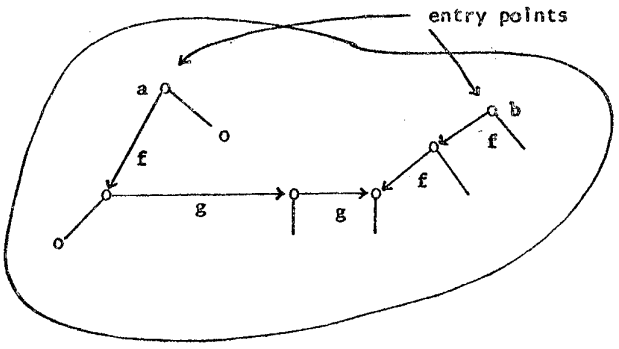
Thus, to specify data is to describe the properties of such structures in an appropriate language.

The group of theoretical computer science at PUC-RJ have been looking into the problem of formal specification of data structures. We shall discuss our needs in terms of a semantic system, i.e. a quadruple $(\mathrm{Sent}, M, V, \mathrm{val})$ where Sent is the language (set of sentences), M is the class of models, V is the set of abstract logical values, and $\mathrm{val}: M \times \mathrm{Sent} \longrightarrow V$ is the semantical evaluation function.

2. DATA STRUCTURES.

A data structure consists of cells, containing data, together with some prescribed way to access these cells, in order to store or retrieve data. In general this accessing mechanism consists of a path from an entry point (a directly accessible cell) to the desired cell; such a path provides a name for this cell.

Pictorially



the path starting at entry point a following the links f, g, g names a cell as ggf(a). Another path gives another name, ff(b).

From a model-theoretical viewpoint, consider a first-order language **L** with sets **C**, **F** and **R** of constant, function, and relation symbols, respectively. Let **T** denote the set of all variable-free terms of **L**. We always assume **C** non-empty, so that $\mathbf{T} \neq \emptyset$.

Now consider a structure $A = \langle \mathbf{A}, \mathbf{C}^A, \mathbf{F}^A, \mathbf{R}^A \rangle$ for **L**. Each $t \in \mathbf{T}$ denotes an element t^A in the domain, which defines a function $\mathbf{d}: \mathbf{T} \longrightarrow \mathbf{A}$ with image $\mathbf{T}^A$.

We shall call A *namable* (by **T**) iff $\mathbf{T}^A = \mathbf{A}$, i.e. the denotation **d** is surjective. So a **T**-*structure* is one in which every element is denoted by a variable-free term, which is a name for it.

A closely related notion is that of Γ-structure (Henkin [7]), where every element is denoted by a constant symbol. Clearly, Γ-structures and **T**-structures are the same structures with different languages. This seemingly minor distinction turns out to be important in the intended application, for two basic reasons:

- namability by **T** captures the mechanism of accessing by following links.

- The language is important here, it gives the basic operations, etc., available for programming.

Thus, we shall say that data structures, in a language, are (generally finite) namable structures of the language.

Let us return to our example of singly-linked linear lists presented in the Introduction. Assume **L** has a binary relation symbol *ac* the intended meaning of which is to be the reflexive-transitive closure of the graph of f, i.e. accessibility by f-links.

Consider the sentence:

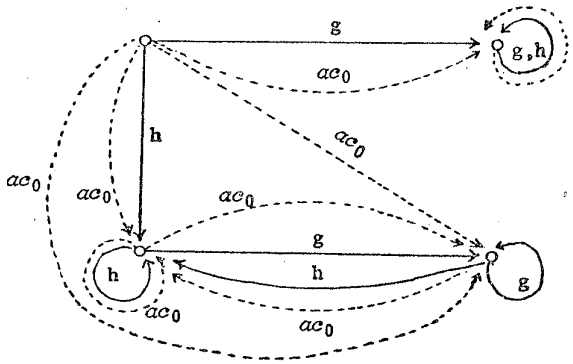
$$\forall x[ac(x,y) \leftrightarrow (x = c \wedge y = c) \vee (f(x) = y) \vee \exists z(ac(x,z) \vee f(z) = y)] \qquad (\alpha)$$

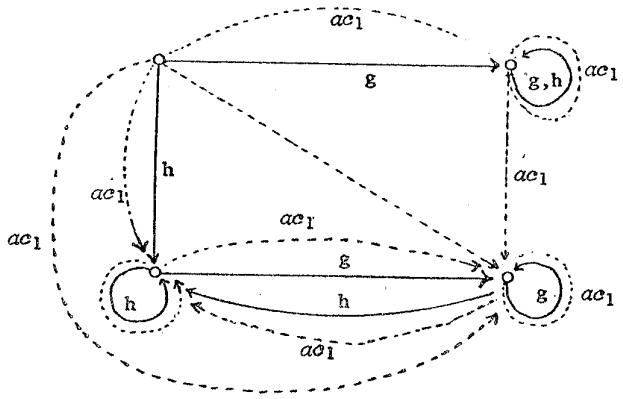
In this case, the intended interpretation for *ac* is the only one satsfying (α).

But, now consider the important case of two unary functions, $\mathbf{F} = \{g,h\}$ and again one entry point c. The natural analog of (α) is

$$\forall x \forall y [\, ac(x,y) \leftrightarrow (x = c \wedge y = c) \vee (g(x) = y \vee h(x) = y)$$
$$\exists z (ac(x,z) \wedge (g(z) = y \vee h(z) = y)] \qquad (\beta)$$

But then one can see that the two structures below satisfy (β)





Here the intended interpretation is the first structure, where $ac_0 \subseteq ac_1$.

We can say that here we are interested in the minimum model for (β). A minimum model (if it exists) of a set of sentences Γ on a domain $\mathbf{D}$ is the model of Γ in $\mathbf{D}$ whose relations are included one by one in the cor-

responding relations of any model of Γ in $\mathbf{D}$.

The points we want to make clear in this section are:

A) Data Structures are namable structures.

B) The concept of accessibility (needed in the description of such structures) is realized only in minimum models even when they are namable.

3. INITIALLITY.

The set $\mathbf{T}$ of variable-free terms can be given a natural algebraic structure, making it into the algebra T, freely generated by the constants (Grätzer [4], p. 162). Namely, take

$c^T = c$ and, for an n-ary $f \in F$, take $f^T(t_1, \ldots, t_n) = f\, t_1 \ldots t_n$.

Notice that any interpretation of the relation symbols will make T into a $\mathbf{T}$-structure.

Given a structure A for $\mathbf{L}$ the denotation $d: T \longrightarrow A$ is the unique homomorphism of T into the algebra A, which will be onto iff A is namable. Now, there is a natural way to interpret the relation symbols on T so that d becomes a strong homomorphism, namely

$$d^{-1}r^A = \{(t_1, \ldots, t_n)/(t_1^A, \ldots, t_n^A) \in r^A\}.$$

Call this structure $T(A)$ the *Herbrand structure induced by* A. Then for any sentence σ, containing no negative occurrence of $=$ interpreted as identity, we have $A \models \sigma$ iff $T(A) \models \sigma$.

Thus, any $\mathbf{T}$-structure is, up to a special quotient, a Herbrand structure. And Herbrand structures present the attractiveness of being specified by their positive diagrams (Chang_Keisler [2], p. 70).

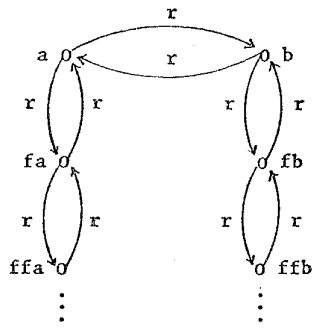
Consider a class $\mathbf{K}$ of $\mathbf{T}$-structures. Assume that $\mathbf{K}$ can be $\mathbf{T}$-axiomatized by Σ, i.e. $\mathbf{K}$ is the class of $\mathbf{T}$-structures satisfying Σ.

In some cases, we may be fortunate enough to have for each $A \in \mathbf{K}$ a simple set of "particularization axioms" Γ_A so that $\Sigma \cup \Gamma_A$ is a complete description of A, up to isomorphism. In any case, each $A \in \mathbf{K}$

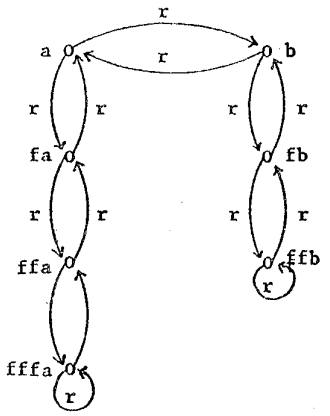
induces a structure $T(A)$ on the domain T. It would be nice to have on this domain a structure having the properties shared by all A in K.

Let us use a simple example to clarify what is intended. Let Σ consist of $r(a, b)$, $\forall x \forall y(r(x, y) \rightarrow r(y, x))$, $\forall x \forall y(r(x, y) \rightarrow r(x, f(x)))$.

Then the T-structures described by Σ look like the following:

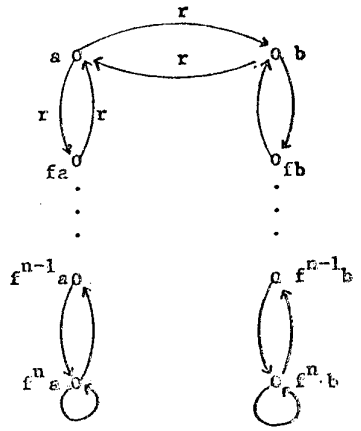


An example of a Herbrand structure $T(A)$ induced by a particular A is



One way to describe $T(A)$ is by saying that it is a Herbrand structure which satisfies $\Sigma \cup \{f^4(a) = f^3(a), f^2(b) = f^3(b)\}$. Similarly,

$\Sigma \cup \{f^{n+1}(a) = f^n(a),\ f^{n+1}(b) = f^n(b)\}$ describes



Notice that the first (infinite) structure is a sort of template for all the others.

More formally, make the algebra T into a T-structure $T(\Sigma)$, where for each $r \in R$, $r^{T(\Sigma)} = \cap\{r^{T(A)} / A \in K\}$ (equivalently $\bar{t} \in r^{T(\Sigma)}$, iff $r\bar{t}$ is a T-consequence of Σ).

Then, for every $A \in K$ there exists a unique weak homomorphism d of $T(\Sigma)$ onto A, hence preserving all positive formulas (A weak homomorphism h of A into B is a homomorphism of the corresponding algebras such that for each n-ary $r \in R$, $(h(a_1),\ldots,h(a_n)) \in r^B$ whenever $(a_1,\ldots,a_n) \in r^A$).

Unfortunately $T(\Sigma)$ may fail to satisfy Σ. Assume L has two distinct constant terms c and t and a unary predicate symbol p. For Σ given by the axiom $p(c) \vee p(t)$, we get $p^{T(K)} = \emptyset$. The same would happen were $\exists v\, p(v)$ the axiom. In either case $T(\Sigma)$ is not in K. The basic reason for the above situation is that Σ does not specify well enough its atomic consequences. For instance, if Σ consists of Horn sentences then $T(\Sigma) \in K$. Clearly $T(\Sigma) \in K$ iff $\cap\{T(A)/A \in K\} \in K$. In this case $T(\Sigma)$ is the initial structure of the class K.

The nature of equality is a point deserving some further comments. In the intended application the equality symbol should not always be interpreted as identity for two reasons.

- The data stored in the cells may be complex objects, so it is not reasonable to assume comparison as a single, primitive relation.
- Two cells may be non-identical because of implementation details (say, different storage addresses) which happen to be irrelevant from the logical viewpoint.

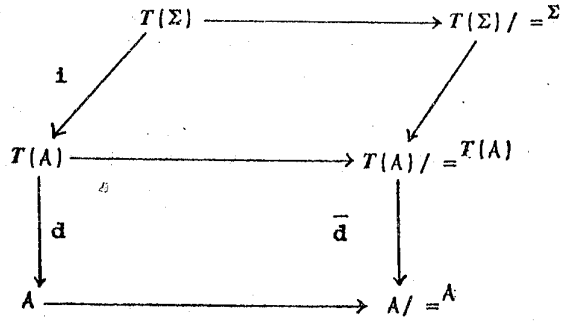
In this case we want to interpret equality as indistinguishability within the language. We then take on A

$a \overset{A}{\sim} a'$ iff for all $r \in R_m$, all $j = 1,\ldots, m$, and all $a_1,\ldots, a_m \in A$ $(a_1,\ldots,a_1,\ldots,a_n) \in r^A$ iff $(a_1,\ldots, a',\ldots, a_n) \in r^A$ and put $=^A$ to be the largest congruence contained in $\sim^A$.

For finite R we define $t \sim^\Sigma t'$ iff the formula corresponding to the rhs of the above definition is a T-consequence of Σ and put $=^\Sigma$ similarly.

By an *abstract data structure* (ADS, for short) we mean a T-structure where this identification has been performed, i.. one of the form $A/=^A$.

Since this identification is consistent with weak homomorphisms, we can summarize its effects in the commutative diagram below, where all the horizontal arrows are natural projections and i is a weak isomorphism of inclusion.



Thus in the category of ADS's with weak homomorphisms, $T(K)/=^{\Sigma}$ is an initial structure for the class of ADS's satisfying Σ.

4. ADDITIONAL CONSIDERATIONS.

When we describe mathematical structures using the apparatus of first order logic one generally considers their elements as "unstructured points".

In the description of ADS's, viewing their elements as mere "points" would not be satisfactory: it is necessary to describe some of their internal properties, as well, in addition to relations between elements. It is quite natural to employ two-level descriptions for such purposes. Likely candidates for this role would be second-order theories. But the usage of higher order logic presents several difficulties and inconveniences stemming their complexity.

The internal structure of the elements is generally fairly simple. In a frequent simple case they can be represented as strings, i.e., finite sequences of symbols drawn from some finite alphabet, the basic operation on them being concatenation.

One can try to give an axiomatic description of such objects by means of the first-order theory of semigroups, i.e., an axiom stating that the binary operation $\frown$ is associative:

$$\forall x \forall y \forall z((x \frown (y \frown z) = (x \frown y) \frown z).$$

Of course, this is not enough: we wanted free semigroups. Another point to appears to espace our first-order attempt is the finite length of the strings: we will have unintended models similar to the nonstandard models of the theory of naturals with addition.

The above problems could be circumvented metalinguistically, an approach we shall take in part. However, there is another problem: the difficulty in using even a simple axiom as the one above.

For instance, suppose a and b are constants denoting elements of the alphabet, the two terms $\frown(a, \frown(\frown(b, a), b))$ and $\frown(\frown(a, b), \frown(a, b))$ obviously denote the same string abab. However, in order to prove their equality within the formal system one would have to go through a deduction involving replacement and the above axiom.

Keeping in mind the intended interpretation a straightforward procedure is available for checking equality, namely: erase all parentheses, commas, and $\frown$ occurring in both terms and compare the results.

Formal languages used in logic generally comprise logical symbols and nonlogical symbols, the latter being susceptible of being interpreted. On the other hand, the logical symbols had their meanings fixed once and for all (despite philosophical disputes) since the early development of logic. Such was the case firstly for the sentential connectives and later and a lesser extent for the quantifiers and modalities.

For the example considered we introduce a new logical-like aymbol Δ, which as in the case of the sentential connectives is to be evaluated outside the domain of interpretation. However, the evaluation of Δ involves not truth-values but comparisons of strings. The formal (but metalinguistical) definition of Δ is as follows: $s \Delta t$ iff $\phi(s) = \phi(t)$ where $\phi: T \rightarrow V^*$ (T is the set of terms of the language and V^* is the free monoid generated by the set V of constants of the language except Δ) is given recursively by $\phi(c) = c$ for a constant c and $\phi(\frown(u, v)) = \phi(u)\,\phi(v))$.

The above recursive definition of ϕ could be replaced by an automaton, which would define ϕ much in the same way as truth-tables for sentential calculus.

Now if we put together our semantical consideration of namable structures, minimality and this (practical) morphological consideration we are limiting the kind of semantic system we are going to work with and (for our purpose as computer scientist) we expand our descriptive power. For instance, by allowing the description of a dynamic concept as that of rewriting system, and so (we hope) bringing about some uniformity in different description tools. So if we consider the rewriting relation $\rightarrow$ then we define its extension by

$$\text{gr. 1}\quad \forall x \forall y(\Rightarrow(x, y) \equiv \exists u \exists v \exists v' \exists w(\rightarrow(v,v') \wedge x \Delta \frown(u, \frown(v, w)) \wedge y \Delta \frown(u, \frown(v', w))))$$

and its reflexive transitive closure $\Rightarrow^*$ by

$$\text{gr. 2}\quad \forall x \forall z(\Rightarrow^*(x, y) \leftrightarrow (x = y \vee \exists z(\Rightarrow^*(x, z) \wedge \rightarrow(z, y)))).$$

5. CONCLUSIONS.

Our initial intention was to describe a semantical system ⟨Sent, M, V, val⟩ appropriate for the specification of data structure. Let us see where we stand:

First of all we have argued that data structures are *namable structures*, hence Herbrand universes suffice as domains for M.

The set Sent would consist of the first order sentences having models (natural structures as models). So it would include Horn sentences and recursive formulas such as (α) and (β) of Section 2.

The exact nature of Sent is not yet completely specified, as it depends on V and val, for which we have several apparently reasonable alternatives.

We can take V as the usual {T,F} and then $\text{val}(\phi, M) = T$ iff M is the natural structure for ϕ. In this case we have uniqueness of models, so an optimal descriptive power, but with the risk of loosing deductive power.

On the other hand, we could leave the realm of classical logic, by introducing extra logical symbols, as discussed in Section 4. This would force to enlarge V, for instance by including strings. This would give us back part of deductive power, with automata in lieu of deduction by equational reasoning. Other directions are also possible.

Some further research is still necessary in order ot determine which alternatives are more adequate. We think that several limited perception logics are.

REFERENCES.

[1] R. L. de Carvalho, T. S. E. Maibaum, T. H. C. Pequeno, A. A. Pereda B., P. A. S. Veloso, *A model-theoretic approach to the semantic data types and structures*, Forthcoming, 1980.

[2] C. C. Chang and H. J. Keisler, MODEL THEORY, North-Holland, Amsterdam, 1973.

[3] N. C. A. da Costa, *α-models for the systems T and T**, Notre Dame Journal of Formal Logic XV (1974), pp. 443-454.

[4] G. Grätzer, UNIVERSAL ALGEBRA, D. van Nostrand, Princeton, 1968.

[5] A. Grzegorczyk, AN OUTLINE OF MATHEMATICAL LOGIC, D. Reidel, Dordrect, 1974.

[6] P. Hájek and T. Havránek, MECHANIZING HYPOTHESIS FORMATION — MATHEMATICAL FOUNDATIONS FOR A GENERAL THEORY, Springer-Verlag, 1978.

[7] L. Henkin, *A generalization of the concept of ω-consistency*, The Journal of Symbolic Logic 19 (1954), pp. 183-196.

[8] Th. Lucas, *Une utilisation du langage des catégories pour la présentation de théorèmes de théorie des modèles*, Univ. Catholique de Louvain, Inst. Mathématiques, Rapp. 62, 1976.

[9] C. Lucchesi, Im. Simon, Ist. Simon, J. Simon and T. Kowaltowski, ASPECTOS TEÓRICOS DA COMPUTAÇÃO, IMPA, Rio de Janeiro, 1979.

[10] Z. Manna, THE MATHEMATICAL THEORY OF COMPUTATION, McGraw Hill, New York, 1974.

[11] T. H. C. Pequeno, P. A. S. Veloso, *Do not write more axioms than you have to*, International Computing Symposium, Taipei, China, Dec. 1978.

[12] A. A. Pereda B., MÉTODOS DE DESCRIÇÃO DE TIPOS DE DADOS E ESTRUTURAS DE DADOS, Univ. Católica do Rio de Janeiro, Depto. de Informática (Doctor's Thesis), Dec. 1979.

[13] J. Shoenfield, MATHEMATICAL LOGIC, Addison-Wesley, 1967.

Pontifícia Universidade Católica-RJ
Departamento de Informática
22453 Rio de Janeiro, RJ, Brazil.