SEYED-KAZEM LELLAHI

NICOLAS SPYRATOS

Deduction over graphs under constraints : a soundness and completeness theorem

Diagrammes, tome 29 (1993), exp. nº 2, p. LS1-LS24

<http://www.numdam.org/item?id=DIA_1993__29__A2_0>

DEDUCTION OVER GRAPHS UNDER CONSTRAINTS: A SOUNDNESS AND COMPLETENESS THEOREM

Seyed-Kazem LELLAHI
e-mail: kl@lipn.univ-paris13.fr
LIPN, URA 1507 du CNRS
Université de Paris 13, Institut Galilée
Av. J.B.Clément, 93430 Villetaneuse, France

Nicolas SPYRATOS
e-mail: spyratos@lri.fr
LRI, URA 410 du CNRS
Bât. 490, Université de Paris 12
91405 Orsay Cedex, France

Abstract

We introduce a notion of computation and a notion of constraint over graphs, and we give an inference system for deducing new computations or constraints from old. The graphs and the inference rules are interpreted in suitable enriched categories, which allow to define the model of a graph under constraints. We prove that our inference system is sound and complete.

Keywords: Categorical Semantics, Categorical Logic, Enriched Categories, Graph-Based Modelling of Knowledge, Algebraic Specification·

1 INTRODUCTION

In advanced computer applications, such as multimedia applications, entities of different systems must cooperate together. These entities have usually different representations and organizations, and they may come from a functional, or a logic, or an applicative or an object-oriented programming language as well as from a data or a knowledge base. As a consequence it seems necessary to have a uniform representation of all kinds of entities at conceptual level. Indeed such a representation makes easier interfacing of the systems concerned on the one hand,

A.M.S. SUB. CLASS. : 18 C 10, 68 C 99, 18 D 20

and the comprehension of the behavior of the whole system on the other hand. In any case, in order to answer user queries, the system must be able to combine old entities of various kinds to deduce new entities. Therefore, the uniform representation of entities should be equipped with constructs such that their instantiation in a given system provides constructs of that system.

In recent years the database community and the knowledge base community have been faced with this kind of problems. Some researchers have argued that at conceptual level data should be structured as graphs, and several graph-based models have been proposed recently [CoMe90], [GPV90], [Wedd92], [VaVa92], [KaVa93]. Others have proposed second order signatures as a modeling tool [Güti93], and some authors have tried to use category theory for such a modelling [TGP91], [TuGu92]. In [LeSp93, 92, 91] we have proposed a data model in which the conceptual level is presented as a graph with constraints and in which data are organized as partial/mutivalued functions, or more generally as morphisms of a special category. However, the approach of [LeSp93, 92, 91] is a model theoretic approach. In this paper we present a proof theoretic approach by introducing a system of effective inference rules, and we show how the deduction process, constructs a free adjoint functor. We prove soundness and completeness of the rules using the properties of enriched categories [Gray74], [Kelly82], [PoWe92] and the Yoneda Lemma [MacL71], [BaWe90]. This may be seen as the main result of the present paper.

The rest of the paper is organized as follows. In Sections 2 we present the notion of semantic universe and semantic function: a semantic universe is a $\mathbb{V}$-category, and a semantic function is a $\mathbb{V}$-functor [Gray74], [Kelly82], when $\mathbb{V}$ is a category of special posets called well-behaved posets. In Section 3 we introduce the basic concepts of graph and interpretation of a graph in a semantic universe. We give a syntactic way that constructs a semantic universe over a graph. The arrows of this semantic universe are called computations. We prove that the 'meaningful' computations with respect to a given interpretation for a given graph is a free construction in a comma category. In Section 4 we define the notion of constraint on a graph. Constraints are declarations of the form $p \leq e$, where p is a path and e is an edge, which must be enforced between computations. We present a set of inference rules which operate on a graph with constraints. Computations obtained by these rules are called computations under constraints. We introduce a notion of constraint satisfaction, a notion of model and a notion of constraint implication and we prove that the inference rules are sound and complete. This is the main result of the paper. In Section 5 we consider interpretations which are not models but are consistent with the constraints. Such an interpretation generates a canonical model and we prove that his model is constructed as a least fixpoint. We call this least fixpoint the *fixpoint semantics*. Finally, in Section 6, we offer some concluding remarks and suggestions for further research.

2 SEMANTIC UNIVERSES

A subset of a partially ordered set (*poset*) is said to be *consistent* if it is bounded. A *well-behaved poset*, or *wposet* for short, is a non empty poset in which every finite consistent subset A has a least upper bound, denoted $lubA$. In particular, the empty subset has a least upper bound called the *zero* of the wposet. Clearly every upper semi-lattice is a wposet. A morphism between two wposets is a function which preserves consistency and lub. That is f is a morphism of wposets if for every finite bounded subset A, of the source of f, the subset $f(A)$ is bounded, in the target of f, and $lub(f(A)) = f(lubA)$. An equivalent way is to say that f is a morphism of wposets if f preserves zero and whenever $\{a, b\}$ is consistent so is $\{fa, fb\}$ and $f(lub\{a, b\}) = lub\{fa, fb\}$. Such a function f is monotonic. The category of small wposets is denoted by $\mathcal{WP}$. It is easy to see that this category is finite complete and finite cocomplete. Moreover, limits and sums in $\mathcal{WP}$ can be computed pointwise.

Definition 1 A $\mathcal{WP}$-category [Gray74], [Kelly82] is called a *semantic universe.* ◇

In fact, semantic universes are special *2-categories* [PoWe92]. More precisely, given an arrow $u : A \to B$, call A the *source* of u, denoted $src(u)$, and B the *target* of u, denoted $tgt(u)$. Now, the category $\mathcal{C}$ is a semantic universe if

- for all objects A and B the set $\mathcal{C}(A, B)$ of arrows[1] from A to B is a wposet; the zero of this wposet is denoted $O(A, B)$,
- $O(tgt(u), A)u = O(src(u), A)$, and $uO(A, src(u)) = O(A, tgt(u))$, for every arrow u and every object A,
- for all arrows x, y, z and t as in the following configuration

$$\begin{tikzcd} H \arrow[r, "x"] & I \arrow[r, shift left, "y"] \arrow[r, shift right, "z"'] & J \arrow[r, "t"] & L \end{tikzcd}$$

  if (y, z) is a consistent pair, then so are (yx, zx) and (ty, tz) and we have:
  - $lub\{yx, zx\} = lub\{y, z\}x$ *(left continuity)*
  - $lub\{ty, tz\} = t\ lub\{y, z\}$ *(right continuity).*

Let $\leq$ denote the partial ordering over arrows of $\mathcal{C}$. We can prove that composition of arrows defines a monotonic function with respect to the ordering, that is, in the above configuration:
- if $y \leq z$ then $yx \leq zx$ *(right augmentation)*, and
- if $y \leq z$ then $ty \leq tz$ *(left augmentation)*

From now on, in a semantic universe we shall write '$\Rrightarrow$' instead of $\leq$. It is clear that the category $\mathcal{WP}$ is itself a semantic universe. Indeed, morphisms

1 All semantic universes considered in this paper are small or locally small categories

between two wposets can be ordered pointwise and this ordering satisfies the above axioms. Other interesting examples of semantic universes are:

- the category $\mathcal{S}et_{\perp}$ of sets and partial functions with the usual ordering on partial functions,
- the category $m\mathcal{S}et$ of sets and multivalued functions with pointwise inclusion,
- the category $\mathcal{B}rel$ of sets and binary relations with the usual ordering on binary relations, and
- the category $\mathcal{C}po$ of continuous functions between complete partial orderings [GuSc90].

A semantic universe with only one object is a monoid equipped with a well behaved partial ordering satisfying continuity. More precisely, let M be a set equipped with an associative product, an identity element e and a partial ordering '$\Rightarrow$' such that:

- there is an element ε of M, called zero, satisfying $\varepsilon \Rightarrow m$ and $\varepsilon m = m\varepsilon = \varepsilon$ for all m in M,
- for all elements m_1 and m_2 if there is m_3 such that $m_1 \leq m_3$ and $m_2 \leq m_3$, then $lub(m_1, m_2)$ exists, and
- if $lub(m_1, m_2)$ exists then so does $lub(m_3m_1, m_3m_2)$, for every m_3 in M, and
  $lub(m_1, m_2)m_3 = lub(m_1m_2, m_1m_3)$ and
  $m_3lub(m_1, m_2) = lub(m_3m_1, m_3m_2)$

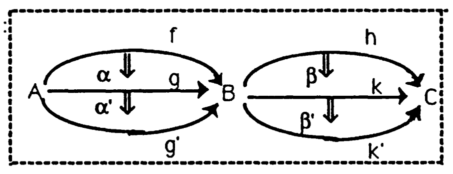
A sub-category $\mathcal{C}'$ of a semantic universe $\mathcal{C}$ is called a *sub-semantic universe* of $\mathcal{C}$ if $\mathcal{C}'$ equipped with the restriction of the ordering of $\mathcal{C}$ becomes a semantic universe.

A semantic universe $\mathcal{C}$ being a 2-category, $\mathcal{C}$ contains three category structures which cooperate together, as sated for general 2-categories in [PoWe92]. These three categories are:
The *base category* $\mathcal{C}$, whose class of objects will be denoted $\mathcal{C}_0$ and whose class of arrows will be denoted $\mathcal{C}_1$.
The *vertical category*, defined by the ordering '$\Rightarrow$' of $\mathcal{C}$, whose objects are all elements of $\mathcal{C}_1$ and whose arrows, called *2-cells*, are pairs of parallel arrows f, g such that $f \Rightarrow g$. Such a cell is seen as a *vertical arrow* from f to g, and the vertical composition is defined by the transitivity of the ordering. That is $(f \Rightarrow g).(g \Rightarrow h) = (f \Rightarrow h)$ corresponds to: if $(f \leq g$ and $g \leq h)$ then $f \leq h$. Moreover, in this category the coproduct of two objects f and g, when it exists, is $lub(f, g)$. So, we use $f \oplus g$ as an alternative notation for $lub(f, g)$.
The *horizontal category*, whose class of objects is $\mathcal{C}_0$, and whose class of arrows is the class $\mathcal{C}_2$ of all 2-cells. A 2-cell $f \Rightarrow g$ is seen as a *horizontal arrow* from A to B where A is the common source and B is the common target of parallel arrows f

and g. The horizontal composition is the composition in the category $\mathcal{C}\times\mathcal{C}$, that is $(h \Rightarrow k)\circ(f \Rightarrow g) = (hf \Rightarrow kg)$ if and only if $src(h) = tgt(f)$ and $src(k) = tgt(g)$. Thus, in a semantic universe, any pair $\alpha = (f, g)$ of arrows such that $f \Rightarrow g$, may be seen as a vertical or a horizontal arrow. We adopt the notation of [PoWe92] and we write $\alpha : f \Rightarrow g$ when α is seen as a vertical arrow, and we write $\alpha : f \Rightarrow g : A \to B$ when α is seen as a horizontal arrow.
The interaction between these three categories is expressed in the following well known proposition:

Proposition 1 In a semantic universe $\mathcal{C}$, for all configurations of objects, arrows and cells as in Figure 1, the following holds:

interchange law: $(\beta'\circ\alpha').(\beta\circ\alpha) = (\beta'.\beta)\circ(\alpha'.\alpha)$. ◇

Figure 1 :


Intuitively, a morphism between two semantic universes is a functor which preserves at least all these three structures. More precisely:

Definition 2 A $\mathcal{WP}$-functor [Gray74], [Kelly82] is called a *semantic function.* ◇

Thus, semantic functions are special 2-functors. More precisely a functor I from a semantic universe $\mathcal{C}$ to a semantic universe $\mathcal{C}'$ is a semantic function if the following holds:

- $I(O(A, B)) = O(IA, IB)$ for all objects A and B, and
- for all parallel and consistent arrows a and b in $\mathcal{C}$, the arrows Ia and Ib are consistent in $\mathcal{C}'$ and $I(a \oplus b) = Ia \oplus Ib$.

It is clear that such a functor is monotonic. We denote by $\mathcal{S}em$ the category whose objects are all locally small semantic universes and whose arrows are semantic functions.

3 COMPUTATIONS OVER A GRAPH

Graphs

In this paper, by *graph* we mean a directed labelled multigraph. An edge from node I to node J with label e is displayed as $e : I \to J$, or $I \xrightarrow{e} J$ or IeJ. The node I is called the source of e, denoted $src(e)$, and the node J is called the target of e, denoted $tgt(e)$. We assume that edges have distinct labels. A sequence $(e_1, e_2,..., e_n)$ of edges in a graph $\mathcal{G}$, is called a *$\mathcal{G}$-path* of *length* n, if $tgt(e_i) = src(e_{i+1})$ for every i, $1 \le i \le n-1$. We denote such a path by $e_1e_2...e_n$. Morphisms of graphs are defined as usual, and the category of locally small graphs is denoted by $\mathfrak{Gr}$.

Computations

Given a graph $\mathcal{G}$, we apply the recursive inference rules, of Figure 2 on $\mathcal{G}$ to obtain what we shall call *$\mathcal{G}$-computations*. In these rules, a computation φ from A to B is denoted $\varphi : A \longrightarrow B$, and

$$\frac{\Phi}{\Psi}$$

means that " in any context, if we have already a premise Φ then we can construct the consequence Ψ.

Figure 2 :

Inclusion	$\dfrac{e \quad (edge)}{e : src(e) \longrightarrow tgt(e)}$
identity	$\dfrac{A \quad (node)}{id(A) : A \longrightarrow A}$
Product	$\dfrac{\varphi : A \longrightarrow B \quad \psi : B \longrightarrow C}{\varphi.\psi : A \longrightarrow C}$
Addition	$\dfrac{\varphi : A \longrightarrow B \quad \psi : A \longrightarrow B}{\varphi+\psi : A \longrightarrow B}$
Parenthesis	$\dfrac{\varphi : A \longrightarrow B}{(\varphi) : A \longrightarrow B}$
zero	$\dfrac{A \quad B \quad (nodes)}{zero(A,B): A \longrightarrow B}$

If we regard src, tgt, id, '.' and '+' as operations, then $\mathcal{G}$-computations can be seen as terms obtained by applying these operations on the symbols representing nodes and edges of $\mathcal{G}$. A $\mathcal{G}$-computation of the form $e_1 e_2 ... e_n$ where all e_i are edges is identified with the path $e_1e_2....e_n$. A $\mathcal{G}$-computation of the form $id(A)$ is seen as a special $\mathcal{G}$-path of length 0 associated with A. Thus, there are several paths of length 0, one for each node. From now on by $\mathcal{G}$-path we mean a path of length 0, or a path of positive length. Two $\mathcal{G}$-computations are said to be *parallel* if they have common source and common target.

Note: It is important to note that all the above rules are syntactic rules. A computation may be seen as a program, and this is the reason why we denote the product of two consecutive computations $\varphi : A \longrightarrow B$ and $\psi : B \longrightarrow C$ by $\varphi.\psi$ and not by $\psi.\varphi$ (which is the usual notation in a category).

Equivalent Computations

Let p_1, p_2, ... , p_n be parallel $\mathcal{G}$-computations with source A and target B. We say the $\mathcal{G}$-computation $p_1+p_2+....+p_n$ is a *$\mathcal{G}$-expression* if each p_i is either a $\mathcal{G}$-path or the $\mathcal{G}$- computation $zero(A, B)$. In particular, every $\mathcal{G}$-path is a $\mathcal{G}$-expression. We shall see that expressions are canonical forms of computations.

Now, with every $\mathcal{G}$-expression e from A to B we associate a set $\mathcal{S}(e)$ of parallel $\mathcal{G}$-paths, from A to B, as follows:

- $\mathcal{S}(zero(A, B)) = \varnothing$,
- $\mathcal{S}(p) = \{p\}$, for every path p, and
- $\mathcal{S}(e+e') = \mathcal{S}(e) \cup \mathcal{S}(e')$, for all parallel $\mathcal{G}$-expressions e and e'.

The particularity of the set $\mathcal{S}(e)$ is that it is a set of parallel paths, so it contains at most one computation of the form $id(A)$ and no computation of the form $zero(A, B)$. We can extend the function $\mathcal{S}$ to all $\mathcal{G}$-computations using the following rules, where the extension is denoted $\mathcal{S}^*$.

For every expression e, all parallel computations φ, φ', all computations ψ, and all edges f with $src(f) = tgt(\varphi) = src(\psi)$, define:

- $\mathcal{S}^*(e) = \mathcal{S}(e)$,
- $\mathcal{S}^*((\varphi)) = \mathcal{S}^*(\varphi.id(src(\varphi))) = \mathcal{S}^*(id(tgt(\varphi)).\varphi) = \mathcal{S}^*(\varphi)$,
- $\mathcal{S}^*(\varphi+\varphi') = \mathcal{S}^*(\varphi) \cup \mathcal{S}^*(\varphi')$,
- $\mathcal{S}^*(\varphi.f) = \bigcup_{g \in \mathcal{S}^*(\varphi)} \mathcal{S}^*(g.f)$, and
- $\mathcal{S}^*(\varphi.\psi) = \bigcup_{f \in \mathcal{S}^*(\psi)} \mathcal{S}^*(\varphi.f)$.

Definition 3 Two $\mathcal{G}$-computations φ and ψ are said to be *equivalent*, denoted $\varphi \equiv \psi$, if they are parallel and $\mathcal{S}^*(\varphi) = \mathcal{S}^*(\psi)$. ◇

For instance, $f+f \equiv f$ and $f+g \equiv g+f$. Similarly, $f+g+zero(src(f), tgt(f)) \equiv f+g$, and so on. The following proposition is an immediate consequence of definitions.

Proposition 2 The relation $\equiv$ is a *congruence* relation for the operations *src*, *tgt*, *zero*, '.', '+' and parenthesizing. That is, $\equiv$ is an equivalence relation and:

- if $\varphi \equiv \psi$ then $src(\varphi) = src(\psi)$ and $tgt(\varphi) = tgt(\psi)$, for all parallel $\mathcal{G}$-computations φ and ψ,
- if $\varphi \equiv \psi$ then $(\varphi) \equiv (\psi)$, for all parallel $\mathcal{G}$-computations φ and ψ,
- if $\varphi \equiv \psi$ then $\varphi + \xi \equiv \psi + \xi$, for all parallel $\mathcal{G}$-computations φ, ψ and ξ,
- if $\varphi \equiv \psi$ then $\varphi.\xi \equiv \psi.\xi$ and $\rho.\varphi \equiv \rho.\psi$, for all parallel $\mathcal{G}$-computations φ, ψ, and all $\mathcal{G}$-computations ρ and ξ such that $\varphi.\xi$, $\psi.\xi$, $\rho.\varphi$ and $\rho.\psi$ are well defined. ◇

Let us denote by $[\varphi]$ the equivalence class of a $\mathcal{G}$-computation φ. Using the above proposition, the operations $[id]$, $[zero]$, $[src]$, $[tgt]$, $[.]$ and $[+]$ can be defined on equivalence classes of $\mathcal{G}$-computations as follows:

- $[id](A) = [id(A)]$, $[zero](A, B) = [zero(A, B)]$,
- $[src](\varphi) = src(\varphi)$, $[tgt](\varphi) = tgt(\varphi)$,
- $[\varphi][.][\psi] = [\varphi.\psi]$ and
- $[\varphi][+][\psi] = [\varphi+\psi]$.

Moreover, equivalence classes of $\mathcal{G}$-computations can be ordered by:

$[\varphi] \leq [\varphi']$ if and only if $\mathcal{S}^*(\varphi) \subseteq \mathcal{S}^*(\varphi')$

That is, $[\varphi] \leq [\varphi']$ if and only if $\varphi + \varphi' \equiv \varphi'$. Moreover, $[\varphi+\varphi'] = lub([\varphi], [\varphi'])$, for all computations φ and φ'.

For simplicity we denote the operations $[id]$, $[zero]$, $[src]$, $[tgt]$, $[.]$ and $[+]$ by id, $zero$, src, tgt, '.' and '+', respectively.

Proposition 3 The equivalence classes of $\mathcal{G}$-computations equipped with the operations $zero$, id and '.' , and the ordering '$\leq$' form a semantic universe, denoted $c\mathcal{G}$.

Proof The objects of $c\mathcal{G}$ are the nodes of $\mathcal{G}$, the arrows of $c\mathcal{G}$ are all equivalence classes of $\mathcal{G}$-computations, and the composition of arrows is defined by:

$[\psi][\varphi] = [\varphi].[\psi]$ if and only if $src(\psi) = tgt(\varphi)$.

The rest of the proof is obvious, but tedious, using the properties of union on sets and the definitions of $\mathcal{S}^*$ and $\equiv$. ◇

The function that associates each graph $\mathcal{G}$ with the semantic universe $c\mathcal{G}$ defines a functor c from the category $\mathcal{G}r$ to the category $\mathcal{S}em$. However, *c is not* a left free adjoint to the forgetful functor $U : \mathcal{S}em \longrightarrow \mathcal{G}r$. That is, there may exist a graph morphism I from $\mathcal{G}$ to a semantic universe $\mathcal{C}$ which cannot be freely extended to $c\mathcal{G}$ as a semantic function. Indeed for a computation φ which uses the addition rule, $I(\varphi)$ may be 'meaningless' in the semantic universe $\mathcal{C}$. However, we shall prove in the sequel, that I can be freely extended to a suitable sub-semantic universe of $c\mathcal{G}$.

The Meaning of Computations

Recall that the ordering of a general semantic universe is denoted '$\Rightarrow$' and its lub operation is denoted '$\oplus$'.

Definition 4 Given a graph $\mathcal{G}$, we say $(\mathcal{C}, \|\ \|)$ is an *interpretation* of $\mathcal{G}$, if $\mathcal{C}$ is a semantic universe and $\|\ \|$ is a graph morphism from $\mathcal{G}$ to $U\mathcal{C}$. ◇

When $\mathcal{C}$ is given, $\|\ \|$ is called a *$\mathcal{C}$-interpretation* of $\mathcal{G}$. Intuitively, for every node/edge x of $\mathcal{G}$, $\|x\|$ is the meaning of x in the universe of discourse $\mathcal{C}$. Now, the important question is: Can $\|\ \|$ be extended to all $\mathcal{G}$-computations ? The answer to this question depends on the semantic universe $\mathcal{C}$. For example, let $\mathcal{G}$ be the graph with only two parallel edges e and e', and let $\|\ \|$ be an interpretation of $\mathcal{G}$ in the semantic universe $\mathcal{S}et_\perp$ (i.e. the universe of partial functions). Suppose that the greatest lower bound of the functions $\|e\|$ and $\|e'\|$ does not exist. Then $e+e'$ is a $\mathcal{G}$-computation, but $\|\ \|$ cannot be extended to $e+e'$. However, if we consider

$\| \ \|$ as an interpretation in the universe of multivalued functions, then $\| \ \|$ can be extended to $e+e'$ by $\| \ \|(e+e')(x) = \|e(x)\| \cup \|e'(x)\|$, for every x in $src(\|e\|)$. This example shows that, given an interpretation $\| \ \|$ of a graph, $\| \ \|$ cannot necessarily be extended to all computations. We call those computations to which $\| \ \|$ can be extended *meaningful* computations with respect to $\| \ \|$. This partial extension of $\| \ \|$, denoted $\| \ \|_m$, is defined as follows:

Definition 5 Given an interpretation $\| \ \|$ of a graph $\mathcal{G}$,

- every $\mathcal{G}$-node or $\mathcal{G}$-edge x is meaningful and $\|x\|_m = \|x\|$,
- $id(A)$ is meaningful and $\|id(A)\|_m = id(\|A\|)$, for every node A,
- every $\mathcal{G}$-path $p = e_1e_2...e_n$ of positive length is meaningful and $\|p\|_m = \|e_n\| ... \|e_2\| \|e_1\|$,
- $zero(A,B)$ is meaningful and $\|zero(A,B)\|_m = 0(\|A\|_m, \|B\|_m)$, for all nodes A and B, and
- a $\mathcal{G}$-expression $e = p_1+p_2+...+p_n$ is meaningful if $\|p_1\|_m \oplus \|p_2\|_m \oplus ... \oplus \|p_n\|_m$ exists in $\mathcal{C}$, and then $\|e\|_m = \|p_1\|_m \oplus \|p_2\|_m \oplus ... \oplus \|p_n\|_m$. $\diamond$

Note that when the meaning function $\| \ \|_m$ is applied to a path $p = e_1e_2...e_n$ it reverses the order of edges in the path.

It follows from this definition that all equivalent $\mathcal{G}$-expressions have the same interpretation (if one exists). Now, let φ be a $\mathcal{G}$-computation, let $\mathcal{A}^*(\varphi) = \{p_1, p_2, ..., p_n\}$, and let $e_\varphi = p_1+p_2+...+p_n$. Moreover, let $e_{\sigma\varphi} = p_{\sigma(1)}+p_{\sigma(2)}+...+p_{\sigma(n)}$, where $\sigma\varphi$ stands for a permutation of the indices $1, 2, ..., n$ in e_φ. Since e_φ and $e_{\sigma\varphi}$ are equivalent, if $\|e_\varphi\|_m$ exists then $\|e_{\sigma\varphi}\|_m$ exists and $\|e_\varphi\|_m = \|e_{\sigma\varphi}\|_m$. The expression e_φ is called the canonical form of φ. Now, we can define the *meaning* of a computation as follows:

Definition 6 A computation φ is said to be *meaningful*, with respect to an interpretation $\| \ \|$, if $\|e_\varphi\|_m$ is defined; otherwise φ is said to be *meaningless*. If φ is meaningful, then $\|e_\varphi\|_m$ is called the *meaning* of φ. $\diamond$

For example if a, b and c are edges then $(a+b).c$ is meaningful if and only if $\|a.c\|_m \oplus \|b.c\|_m = \|b\| \|a\| \oplus \|c\| \|a\|$ exists. Moreover, $\|(a+b).c\|_m = \|b\| \|a\| \oplus \|c\| \|a\|$. We note that $(a+b).c$ may be meaningful while $(a+b)$ may not be meaningful.

It is not difficult to prove that:

- if φ is meaningful and $\varphi \equiv \varphi'$ then φ' is meaningful and $\|\varphi\|_m = \|\varphi'\|_m$,
- if φ and φ' are meaningful and $\varphi.\varphi'$ is defined then $\varphi.\varphi'$ is meaningful and $\|\varphi.\varphi'\|_m = \|\varphi'\|_m \|\varphi\|_m$, and
- if φ and φ' are meaningful and if $\|\varphi\|_m \oplus \|\varphi'\|_m$ is defined then $\varphi+\varphi'$ is meaningful and $\|\varphi+\varphi'\|_m = \|\varphi\|_m \oplus \|\varphi'\|_m$.

Thus we can state

Proposition 4 Let $\mathbf{G}$ be a graph. For every interpretation $(\mathcal{C}, \|\ \|)$ of $\mathbf{G}$, the equivalence classes of meaningful $\mathbf{G}$-computations, with respect to $\|\ \|$, form a semantic universe $m\mathbf{G}$ which is a sub-semantic universe of $c\mathbf{G}$. Moreover, $\|\ \|_m$ is a semantic function from $m\mathbf{G}$ to $\mathcal{C}$. ◇

We shall prove that the semantic universe $m\mathbf{G}$ may be seen as a free construction over $(\mathbf{G}, \|\ \|)$. Let us consider a category $\mathcal{S}em1$ in which the objects are semantic universes, and morphisms from $\mathcal{C}$ to $\mathcal{C}'$ are functors I such that the following holds:

- I preserves zero, and
- for all parallel arrows a and b in $\mathcal{C}$, if $Ia \oplus Ib$ exists in $\mathcal{C}'$ then $a \oplus b$ exists in $\mathcal{C}$ and $I(a \oplus b) = Ia \oplus Ib$.

It is easy to prove that the semantic function $\|\ \|_m$ is also a morphism of $\mathcal{S}em1$. A morphism $I : \mathcal{C} \to \mathcal{C}'$ of $\mathcal{S}em1$ is not necessarily a semantic function nor a monotonic function. However, if $a \Rightarrow b$ and $Ia \oplus Ib$ exists (for instance, if Ia and Ib are consistent) then $Ia \Rightarrow Ib$. Again we have a forgetful functor $V : \mathcal{S}em1 \to \mathcal{G}r$. Let us denote by $\mathcal{G}r/\mathcal{C}$ the category of objects over $V\mathcal{C}$ [MacL71]. An object of $\mathcal{G}r/\mathcal{C}$ is a pair $(\mathbf{G}, \|\ \|)$ where $\|\ \|$ is a $\mathcal{C}$-interpretation of $\mathbf{G}$. An arrow of $\mathcal{G}r/\mathcal{C}$ from $(\mathbf{G}, \|\ \|)$ to $(\mathbf{G}', \|\ \|')$ is a graph morphism $F : \mathbf{G} \to \mathbf{G}'$ satisfying $\|Fx\|' = \|x\|$, for every node or arrow x in $\mathbf{G}$. We define $\mathcal{S}em1/\mathcal{C}$ similarly, replacing graph morphisms by morphisms of $\mathcal{S}em1$. The forgetful functor V defines a forgetful functor $V/\mathcal{C}$ from $\mathcal{S}em1/\mathcal{C}$ to $\mathcal{G}r/\mathcal{C}$.

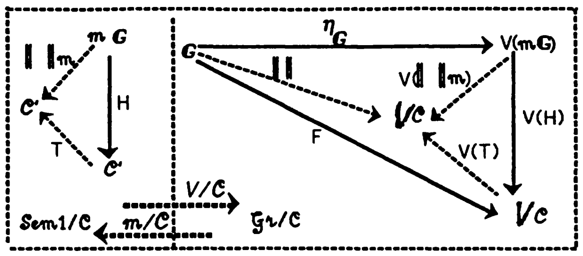
Theorem 1 Let $\mathcal{C}$ be a semantic universe. The function that associates each object $(\mathbf{G}, \|\ \|)$ of $\mathcal{G}r/\mathcal{C}$ with the object $(m\mathbf{G}, \|\ \|_m)$ of $\mathcal{S}em1/\mathcal{C}$, defines a functor $m/\mathcal{C} : \mathcal{G}r/\mathcal{C} \to \mathcal{S}em1/\mathcal{C}$ which is a left free adjoint to $V/\mathcal{C} : \mathcal{S}em1/\mathcal{C} \to \mathcal{G}r/\mathcal{C}$.

Proof Let $\eta_{\mathbf{G}} : \mathbf{G} \to V(m\mathbf{G})$ be the inclusion graph morphism. As $V(\|\ \|_m)\eta_{\mathbf{G}} = \|\ \|$, so $\eta_{\mathbf{G}}$ is an arrow of $\mathcal{G}r/\mathcal{C}$ from $(\mathbf{G}, \|\ \|)$ to $V/\mathcal{C}(m\mathbf{G}, \|\ \|_m) = (V(m\mathbf{G}), V(\|\ \|_m))$. We shall prove that $\eta_{\mathbf{G}}$ is the unit of an adjunction. The notations in the proof are referred to Figure 3. In this Figure, dotted lines represent objects and other lines represent arrows of the categories $\mathcal{G}r/\mathcal{C}$ or $\mathcal{S}em1/\mathcal{C}$.

Let $(\mathcal{C}', T)$ be an object of $\mathcal{S}em1/\mathcal{C}$ and let $F : \mathbf{G} \to V\mathcal{C}'$ be an arrow of $\mathcal{G}r/\mathcal{C}$ from $(\mathbf{G}, \|\ \|)$ to $V/\mathcal{C}(\mathcal{C}', T) = (V(\mathcal{C}'), V(T))$. Let φ be a meaningful computation with canoical form $e_\varphi = p_1 + p_2 + \dots + p_n$. Thus, $\|\varphi\|_m = \|e_\varphi\|_m = \|p_1\|_m \oplus \|e2\|_m \oplus \dots \oplus \|p_n\|_m$ exists in $\mathcal{C}$. If $p_i = e_{i_1} e_{i_2} \dots e_{i_{n_i}}$, define $p_i^* = F(e_{i_{n_i}}) \dots F(e_{i_2})F(e_{i_1})$. As $V(T)F = \|\ \|$ we have $V(T)p_i^* = \|e_{i_{n_i}}\| \dots \|e_{i_2}\| \|e_{i_1}\| = \|p_i\|_m$. So $\|\varphi\|_m = V(T)p_1^* \oplus \dots \oplus V(T)p_n^* = Tp_1^* \oplus \dots \oplus Tp_n^*$. We conclude that the arrow $\varphi^* = p_1^* \oplus \dots \oplus p_n^*$ exists in $\mathcal{C}'$ and $T(\varphi^*) = Tp_1^* \oplus \dots \oplus Tp_n^*$ (because T is a morphism of $\mathcal{S}em1$). If we define $H(\varphi) = \varphi^*$ then H is an arrow of $\mathcal{S}em1/\mathcal{C}$ as it satisfies $TH = \|\ \|_m$. Moreover, the equation $V(H)\eta_{\mathbf{G}} = F$ is satisfied in

$\mathcal{G}r/\mathcal{C}$ and H is the unique arrow of $\mathcal{S}em1/\mathcal{C}$ satisfying this equation. This completes the proof. ◇

Figure 3:



Initial Semantics An immediate consequence of this adjunction is the following: Consider the category of V-objects under $(G, \| \ \|)$ [MacL71]; an object of this category is a pair $(F, (\mathcal{C}', T))$ where $(\mathcal{C}', T)$ is an object of $\mathcal{S}em1/\mathcal{C}$ and $F : (G, \| \ \|) \to V/\mathcal{C}(\mathcal{C}', T)$ is an arrow of $\mathcal{G}r/\mathcal{C}$; an arrow from $(F, (\mathcal{C}', T))$ to

$(F', (\mathcal{C}'', T'))$ is a semantic function H satisfying $F(V(H)) = F'$. In this category the object $(\eta_G, (\mathcal{M}G, \| \ \|_{\mathcal{M}}))$ is an *initial object*. Therefore, we may call $(\mathcal{M}G, \| \ \|_{\mathcal{M}})$ the *initial semantics* of $(G, \| \ \|)$.

4 COMPUTATIONS UNDER CONSTRAINTS

Inference Rules

Informally, constraints are 'relationships' that must be enforced on computations and the question is: what is the effect of constraints on computations ? In other words, how do we compute under constraints? In this section we answer this and other related questions.

Definition 7 A *constraint* over a graph G is any statement of the form $\varphi \leq \varphi'$, where φ and φ' are parallel G-computations and $\varphi \neq \varphi'$. A *specification* $G|K$ consists of a graph G and a set K of constraints over G (usually G and K are finite). ◇

Intuitively, a specification $G|K$ is a graph G under constraints K, so $G|K$ can be read "G such that K". In order to extend the results of the previous section to computations under constraints we restrict our attention to a special class of constraints, called *edge constraint*, Such a constraint has the form $\pi \leq e$, where π is a path called the *body* of the constraint, and e is an edge called the *head* of the constraint. From now on constraints in a specification will be edge constraints.

Given a specification $\mathbb{G}|K$ we define $\mathbb{G}|K$-*computations* and their preordering, denoted $\leq_K^*$ using the recursive inference rules shown in Figure 4. It is important to note that $\mathbb{G}|K$-computations and their preordering $\leq_K^*$ are defined simultaneously.

Figure 4:

Constraint Inference rules		Computation Inference rules
Inclusion	$\dfrac{\pi \leq e \quad (constraint)}{\pi \leq_K^* e}$	$\dfrac{e \quad (edge)}{\pi : src(e) \longrightarrow tgt(e)}$
Reflexivity (and Identity)	$\dfrac{\varphi}{\varphi \leq_K^* \varphi}$	$\dfrac{A \quad (node)}{id(A) : A \longrightarrow A}$
Transitivity (and Product)	$\dfrac{\varphi \leq_K^* \psi \quad \psi \leq_K^* \theta}{\varphi \leq_K^* \theta}$	$\dfrac{\varphi : A \longrightarrow B \quad \psi : B \longrightarrow C}{\varphi.\psi : A \longrightarrow C}$
Addition	$\dfrac{\varphi \leq_K^* \theta \quad \psi \leq_K^* \theta}{\varphi + \psi \leq_K^* \theta \quad \varphi \leq_K^* \varphi + \psi \quad \psi \leq_K^* \varphi + \psi}$	$\dfrac{\varphi \leq_K^* \theta \quad \psi \leq_K^* \theta}{\varphi + \psi : src(\theta) \longrightarrow tgt(\theta)}$
Parenthesis		$\dfrac{\varphi : A \longrightarrow B}{(\varphi) . A \longrightarrow B}$
Zero	$\dfrac{\varphi}{zero(src(\varphi), tgt(\varphi)) \leq_K^* \varphi}$ $\dfrac{\varphi \quad X\ (node)}{zero(X, src(\varphi))\varphi \leq_K^* zero(X, tgt(\varphi))}$ $\dfrac{\varphi \quad X\ (node)}{\varphi zero(tgt(\varphi), X) \leq_K^* zero(src(\varphi), X)}$	$\dfrac{A \quad B\ (nodes)}{zero(A, B) \cdot A \longrightarrow B}$
Augmentation	$\dfrac{\varphi \leq K^* \psi \quad \xi : tgt(\varphi) \longrightarrow A}{\varphi.\xi \leq_K^* \psi.\xi}$ $\dfrac{\varphi \leq K^* \psi \quad \rho : A \longrightarrow src(\varphi)}{\rho.\varphi \leq K^* \rho.\psi}$	
Distributivity	$\dfrac{(\varphi + \psi) \quad \xi : tgt(\varphi) \longrightarrow A}{(\varphi + \psi).\xi \leq_K^* \varphi.\xi + \psi.\xi}$ $\dfrac{(\varphi + \psi) \quad \rho : A \longrightarrow src(\varphi)}{\rho.(\varphi + \psi) \leq K^* \rho.\varphi + \rho.\psi}$	

We note that all $\mathbb{G}$-paths and, in particular, all $\mathbb{G}$-edges are $\mathbb{G}|K$-computations. Thus the $\mathbb{G}$-computations used in edge constraints are already $\mathbb{G}|K$-computations. We also note that the operation '+' of $\mathbb{G}|K$-computations is a restriction of the operation '+' for $\mathbb{G}$-computations. Indeed, the application of '+' is now conditioned on the existence of a bound for the operands.

We use the notation $K \vdash \varphi' \leq \varphi$ (read K *infers* $\varphi \leq \varphi'$) as an alternative notation for $\varphi \leq_K^* \varphi'$. In fact $\varphi \leq_K^* \varphi'$ means that φ and φ' are $\mathbb{G}|K$-computations and the constraint $\varphi \leq \varphi'$ is deduced from K using the inference rules of Figure 4. Given two sets of constrains K and K' over $\mathcal{G}$, we write $K \vdash K'$ if $K \vdash k$ for every k in K'.

Now, two $\mathcal{G}|K$-computations φ and φ' are said to be ***equivalent***, denoted $\varphi \equiv_K \varphi'$, iff $\varphi \overset{*}{\leq}_K \varphi'$ and $\varphi' \overset{*}{\leq}_K \varphi$. We extend the relation $\equiv_K$ as follows:

$\varphi \equiv_K (\varphi) \equiv_K id(src(\varphi)).\varphi \equiv_K \varphi.id(tgt(\varphi))$, for every $\mathcal{G}|K$-computation φ.

One can prove that this extended relation $\equiv_K$ is actually a congruence relation and has all properties seen earlier for the relation $\equiv$. Therefore, the operations src, tgt, '.', $zero$ and id can be defined on equivalence classes of $\mathcal{G}|K$-computations. These operations have the same properties as in Section 3. For example, from the addition rules, the operation '+' is idempotent and product is distributive with respect to addition. Moreover, the preordering $\overset{*}{\leq}_K$ induces an ordering on parallel $\mathcal{G}|K$-computations, denoted $\leq_K$, which has also the same properties as the ordering $\leq$ defined earlier on $\mathcal{G}$-computations. In particular, $[\varphi] \leq_K [\varphi']$ if and only if $[\varphi] + [\varphi'] \equiv_K [\varphi']$, i.e. if and only if $\varphi + \varphi' \equiv_K \varphi'$. In view of the above rules and definitions, Proposition 4 can be extended as follows:

Proposition 4 (continued) The equivalence classes of $\mathcal{G}|K$-computations equipped with the operations $zero$, id and '.', and the ordering '$\leq_K$' form a semantic universe, denoted $c\mathcal{G}|K$. ◇

The meaning of Computations under Constraints

Clearly, every $\mathcal{G}|K$-computation is a $\mathcal{G}$-computation. Thus as in the previous section one can define the meaning of a $\mathcal{G}|K$-computation with respect to an interpretation $(\mathcal{C}, \| \|)$ of $\mathcal{G}$. We recall that not all $\mathcal{G}$-computations are necessarily meaningful, and thus not all $\mathcal{G}|K$-computations are necessarily meaningful. We also recall, however, that two meaningful and equivalent $\mathcal{G}$-computations have the same meaning. This unfortunately, is not always the case for $\mathcal{G}|K$-computations, i.e. two meaningful and equivalent $\mathcal{G}|K$-computations may have ***different*** meanings. For example let $\mathcal{C} = \mathcal{S}et_\perp$ and $K = \{e \leq e'\}$, where e and e' are edges. The computation $e+e'$ is a $\mathcal{G}|K$-computation obtained by addition and reflexivity rules. Now, if $lub(\|e\|, \|e'\|)$ exists in $\mathcal{C}$, but $lub(\|e\|, \|e'\|) \neq \|e'\|$ then $\|e+e'\|_m = lub(\|e\|, \|e'\|) \neq \|e'\| = \|e'\|_m$. However, $e+e'$ and e' are equivalent $\mathcal{G}|K$-computations.

So the effect of constraints on computations is that the constraints may cause violation of our basic requirement, i.e. that two meaningful and equivalent computations must have the same meaning. Clearly, this is an undesirable situation, and in what follows we characterize interpretations in which equivalent $\mathbb{G}|K$-computations *do* have the same meaning, if they have a meaning at all.

Definition 8 Given an interpretation $(\mathcal{C}, \|\ \|)$ of a graph $\mathbb{G}$, we say that

- $(\mathcal{C}, \|\ \|)$ *satisfies* the constraint $\varphi \leq \varphi'$, denoted $\|\ \| \models_{\mathcal{C}} \varphi \leq \varphi'$, if φ and φ' are meaningful with respect to $(\mathcal{C}, \|\ \|)$ and $\|\varphi\|_m \Rightarrow \|\varphi'\|_m$,
- $(\mathcal{C}, \|\ \|)$ satisfies a set K of constraints, denoted $\|\ \| \models_{\mathcal{C}} K$, if $(\mathcal{C}, \|\ \|)$ satisfies every constraint in K, and
- $(\mathcal{C}, \|\ \|)$ is a *model* of the specification $\mathbb{G}|K$ if $\|\ \| \models_{\mathcal{C}} K$. ◇

To simplify matters we shall drop the index $\mathcal{C}$ when no confusion is possible. Clearly, in the absence of constraints, every $\mathcal{C}$-interpretation of $\mathbb{G}$ is a $\mathcal{C}$-model of $\mathbb{G}$. Note that if $\pi = e_1 e_2 \ldots e_n$ is a path then π is meaningful and $\|\pi\|_m = \|e_n\| \|e_{n-1}\| \ldots \|e_1\|$. Thus, $(\mathcal{C}, \|\ \|)$ satisfies the edge constraint $\pi \leq e$ if $\|\pi\| \Rightarrow \|e\|$. A specification $\mathbb{G}|K$ has always at least one $\mathcal{C}$-model, namely, any interpretation I such that $I(e) = O(src(Ie), tgt(Ie))$ for every edge e present in K, is a model of $\mathbb{G}|K$. Such $\mathcal{C}$-models are called *trivial* $\mathcal{C}$-models. From now on by $\mathcal{C}$-model we mean a nontrivial $\mathcal{C}$-model.

Let K be a set of constraints and let k be a constraint, which may or may not be in K. We say that K ***implies k*** denoted $K \models k$, if every interpretation which satisfies K also satisfies k. Given two sets of constraints K and K' over $\mathbb{G}$, we say that K implies K', denoted $K \models K'$, if K implies every constraint in K'.

Theorem 2 The constraint inference rules of Figure 4 are sound and complete. That is, for every specification $\mathbb{G}|K$, if k is a constraint over $\mathbb{G}$ then $K \models k$ if and only if $K \vdash k$.

Proof Note that soundness means that the consequences of any constraint inference rules are implied from its hypotheses. Let $(\mathcal{C}, \|\ \|)$ be an interpretation which satisfies a set of constraints K. Clearly the inclusion rule is sound. Since $\Rightarrow$ is an ordering in the semantic universe, reflexivity and transitivity rules are sound too. To prove the soundness of addition rule, assume φ, ψ and θ to be $\mathbb{G}|K$-computations and assume $\varphi \overset{*}{\leq}_K \theta$ and $\psi \overset{*}{\leq}_K \theta$ to be satisfied, and prove that $\varphi + \psi \overset{*}{\leq}_K \theta$, $\varphi \overset{*}{\leq}_K \varphi + \psi$ and $\psi \overset{*}{\leq}_K \varphi + \psi$ are satisfied. The assumption means that φ, ψ and θ are meaningful and $\|\varphi\|_m \Rightarrow \|\theta\|_m$ and $\|\psi\|_m \Rightarrow \|\theta\|_m$ in $\mathcal{C}$. Thus, $\|\varphi\|_m \oplus \|\psi\|_m$ exists in $\mathcal{C}$, $\|\varphi\|_m \oplus \|\psi\|_m \leq \|\theta\|_m$, $\|\varphi\|_m \leq \|\varphi\|_m \oplus \|\psi\|_m$ and $\|\psi\|_m \leq \|\varphi\|_m \oplus \|\psi\|_m$. Thus, it is enough to prove that $\|\varphi\|_m \oplus \|\psi\|_m = \|\varphi + \psi\|_m$. With earlier notations we can write $\|\varphi + \psi\|_m = \|e_{\varphi+\psi}\|_m$, $\|\varphi\|_m = \|e_\varphi\|_m$ and $\|\psi\|_m = \|e_\psi\|_m$. But $\|e_{\varphi+\psi}\|_m$ is the sum of paths in $\mathcal{S}^*(\varphi + \psi)$. Similarly, $\|e_\varphi\|_m$ is the sum of paths in $\mathcal{S}^*(\varphi)$ and $\|e_\psi\|_m$ is the sum of paths in $\mathcal{S}^*(\psi)$. As $\mathcal{S}^*(\varphi + \psi) = \mathcal{S}^*(\varphi) \cup \mathcal{S}^*(\psi)$ so $\|\varphi + \psi\|_m$ differs from $\|\varphi\|_m \oplus \|\psi\|_m$ by a permutation of its terms and by repetition of some of its terms. But $\oplus$ is idempotent, associative and commutative, so $\|\varphi + \psi\|_m = \|\varphi\|_m \oplus \|\psi\|_m$.

The proof of the soundness of the remaining rules is similar. In fact each rule reflects a property of the ordering of the semantic universe. Augmentation reflects the monotonicity of composition, zero reflects the axiom of zero and distributivity reflects continuity.
To prove completeness we show that if a constraint k cannot be inferred from K using the rules, then there must be a nontrivial model $(I, \mathcal{C})$ which does not satisfy k. We shall give such a model when $\mathcal{C}$ is $\mathcal{WP}$, using the (enriched) Yoneda's Lemma [MacL71] as follows:
Let X be a node in $\mathcal{G}$ and consider the graph morphism $I_X : \mathcal{G} \rightarrow \mathcal{WP}$ defined by:

- for every node A, $I_X(A) = c\mathcal{G}|K(X, A)$, i.e. the well behaved set of all equivalece classes of $\mathcal{G}|K$-computations from X to A, and
- for every edge u, $I_X(u)$ is the morphism from $I_X(src(u))$ to $I_X(tgt(u))$ that associates the class of xu with every class x.

The graph morphism I_X is a $\mathcal{WP}$-model. Obviously I_X is a nontrivial interpretation. Moreover, if $p \leq e$ is in K then for every $u : X \rightarrow src(p)$ we can write $u.p \overset{*}{\leq}_K u.e$ that is, $I_X(p)(u) \overset{*}{\leq}_K I_X(e)(u)$. In other words $I_X(p) \Rightarrow I_X(e)$ which

means that I_X satisfies $p \leq e$. Now, assume that $k = c \leq c'$ and $K \not\vdash k$. The model $I_{src(c)}$ does not satisfy k, otherwise we must have $I_X(c) \Rightarrow I_X(c')$. This means that

$I_{src(c)}(c)(u) \overset{*}{\leq}_K I_{src(c)}(c')(u)$ or, equivalently,

$u.c \overset{*}{\leq}_K u.c'$, for every $u : src(c) \rightarrow src(c)$.

Taking $u = id(src(c))$ we must have $c \overset{*}{\leq}_K c'$ which contradicts $K \not\vdash k$. This complete the proof. ◇

Now, we shall prove the soundness and completeness of computation rules. Let us first define $\vdash$ and $\models$ notations for computations. We write
$K \vdash \varphi$ when φ is a $\mathcal{G}|K$-computation. That is, φ is a $\mathcal{G}$-computation and φ can be obtained by a finite number of application of computation rules beginning with $\mathcal{G}$ and K. Similarly, $K \models \varphi$ means that φ is a $\mathcal{G}$-computation and φ is meaningful with respect to every model $(\mathcal{C}, \| \|)$ of K.

Theorem 3 The inference computation rules of Figure 4 are sound and complete, i.e. for each set K of constraints over $\mathcal{G}$, and for each $\mathcal{G}$-computation φ, $K \vdash \varphi$ if and only if $K \models \varphi$.

Proof The proof of soundness of computation rules is already contained in the proof of Theorem 2. To prove completeness we use again the enriched Yoneda's Lemma. Let φ be a $\mathcal{G}$-computation such that φ is meaningful with respect to every model $(\mathcal{C}, \| \|)$ of K. We must prove that φ is a $\mathcal{G}|K$-computation. Let $X = src(\varphi)$ and consider the model $(\mathcal{WP}, I_X)$ constructed in the proof of Theorem 2. Thus, $I_X(\varphi)$ is meaningful in $\mathcal{WP}$. In particular $I_X(\varphi)(id(X)) = \varphi$ has a meaning in $c\mathcal{G}|K$, that is $K \vdash \varphi$. This completes the proof. ◇

Corollary 1 If $(\mathcal{C}, \| \|)$ is a model of the specification $\mathbb{G}|K$ then any two equivalent $\mathbb{G}|K$-computations have the same meaning with repect to $(\mathcal{C}, \| \|)$.

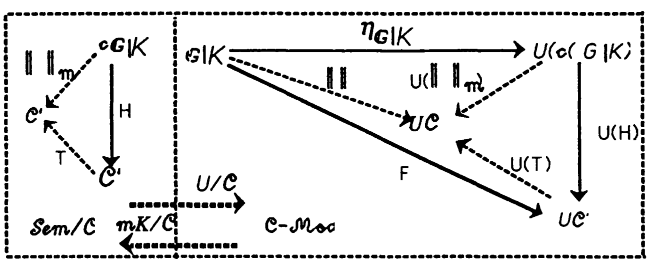
Proof if $(\mathcal{C}, \| \|)$ is a model then, by Theorem 2, all $\mathbb{G}|K$-computations are meaningful. Moreover, if $\varphi \equiv_K \psi$ then if $\varphi \overset{*}{\leq}_K \psi$ and $\psi \overset{*}{\leq}_K \varphi$ then we can write $\|\varphi\|_m \Rightarrow \|\psi\|_m$ and $\|\psi\|_m \Rightarrow \|\varphi\|_m$ so $\|\psi\|_m = \|\varphi\|_m$. Otherwise, ψ has the form (φ), $id.\varphi$ or $\varphi.id$. In all these cases we obviously have $\|\psi\|_m = \|\varphi\|_m$. ◇

Specifications are objects of a category $\mathcal{S}pec$. A morphism of $\mathcal{S}pec$ from $\mathbb{G}_1|K_1$ to $\mathbb{G}_2|K_2$ is a graph morphism $F : \mathbb{G}_1 \to \mathbb{G}_2$ such that for every constraint $e_1e_2...e_n \leq e$ in K_1 the constraint $Fe_1.Fe_2....Fe_n \leq Fe$ is in K_2. Let us see a semantic universe $\mathcal{C}$ as a specification (not necessarily finite) whose constraints are all $\pi \Rightarrow e$ where $\pi = e_1.e_2....e_n$ is a path and $e = e_{n1}.e_{n-1}....e_1$. As every semantic function is monotonic, so there is a forgetful functor $U : \mathcal{S}em \to \mathcal{S}pec$. Let us denote by $\mathcal{S}pec/\mathcal{C}$ the category of objects over $U\mathcal{C}$ and by $\mathcal{S}em/\mathcal{C}$ the category of objects over $\mathcal{C}$. We denote by $\mathcal{C}$-$\mathcal{M}od$ the full sub-category of $\mathcal{S}pec/\mathcal{C}$ whose objects are $\mathcal{C}$-models. The functor U defines a functor $U/\mathcal{C} : \mathcal{S}em/\mathcal{C} \to \mathcal{C}$-$\mathcal{M}od$. Theorem 1 and its proof can now be extended as follows:

Theorem 1 (continued) Let $\mathcal{C}$ be a semantic universe. The function that associates each object $(\mathbb{G}|K, \| \|)$ of $\mathcal{C}$-$\mathcal{M}od$ with the object $(c\mathbb{G}|K, \| \|_m)$ of $\mathcal{S}em/\mathcal{C}$ defines a functor $mK/\mathcal{C} : \mathcal{C}$-$\mathcal{M}od \to \mathcal{S}em/\mathcal{C}$ which is a left free adjoint to the functor $U/\mathcal{C} : \mathcal{S}em/\mathcal{C} \to \mathcal{C}$-$\mathcal{M}od$.

Proof Since $\| \|_m$ is a semantic function $\| \|_m$ is a $\mathcal{C}$-model of $c\mathbb{G}|K$, so

$U/\mathcal{C}(c\mathbb{G}|K, \| \|_m)$ is an object of $\mathcal{C}$-$\mathcal{M}od$. Let $\eta_{\mathbb{G}|K} : \mathbb{G}|K \to U(c\mathbb{G}|K)$ be the inclusion graph morphism. As $U(\| \|_m)\eta_{\mathbb{G}|K} = \| \|$, so $\eta_{\mathbb{G}|K}$ is an arrow of $\mathcal{C}$-$\mathcal{M}od$ from object $(\mathbb{G}|K, \| \|)$ to $U/\mathcal{C}(c\mathbb{G}|K, \| \|_m) = U/\mathcal{C}(c\mathbb{G}|K, \| \|_m)$. We shall prove that $\eta_{\mathbb{G}|K}$ is the unit of an adjunction (see Figure 5, with the same convention as in Figure 3).

Figure 5


Let $(\mathcal{C}', T)$ be an object of $\mathcal{S}em/\mathcal{C}$ and let $F : \mathbb{G} \to U\mathcal{C}'$ be an arrow of $\mathcal{C}$-$\mathcal{M}od$ from $(\mathbb{G}|K, \| \|)$ to $U/\mathcal{C}(\mathcal{C}', T) = (U(\mathcal{C}'), U(T))$. Let φ be a $\mathbb{G}|K$-computation with canonical form $e_\varphi = p_1 + p_2 + \ldots + p_n$. As $\| \; \|$ is a model, φ is meaningful, that is, $\|\varphi\|_m = \|p_1\|_m \oplus \|p_2\|_m \oplus \ldots \oplus \|p_n\|_m$ exists in $\mathcal{C}$. As φ is obtained by the addition rule, there is a computation π such that $p_i \leq_K \pi$ for all i, $1 \leq i \leq n$. But F is an arrow of $\mathcal{C}$-$\mathcal{M}od$, so $F(p_i) \Rightarrow_K F(\pi)$ for all i, $1 \leq i \leq n$. This implies that $\varphi^* = F(p_1) \oplus \ldots \oplus F(p_n)$ exists in $\mathcal{C}'$. If we define $H(\varphi) = \varphi^*$ then H is a semantic function and defines an arrow of $\mathcal{S}em/\mathcal{C}$, as it satisfies $TH = \| \; \|_m$. Moreover, the equation $U(H)\eta_{\mathcal{G}|K} = F$ is satisfied in $\mathcal{C}$-$\mathcal{M}od$ and H is the unique arrow of $\mathcal{S}em/\mathcal{C}$ satisfying this equation. This completes the proof. ◇

Initial Semantics Similarly to Section 3 we can call $(c\mathcal{G}|K, \| \; \|_m)$ the *initial semantics* of $(\mathcal{G}|K, \| \; \|)$ because it provides an initial object in the category of U-objects under $(\mathcal{G}|K, \| \; \|)$.

5 CONSISTENCY AND FIXPOINT SEMANTICS

Let $\mathcal{G}|K$ be a specification and let $\| \; \|$ be an interpretation of $\mathcal{G}$ in some semantic universe $\mathcal{C}$. If $\| \; \|$ is not a model of $\mathcal{G}|K$ then the question is: should we discard $\| \; \|$? The answer is no, provided that $\| \; \|$ is consistent with K because, as we shall prove, a consistent interpretation can be embedded in a model.

Definition 9 Given a specification $\mathcal{G}|K$, we say that an interpretation $(\mathcal{C}, \| \; \|)$ of $\mathcal{G}$ is *consistent* with K if every $\mathcal{G}|K$-computation is meaningful with respect to $\| \; \|$. ◇

By Theorem 2, every model $(\mathcal{C}, \| \; \|)$ of $\mathcal{G}|K$ is consistent with K.

Proposition 5 Let $\mathcal{G}|K$ be a specification and let $(\mathcal{C}, \| \; \|)$ be an interpretation of $\mathcal{G}|K$. Then $(\mathcal{C}, \| \; \|)$ is a $\mathcal{C}$-model of $\mathcal{G}|K$ if and only if $(\mathcal{C}, \| \; \|)$ is consistent with K and any two equivalent and meaningful $\mathcal{G}|K$-computations have the same meaning with repect to $\| \; \|$.

Proof The 'only if' part is expressed by corollary 1 of Theorem 2. Conversely, assume that $\| \; \|$ is consistent and that any two equivalent and meaningful $\mathcal{G}|K$-computations have the same meaning. Let $\pi \leq e$ be an edge constraint in K. We can write $e \overset{*}{\leq}_K e$ by reflexivity, and then by addition rules, $e + \pi$ is a $\mathcal{G}|K$-computation and $e + \pi \overset{*}{\leq}_K e \overset{*}{\leq}_K e + \pi$. Thus, $e + \pi \equiv_K e$ and we have $\|e + \pi\|_m = \|e\|_m$. That is $\|e\|_m \oplus \|\pi\|_m = \|e\|_m$ or, equivalently, $\|\pi\|_m \Rightarrow \|e\|_m$. This means that $\| \; \|$ satisfies $\pi \leq e$. This completes the proof. ◇

Definition 10 Let $G|K$ be a specification. The *derivative* of a G-computation φ, denoted $\partial\varphi$, is defined recursively as follows:

- $\partial(id(A)) = zero(A, A)$ and $\partial(zero(A, B)) = zero(A, B)$,
- if φ is an edge of G then if φ is not the head of any constraint in K then $\partial\varphi$ is $zero(src(e), tgt(e))$ else $\partial\varphi$ is the sum of the bodies of all constraints in K with head φ,
- if $\varphi = (\varphi_1)$ then $\partial(\varphi) = (\partial\varphi_1)$,
- if $\varphi = \varphi_1.\varphi_2$ then $\partial(\varphi) = \varphi_1.\partial\varphi_2+(\partial\varphi_1).\varphi_2+\partial\varphi_1.\partial\varphi_2$, and
- if $\varphi = \varphi_1+\varphi_2$ then $\partial(\varphi) = \partial\varphi_1+\partial\varphi_2$. ◇

Now, for every $i \geq 0$ and for every computation φ, define $\partial^{(i)}\varphi$ as follows:

$\partial^{(0)}\varphi = \varphi$, $\partial^{(1)}\varphi = \partial\varphi$, $\partial^{(i)}\varphi = \partial(\partial^{(i-1)}\varphi)$

Lemma 1 If φ is a $G|K$-computation then so is $\partial^{(i)}\varphi$ and $\partial^{(i)}\varphi \overset{*}{\leq}_K \varphi$, for all $i \geq 0$.

Proof It is enough to prove the proposition for $i = 1$. We prove it by structural induction. If $\partial\varphi = zero(src(\varphi), tgt(\varphi))$ then, by the zero rule, $\partial\varphi$ is a $G|K$-computation and $\partial\varphi \overset{*}{\leq}_K \varphi$. If φ is an edge of G and $\partial\varphi \neq zero(src(\varphi), tgt(\varphi))$ then φ is the head of some constraint, $\partial\varphi$ is obtained by the addition rule and $\partial\varphi \overset{*}{\leq}_K \varphi$. Now, let φ be a derived $G|K$-computation such that, for each component e of φ, ∂e is a $G|K$-computation and $\partial e \overset{*}{\leq}_K e$ (structural induction hypothesis). If φ has the form $u.v$ then ∂u and∂v are $G|K$-computations and $\partial u \overset{*}{\leq}_K u$ and $\partial v \overset{*}{\leq}_K v$ by the induction hypothesis. So by augmentation $\partial u.v \overset{*}{\leq}_K u.v$, $u.\partial v \overset{*}{\leq}_K u.v$ and $\partial u.\partial v \overset{*}{\leq}_K u.v$. Thus, $\partial\varphi = u.\partial v + \partial u.v + \partial u.\partial v$ is a $G|K$-computation and $\partial\varphi \overset{*}{\leq}_K u.v = \varphi$, by the addition rule. Similarly, if φ has the form $u+v$, where ∂u and ∂v are $G|K$-computations satisfying $\partial u \overset{*}{\leq}_K u$ and $\partial v \overset{*}{\leq}_K v$, then $\partial u \overset{*}{\leq}_K u \overset{*}{\leq}_K \varphi$ and $\partial v \overset{*}{\leq}_K v \overset{*}{\leq}_K \varphi$, so$\partial\varphi = \partial u+\partial v$ is a $G|K$-computation and $\partial\varphi \overset{*}{\leq}_K u.v = \varphi$ by addition rules. This completes the proof. ◇

Definition 11 An edge e in a specification $G|K$ is said to be *non recursive* if there is an integer $k \geq 0$ such that $\partial^{(k+1)}e = zero(src(e), tgt(e))$; otherwise e is called a *recursive* edge. A specification with no recursive edges is called *acyclic*, otherwise is said to be *cyclic*. For a non recursive edge e the smallest integer i_e satisfying $\partial^{(i_e+1)}e = zero(src(e), tgt(e))$; is called the *depth* of e, and is denoted *depth(e)*. ◇

We stress that acyclicity refers to constraints K and not to graph $\mathcal{G}$. The way edges in K depend on one another, may be represented as a graph $\mathcal{K}$ called the *dependency graph*. This graph is defined as follows:

- the nodes of $\mathcal{K}$ are the edges of $\mathcal{G}$, and
- there is an edge from f to e in $\mathcal{K}$ whenever e is the head of a constraint in $\mathcal{K}$

and f is the body of that constraint.

Note that for every edge e of $\mathcal{G}$ the edges which appear in ∂e are the immediate sons of e in $\mathcal{K}$. In particular, leaves of $\mathcal{K}$ are those edges of $\mathcal{G}$, which appear only in the body of constraints. This allows to state the following proposutions and lemmas which express important properties of cyclic specifications.

Proposition 6 $\mathcal{G}|K$ is cyclic if and only if its dependency graph is cyclic.

Proof If the dependency graph $\mathcal{K}$ of $\mathcal{G}|K$ is cyclic then there is an edge e of $\mathcal{G}$ (i.e. a node of $\mathcal{K}$) such that $\partial^{(i)}e$ contains e for some i (in fact i is the length of a cycle in $\mathcal{K}$ traversing e). This means that $\partial^{(k)}e \neq zero(src(e),\ tgt(e))$ for every k, that is $\mathcal{G}|K$ is cyclic. Conversely, if $\mathcal{K}$ is acyclic then an edge e of $\mathcal{G}$ cannot be a son of e in the graph $\mathcal{K}$. Thus, for some k, $\partial^{(k)}e$ is formed by leaves of $\mathcal{K}$. This means that $\partial^{(k+1)}e = zero(src(e),\ tgt(e))$. This completes the proof. ◇

Lemma 2 If $\mathcal{G}|K$ is cyclic then there is at least one edge e which is head of a constraint in K, and two paths l_e and r_e such that $K \vdash l_e.e.r_e \overset{*}{\leq}_K e$. Moreover, l_e and r_e are cycles of $\mathcal{G}$, but l_e and r_e are not necessarily unique nor with positive length.

Proof Note that when we write $K \vdash l_e.e\ r_e \leq e$, the paths l_e and r_e are cycles in $\mathcal{G}$. Indeed, $l_e.e.r_e$ and e are parallel so $src(l_e) = src(l_e.e.r_e) = src(e)$ and $tgt(l_e) = src(e)$ so l_e is a cycle. Similarly r_e is a cycle. Now, assume that the specification is cyclic. From Lemma 2, there are edges $e,\ e_1,\ ...,\ e_n$ in $\mathcal{G}$ such that the constraints $l_0.e.r_0 \leq e_1,\ l_i.e_i.r_i \leq e_{i+1}\ (i = 1,\ 2,\ ...,\ n-1)$ and $l_n.e_n.r_n \leq e$ are in K, where l_i or r_i may be the empty path for $i = 0,\ 1,\ ...,\ n$. It follows that $K \vdash l_e e.r_e \leq e$, where $r_e = r_0 r_1\ ...r_{n-1}r_n$ and $l_e = l_n l_{n-1}\ ...l_1 l_0$. Of course, the paths l_e and r_e depend on the given cycle c but the last equalities show how to construct them from K. ◇

Now, if l_e and r_e in the above lemma are unique for every recursive edge e then $\mathcal{G}|K$ is said to be *linearly cyclic*. Note that uniqueness of l_e and r_e means that they are formed by non recursive edges. Now the notion of depth can be generalized for recursive edges.

Definition 12 The *depth* of a recursive edge e is the smallest integer $l_e > 1$ such that $\partial^{(l_e)}e$ contains e. ◇

Lemma 3 In a linear cyclic specification $\mathcal{G}|K$, for every recursive edge e, and for all integers n and k, if $1<n$ and $0\leq k<depth(e)$ then

$$\partial^{(n\times depth(e)+k)}e = l_e^n.\partial^{(k)}e.r_e^n \qquad (*)$$

Proof We shall prove $(*)$ by induction on k and n. From the proof of Lemma 2 we can see that $\partial^{(depth(e))}e = l_e.e.r_e$. This means that the equality $(*)$ stands for $k=0$ and $n=1$. Assume that for a given n≥1, equality $(*)$ stands for $0\leq k<depth(e)-1$ (induction hyopothesis for k). As l_e and r_e are formed by non recursive edges, we can write

$$\partial^{(n\times depth(e))+k+1)}e = \partial(l_e^n.\partial^{(k)}e.r_e^n) = l_e^n.\partial^{(k+1)}e.r_e^n.$$

Similarly, assume that for a given k, $0\leq k\leq depth(e)$, equality $(*)$ stands for $n\geq 1$ (induction hypothesis for n). We can then write,

$$\begin{aligned}\partial^{((n+1)\times depth(e)+k)}e &= \partial^{(deph(e))}(\partial^{(n\times depth(e)+k)}) = \partial^{(deph(e))}(l_e^n.\partial^{(k)}e.r_e^n)\\ &= l_e^n.\partial^{(depth(e)+k)}.r_e^n = l_e^n.(l_e.\partial^{(k)}e.r_e).r_e^n = l_e^{n+1}.\partial^{(k)}e.r_e^{n+1}.\end{aligned}$$

This completes the proof. ◇

Lemma 4 For every edge e in an acyclic or in a linearly cyclic specification and for every natural number $i\geq 0$, the $\mathcal{G}$-computation

$$T_i(e) = \sum_{k=0}^{i}\partial^{(k)}e$$

is a $\mathcal{G}|K$-computation and $T_i(e)\overset{*}{\leq}_K e$. Moreover,

$$T_i(e) = T_{i-1}(e) + l_e^{\lambda_i(e)}.\partial^{\eta_i(e)}e.r_e^{\lambda_i(e)}$$

where $\lambda_i(e) = i \; div \; depth(e)$, $\eta_i(e) = i \; mod \; depth(e)$.

Proof The first part is an immediate consequence of Lemma 1 and addition rule. The second part is a result of Lemma 3. ◇

An important remark at this point is that, even for $i\geq depth(e)$, the $\mathcal{G}|K$-computation $T_i(e)$ uses only $\partial^{(k)}e$, for $0\leq i<depth(e)$. Moreover, since the specification is assumed to be linearly cyclic, the cycles l_e and r_e and all $\partial^{(k)}e$, $0<k<depth(e)$, do not contain any recursive edge. ◇

Now, we shall prove that every consistent interpretation can be embedded in a model, and that this model can be constructed by a fixpoint operator. To this end, let us first define an ordering on the class of all $\mathcal{C}$-interpretations of a given specification $\mathcal{G}|K$ as follows:

- $\| \, \| \leq \| \, \|'$ if and only if 1) $\|A\| = \|A\|'$ for every node A,and 2) $\|e\| \leq \|e\|'$ for every edge e.

Definition 13 Given a cycle $u : X \to X$ in a semantic universe, we say that u is *finitely representable* if there is integer n such that $\{u, u^2, \ldots, u^{n-1}\}$ is consistent and $u^i \leq u + u^2 + \ldots + u^{n-1}$ for every $i \geq n$. The least n satisfying this inequality is called the ***height*** of u and is denoted $height(u)$. ◇

For example, every finite multivalued function is finitely representable (see [LeSp93] for more details).

Let $\mathcal{G}|K$ be acyclic or linearly cyclic specification. Let $\mathcal{C}$ be a semantic universe in which every cycle is finitely representable, and let $(\mathcal{C}, \| \ \|)$ be a given interpretation of $\mathcal{G}$ which is consistent with K. Let $\mathcal{I}$ be the set of all $\mathcal{C}$-interpretations I which are consistent with K and $\| \ \| \leq I$. Each I in $\mathcal{I}$ can be extended to all $\mathcal{G}|K$-computations. For simplicity, let us denote the extension of I also by I. Then $I(e+\partial e)$ exists and is equal to $I(e) \oplus I(\partial e)$, for every edge e in $\mathcal{G}$. Now, we define $T : \mathcal{I} \to \mathcal{I}$ by:

- $T(I)(A) = I(A)$, for every node A, and
- $T(I)(e) = I(e) \oplus I(\partial e)$, for every edge e.

Theorem 4 With the above hypotheses, $\mathcal{I}$ is a wposet with least element $\| \ \|$, and the function T is continuous and has a least fixpoint $\| \ \|^S$ which is a model of $\mathcal{G}|K$.

Proof From the definition of the ordering of $\mathcal{I}$, it is clear that $\| \ \|$ is the least element of $\mathcal{I}$. Moreover, $\mathcal{I}$ is a wposet, because the ordering of $\mathcal{I}$ is defined pointwise and $\mathcal{C}$ is a semantic universe. Let I_1, I_2 be in $\mathcal{I}$ and assume that $lub(I_1, I_2)$ exists in $\mathcal{I}$. This means that for every edge e in $\mathcal{G}$, $I_1(e) \oplus I_2(e)$ exists in $\mathcal{C}$. Thus we have $T(lub(I_1,I_2))(e) = I_1(e) \oplus I_2(e) \oplus I_1(\partial e) \oplus I_2(\partial e) = T(I_1)(e) \oplus T(I_2)(e) = lub(T(I_1), T(I_2))(e)$. That is, the function T is continuous.

As usual, the least fixpoint is obtained by iterating T on the least element of $\mathcal{I}$. That is, for every edge e, $\| e \|^S$ is the limit of the sequence $(\| e \|, \| e \| \oplus \| \partial e \|, \| e \| \oplus \| \partial e \| \oplus \| \partial^2 e \|, \ldots)$. Using the above notation we can write $\| e \|^S = lim_i \| T_i(e) \|$.

We must show that $\| e \|^S$ can be computed in a finite number of steps. i.e. we must show that $\| e \|^S$ is bouded by an arrow of $\mathcal{C}$. Using the above Lemmas, if e is not recursive then $\| e \|^S = \| e \| \oplus \| \partial e \| \ldots \oplus \| \partial^{(depth(e)-1)} e \|$.
Otherwise, consider l_e and r_e as in Lemma 2 and denote

$h_e = max(height(\| l_e \|), height(\| r_e \|))$,
$L_e = \| l_e \| \oplus \| l_e^2 \| \oplus \ldots \oplus \| l_e^{n-1} \|$, and
$R_e = \| r_e \| \oplus \| r_e^2 \| \oplus \ldots \oplus \| r_e^{n-1} \|$.

Now, for all $n>0$ and all k, $0 \leq k < depth(e)$ we can write

$$\| l_e \|^i \| \partial^{(k)} e \| \| r_e \|^i \Rightarrow L_e \| \partial^{(k)} e \| R_e \text{ for every } i>0. \qquad (*)$$

From Lemma 4 we can write

$$T_i(e) = \sum_{k=0}^{depth(e)-1} \partial^{(k)} e + \sum_{l=1}^{\lambda_i(e)} \sum_{k=0}^{depth(e)-1} l_e^{\,l} . \partial^{(k)} e . r_e^{\,l}$$

$$+ \sum_{k=0}^{\eta_i(e)} l_e^{\lambda_i(e)+1} . \partial^{(k)} e . r_e^{\lambda_i(e)+1} \qquad (**)$$

Now, from (*) and (**) we can write
$\| e \|^{s} \leq \bigoplus_{k=0}^{depth(e)-1} \| \partial^{(k)} e \| \oplus R_{e}(\bigoplus_{k=0}^{depth(e)-1} \| \partial^{(k)} e \|) L_{e}$.

This inequality prove that $\| e \|$ is bounded which completes the proof. ◇

In [LeSp93] we show that $\| \, \|^{s}$ can be computed by an effective algorithm, in the semantic universe $\mathcal{mSet}$.

Fixpoint Semantics Using Theorem 1, the model $\| \, \|^{s}$ can be extended to $c\mathcal{G}|K$ as a semantic function $\| \, \|^{s}_{m}$. This semantic function provides meaning for all computations under the constraints of K. Let us consider the functor $U : \mathcal{Sem} \rightarrow \mathcal{Spec}$ seen earlier and let us consider the full sub-category of the category of U-objects under $(\mathcal{G}|K, \| \, \|)$, whose object are consistent interpretations. The pair $(c\mathcal{G}|K, \| \, \|^{s}_{m})$ is an initial object in this sub-category, on the one hand, and is obtained by a fixpoint operator, on the other hand. We call $(c\mathcal{G}|K, \| \, \|^{s}_{m})$ *the fixpoint semantics* of $(\mathcal{G}|K, \| \, \|)$.

6 CONCLUSION AND FURTHER RESEARCH

We have considered computations over a graph as special subgraphs, and we have introduced a set of inference rules for deducing new constraints and new computations from old. These notions received interpretations in an enriched category. We proved that the inference rules are sound and complete.
One aspect of our approach that has not been developped here is its possibility of incremental specification. This aspect, is of particular interest for modular specification, especially in databases specifictaion. We are currently investigating this research direction.

Acknowledgements: This paper would not have been written in its present form without several valuable discussions that S.K. Lellahi had with Christian Lair. The authors would like to thank him. The authors would also like to thank the referees for their comments that have contributed to improve the paper.

REFERENCES

[BaWe90] M. Barr, C. Wells, Category for Computing Science (Prentice Hall, 1990).
[CoMe90] M.P. Consens, A. Mendelzon, Graphlog : A Visual Formalism for Real Life Recursion, in : Proc. ACM-SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (1990) 404-416.
[GPV90] M. Guyssen, J. Paredaens, D. Van Gusht, A Graph-Oriented Object Database Model, Proc. ACM-SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (1990) 417-424.
[Gray74] J.W. Gray, Formal Category Theory : Adjointness for 2-categories, Lecture Notes in Mathematics, vol 391 (Springer, Berlin, 1974).
[Gray89] J.W. Gray, The Theory of Sketches as a Model for Algebraic Semantics, In : Category in Computer Science and Logic, Contemporary Mathematics vol. 92 (American Math. Society, 1989) 109-135
[GuSc90] C.A. Gunter, D.S. Scott, Semantic Domains, In : J.van Leeuwen ed., Handbook of Theoretical Computer Science vol. B (Elsevier Science Publishers, 1990) 635-674.
[Güti93] R.H. Güting, Second-Order Signature : A Tool for Specifying Data Model, Query Processing, and Optimization, Sigmod's 93, Proc. of the ACM-Sigmod International Conference on Management of Data. Washington DC may 26-28 1993, edited by P. Buneman and S. Jajodia, acm Press.
[KuVa93] G.M. Kuper, M.Y. Vardi, The Logical Model, ACM transactions on Database Systems, vol. 18, No. 3, September 1993, pages 379-413.
[Kelly 82] G.M. Kelly, Basic Concepts of Enriched Category Theory, vol. 64 of London Mathematical Society Lecture Note Series (Cambridge University Press, 1982).
[Lell93] S.K. Lellahi, Une formalisation Algébrique pour la Modelisation des Données, 5ème journées du LIPN, 6-7 septembre 1993, Univ. Paris13, France.
[LeSp91] S.K. Lellahi, N. Spyratos, Towards a Categorical Data Model Supporting Structured Objects and Inheritance, in : proc. Next Generation Information System Technology, Lecture Notes in Computer Science, vol.504, (Springer, Berlin, 1991) 86-105.
[LeSp92] S.K. Lellahi, N. Spyratos, Categorical Modelling of Database Concepts, Esprit BRA Project 3070, Technical Report Series, FIDE/92/38; University of Glasgow, Dept. of Computer Science, also Research Report No 746, LRI, Univ. Paris XI, Orsay, 1992.
[LeSp93] S.K. Lellahi, N. Spyratos, An Algebraic Semantics for Data Modelling under Constraints, Research Report No 93-05, LIPN, Univ. Paris-Nord, Villetaneuse, France, 1993.

[MacL71] S. Mac Lane, Categories for the Working Mathematician (Springer, Berlin, 1971)
[PoWe92] A.J. Power, C. Wells, A formalism for the specification of essentially-algebraic structures in 2-categories, Mathematical Structures in Computer Science Vol. 2 (1992) 1-28.
[TGP91] C. Tuijn, M. Guyssen, J. Paredaens, A Categorical Approach to Object-Oriented Data Modelling, Research Report No 91-09, University of Antwerp (UIA), Belgium.
[TuGu92] C. Tuijn, M. Guyssen, Views and Decomposition of Databases from a Categorical Perspective, Proc. International Conference on Database Theory ICDT 92, Lecture Notes in Computer Science vol. 646 (Springer, Berlin, 1992) 99-111.
[Ullm88] J.D. Ullman, Principles of Database and Knowledge-Base Systems, vol. I (Computer Science Press, 1988).
[VaVa92] J. Van den Bussche, D. Van Gucht, A Hierarchy of Faithful Set Creation in Pure OODB's, in : Proc. of International Conference on Database Theory ICDT, Lecture Notes in Computer Science vol. 646 (Springer-Verlag, 1992) 326-340.
[Wedd92] G.E. Weddell, Resoning about Functional Dependencies Generalized for Semantic Data Models, ACM Transactions on Database Systems vol 17, No 1 (1992) 32-64.