SEYED-KAZEM LELLAHI
NICOLAS SPYRATOS

Deduction over graphs under constraints : a soundness and completeness theorem

Diagrammes, tome 29 (1993), exp. n° 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,
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/multivalued 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 $V$-category, and a semantic function is a $V$-functor [Gray74], [Kelly82], when $V$ 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 $pse$, 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 \( \text{lub} A \). 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 \( \text{lub} \). 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 \( \text{lub}(f(A)) = f(\text{lub} A) \). 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 \( f(\text{lub}(a, b)) = \text{lub}(fa, fb) \). 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 \( \text{src}(u) \), and \( B \) the target of \( u \), denoted \( \text{tgt}(u) \). Now, the category \( \mathcal{C} \) is a semantic universe if:

- for all objects \( A \) and \( B \) the set \( \mathcal{G}(A, B) \) of arrows from \( A \) to \( B \) is a wposet; the zero of this wposet is denoted \( 0(A, B) \),
- \( 0(\text{tgt}(u), A) = 0(\text{src}(u), A) \), and \( u0(A, \text{src}(u)) = 0(A, \text{tgt}(u)) \), for every arrow \( u \) and every object \( A \),
- for all arrows \( x, y, z \) and \( t \) as in the following configuration

\[
\begin{array}{ccc}
H & \xrightarrow{x} & I \\
\downarrow & & \downarrow \\
J & \xrightarrow{y} & L
\end{array}
\]

if \( (y, z) \) is a consistent pair, then so are \( (yx, zx) \) and \( (ty, tz) \) and we have:

- \( \text{lub}(yx, zx) = \text{lub}(y, z)x \) (left continuity)
- \( \text{lub}(ty, tz) = t \text{lub}(y, z) \) (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 '\( \Rightarrow \)' 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 $\mathbb{Set}$ of sets and partial functions with the usual ordering on partial functions,
- the category $\mathbb{mSet}$ of sets and multivalued functions with pointwise inclusion,
- the category $\mathbb{Brel}$ of sets and binary relations with the usual ordering on binary relations, and
- the category $\mathbb{Op}$ 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 `$\leq$' such that:

- there is an element $e$ of $M$, called zero, satisfying $e \cdot m = m = me = e$ 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 $\text{lub}(m_1, m_2)$ exists, and
- if $\text{lub}(m_1, m_2)$ exists then so does $\text{lub}(m_3m_1, m_3m_2)$, for every $m_3$ in $M$, and

$$
\text{lub}(m_1, m_2)m_3 = \text{lub}(m_1m_2, m_3) \text{ and }
\text{m_3lub}(m_1, m_2) = \text{lub}(m_3m_1, m_3m_2)
$$

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 stated 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 `$\cdot$' of $\mathcal{C}$, whose objects are all elements of $\mathcal{C}_0$ and whose arrows, called 2-cells, are pairs of parallel arrows $f$, $g$ such that $f \cdot g$. 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 $(f \cdot g)(g \cdot h) = (f \cdot h)$ corresponds to: if $(fxg$ and $gsh)$ then $fsh$. Moreover, in this category the coproduct of two objects $f$ and $g$, when it exists, is $\text{lub}(f, g)$. So, we use $f \cdot g$ as an alternative notation for $\text{lub}(f, g)$.

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 $f \cdot g$ 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 $f$. 

1.8 4
and $g$. The horizontal composition is the composition in the category $C \times C$, that is $(h \circ k) \circ (f \circ g) = (hf \circ kg)$ if and only if $\text{src}(h) = \text{tgt}(f)$ and $\text{src}(k) = \text{tgt}(g)$. Thus, in a semantic universe, any pair $\alpha = (f, g)$ of arrows such that $f \circ g$, may be seen as a vertical or a horizontal arrow. We adopt the notation of [PoWe92] and we write $\alpha : f \Rightarrow g$ when $\alpha$ is seen as a vertical arrow, and we write $\alpha : f \Rightarrow g : A \to B$ when $\alpha$ 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 $C$, for all configurations of objects, arrows and cells as in Figure 1, the following holds:

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

![Figure 1](image)

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

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

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

- $\mathcal{F}(0(A, B)) = 0(1A, 1B)$ for all objects $A$ and $B$, and
- for all parallel and consistent arrows $a$ and $b$ in $C$, the arrows $\mathcal{F}(a)$ and $\mathcal{F}(b)$ are consistent in $C'$ and $\mathcal{F}(a \circ b) = \mathcal{F}(a) \circ \mathcal{F}(b)$.

It is clear that such a functor is monotone. We denote by $\text{Sem}$ 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 $\text{src}(e)$, and the node $j$ is called the...
target of \( e \), denoted \( \text{tgt}(e) \). We assume that edges have distinct labels. A sequence \( (e_1, e_2, \ldots, e_n) \) of edges in a graph \( G \), is called a \( G \)-path of length \( n \), if \( \text{tgt}(e_i) = \text{src}(e_{i+1}) \) for every \( i, 1 \leq i \leq n-1 \). We denote such a path by \( e_1 e_2 \ldots e_n \). Morphisms of graphs are defined as usual, and the category of locally small graphs is denoted by \( \mathcal{G} \).

**Computations**

Given a graph \( G \), we apply the recursive inference rules, of Figure 2 on \( G \) to obtain what we shall call \( G \)-computations. In these rules, a computation \( \varphi \) from \( A \) to \( B \) is denoted \( \varphi : A \rightarrow B \), and

\[
\begin{array}{c}
\varphi \\
\Psi
\end{array}
\]

means that "in any context, if we have already a premise \( \varphi \) then we can construct the consequence \( \Psi \)."

| Inclusion | \( e \) (edge) | \( e : \text{src}(e) \rightarrow \text{tgt}(e) \) |
|-----------|-----------------|
| Identity  | \( A \) (node)  | \( \text{id}(A) : A \rightarrow A \) |
| Product   | \( \varphi : A \rightarrow B \) \( \psi : B \rightarrow C \) |
| Addition  | \( \varphi : A \rightarrow B \) \( \psi : A \rightarrow B \) |
| Parenthesis | \( \varphi : A \rightarrow B \) |
| zero      | \( A \) \( B \) (nodes) | \( \text{zero}(A,B) : A \rightarrow B \) |

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

**Equivalent Computations**

Let \( \rho_1, \rho_2, \ldots, \rho_n \) be parallel \( G \)-computations with source \( A \) and target \( B \). We say the \( G \)-computation \( \rho_1 + \rho_2 + \ldots + \rho_n \) is a \( G \)-expression if each \( \rho_i \) is either a \( G \)-path
or the $G$-computation $\text{zero}(A, B)$. In particular, every $G$-path is a $G$-expression. We shall see that expressions are canonical forms of computations.

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

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

The particularity of the set $\mathcal{A}(e)$ is that it is a set of parallel paths, so it contains at most one computation of the form $\text{id}(A)$ and no computation of the form $\text{zero}(A, B)$.

We can extend the function $\mathcal{A}$ to all $G$-computations using the following rules, where the extension is denoted $\mathcal{A}^*$.

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

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

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

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

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

- if $\varphi \equiv \psi$ then $\text{src}(\varphi) = \text{src}(\psi)$ and $\text{tgt}(\varphi) = \text{tgt}(\psi)$, for all parallel $G$-computations $\varphi$ and $\psi$,
- if $\varphi \equiv \psi$ then $\varphi + \xi \equiv \psi + \xi$, for all parallel $G$-computations $\varphi$ and $\psi$ and $\xi$,
- if $\varphi \equiv \psi$ then $\varphi.\xi \equiv \psi.\xi$ and $\varphi.\psi = \rho.\psi$, for all parallel $G$-computations $\varphi$, $\psi$, and $\rho.\psi$ are well defined.

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

- $[\text{id}(A)] = [\text{id}(A)]$, $[\text{zero}(A, B)] = [\text{zero}(A, B)]$,
- $[\text{src}(\varphi)] = \text{src}(\varphi)$, $[\text{tgt}(\varphi)] = \text{tgt}(\varphi)$,
- $[\varphi.\psi] = [\varphi.\psi]$ and
- $[\varphi + \psi] = [\varphi + \psi]$.
Moreover, equivalence classes of $G$-computations can be ordered by:

$\varphi \leq \varphi'$ if and only if $A^*(\varphi) \subseteq A^*(\varphi')$.

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

For simplicity we denote the operations $[\text{id}], [\text{zero}], [\text{src}], [\text{tgt}], [\text{.}], \text{and } +$ by $\text{id}$, $\text{zero}$, $\text{src}$, $\text{tgt}$, $\text{.}$, and $+$, respectively.

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

**Proof** The objects of $cG$ are the nodes of $G$, the arrows of $cG$ are all equivalence classes of $G$-computations, and the composition of arrows is defined by:

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

The rest of the proof is obvious, but tedious, using the properties of union on sets and the definitions of $A^*$ and $\equiv$.

The function that associates each graph $G$ with the semantic universe $cG$ defines a functor $c$ from the category $\mathbf{G}$ to the category $\text{Sem}$. However, $c$ is not a left free adjoint to the forgetful functor $U : \text{Sem} \rightarrow \mathbf{G}$. That is, there may exist a graph morphism $i$ from $G$ to a semantic universe $\mathcal{C}$ which cannot be freely extended to $cG$ as a semantic function. Indeed for a computation $\varphi$ 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 $cG$.

**The Meaning of Computations**

Recall that the ordering of a general semantic universe is denoted '⇒' and its $\text{lub}$ operation is denoted '$\ast$'.

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

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

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

**Definition 5**

Given an interpretation \( \mathcal{I} \) of a graph \( \mathcal{G} \),

- every \( \mathcal{G} \)-node or \( \mathcal{G} \)-edge \( x \) is meaningful and \( \mathcal{I}_m(x) = \mathcal{I}(x) \),
- \( \text{id}(A) \) is meaningful and \( \mathcal{I}_m(\text{id}(A)) = \text{id}(\mathcal{I}(A)) \), for every node \( A \),
- every \( \mathcal{G} \)-path \( p = e_1 e_2 ... e_n \) of positive length is meaningful and \( \mathcal{I}_m(p) = \mathcal{I}_m(e_1) \mathcal{I}_m(e_2) ... \mathcal{I}_m(e_n) \),
- \( \text{zero}(A,B) \) is meaningful and \( \mathcal{I}_m(\text{zero}(A,B)) = \mathcal{I}(A) \mathcal{I}(B) \), for all nodes \( A \) and \( B \), and
- a \( \mathcal{G} \)-expression \( e = p_1 p_2^* ... p_n^* \) is meaningful if \( \mathcal{I}_m(p_1) \mathcal{I}_m(p_2) ... \mathcal{I}_m(p_n) \) exists in \( \mathcal{G} \), and then \( \mathcal{I}_m(e) = \mathcal{I}_m(p_1) \mathcal{I}_m(p_2) ... \mathcal{I}_m(p_n) \).

Note that when the meaning function \( \mathcal{I}_m \) is applied to a path \( p = e_1 e_2 ... e_n \) 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 \( \varphi \) be a \( \mathcal{G} \)-computation, let \( \sigma^*(\varphi) = \{ p_1, p_2, ..., p_n \} \), 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 \( \sigma^* \) stands for a permutation of the indices 1, 2, ..., \( n \) in \( e_\varphi \). Since \( e_\varphi \) and \( e_{\sigma\varphi} \) are equivalent, if \( \mathcal{I}_m(e_\varphi) \) exists then \( \mathcal{I}_m(e_{\sigma\varphi}) \) exists and \( \mathcal{I}_m(e_{\sigma\varphi}) = \mathcal{I}_m(e_\varphi) \). The expression \( e_\varphi \) is called the canonical form of \( \varphi \). Now, we can define the meaning of a computation as follows:

**Definition 6**

A computation \( \varphi \) is said to be **meaningful**, with respect to an interpretation \( \mathcal{I} \), if \( \mathcal{I}_m(e_\varphi) \) is defined; otherwise \( \varphi \) is said to be **meaningless**. If \( \varphi \) is meaningful, then \( \mathcal{I}_m(e_\varphi) \) is called the **meaning** of \( \varphi \).

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

It is not difficult to prove that:

- if \( \varphi \) is meaningful and \( \varphi = \varphi' \) then \( \varphi' \) is meaningful and \( \mathcal{I}_m(\varphi) = \mathcal{I}_m(\varphi') \),
- if \( \varphi \) and \( \varphi' \) are meaningful and \( \varphi \varphi' \) is defined then \( \varphi \varphi' \) is meaningful and \( \mathcal{I}_m(\varphi \varphi') = \mathcal{I}_m(\varphi) \mathcal{I}_m(\varphi') \), and
- if \( \varphi \) and \( \varphi' \) are meaningful and if \( \mathcal{I}_m(\varphi) \mathcal{I}_m(\varphi') \) is defined then \( \varphi \varphi' \) is meaningful and \( \mathcal{I}_m(\varphi \varphi') = \mathcal{I}_m(\varphi) \mathcal{I}_m(\varphi') \).

Thus we can state...
Proposition 4  Let $G$ be a graph. For every interpretation $(G, \| \|)$ of $G$, the equivalence classes of meaningful $G$-computations, with respect to $\| \|$, form a semantic universe $mG$ which is a sub-semantic universe of $cG$. Moreover, $\| \|_m$ is a semantic function from $mG$ to $G$.

We shall prove that the semantic universe $mG$ may be seen as a free construction over $(G, \| \|)$. 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 $\mathcal{I}$ such that the following holds:

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

It is easy to prove that the semantic function $\| \|_m$ is also a morphism of $\mathcal{S}em1$. A morphism $\mathcal{I} : \mathcal{C} \to \mathcal{C}'$ of $\mathcal{S}em1$ is not necessarily a semantic function nor a monotonic function. However, if $a \mathcal{I} b$ and $\mathcal{I}a \mathcal{I}b$ exists (for instance, if $\mathcal{I}a$ and $\mathcal{I}b$ are consistent) then $\mathcal{I}a \mathcal{I}b$.

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

Proof  Let $(\mathcal{G}, \| \|)$ be an object of $\mathcal{S}em1/G$ and let $F : \mathcal{G} \to V(\mathcal{G})$ be an arrow of $\mathcal{S}em1/G$ from $(\mathcal{G}, \| \|)$ to $V(\mathcal{G})$. Let $\varphi$ be a meaningful computation with canonical form $e_\varphi = p_1 + p_2 + \ldots + p_n$. Thus, $\| \|_m = \| \|_m \mathcal{C} = \| p_1 \|_m \star \| e_2 \|_m \star \ldots \star \| p_n \|_m$ exists in $\mathcal{C}$. If $p_i = e_{i_1} e_{i_2} \ldots e_{i_{n_i}}$, define $\varphi^* = F(e_{i_1}) \ldots F(e_{i_2}) F(e_{i_1})$. As $V(\mathcal{G})F = \| \|$ we have $V(\mathcal{G})p_1^* = \| e_{i_1} \| \ldots \| e_{i_2} \| \ldots \| e_{i_1} \| = \| p_1 \|_m$. So $\| \|_m = V(\mathcal{G})p_1^* \star \ldots \star V(\mathcal{G})p_n^* = Tp_1^* \star \ldots \star Tp_n^*$. We conclude that the arrow $\varphi^* = p_1^* \star \ldots \star p_n^*$ exists in $\mathcal{C}$ and $T(\varphi^*) = Tp_1^* \star \ldots \star Tp_n^*$ (because $T$ is a morphism of $\mathcal{S}em1$). If we define $H(\varphi) = \varphi^*$ then $H$ is an arrow of $\mathcal{S}em1/G$ as it satisfies $TH = \| \|_m$. Moreover, the equation $V(H)\eta_G = F$ is satisfied in
DEDUCTION OVER GRAPHS UNDER CONSTRAINTS...

\( t^j / G \) and \( H \) is the unique arrow of \( \text{Sem} \mathcal{G} / \mathcal{C} \) satisfying this equation. This completes the proof. \( \diamond \)

![Figure 3](image)

**Initial Semantics** An immediate consequence of this adjunction is the following: Consider the category of \( \mathcal{V} \)-objects under \((\mathcal{G}, \mathcal{I})\) [MacL71]; an object of this category is a pair \((F, (\mathcal{G}', T))\) where \((\mathcal{G}', T)\) is an object of \( \text{Sem} \mathcal{G} / \mathcal{C} \) and \( F : (\mathcal{G}, \mathcal{I}) \rightarrow \mathcal{V} / \mathcal{C}(\mathcal{G}', T) \) is an arrow of \( \mathcal{G} / \mathcal{C} \), an arrow from \((F, \mathcal{G'}, T)\) to \((F', \mathcal{G}''', T')\) is a semantic function \( \eta \) satisfying \( F(\mathcal{V}(\eta)) = F' \). In this category the object \((\eta_{\mathcal{G}}, (\mathcal{G}, \mathcal{I}))\) is an initial object. Therefore, we may call \((\mathcal{G}, \mathcal{I})\) the initial semantics of \((\mathcal{G}, \mathcal{I})\).

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 \( \mathcal{G} \) is any statement of the form \( \varphi \& \varphi' \), where \( \varphi \) and \( \varphi' \) are parallel \( \mathcal{G} \)-computations and \( \varphi = \varphi' \). A specification \( \mathcal{G} | \mathcal{K} \) consists of a graph \( \mathcal{G} \) and a set \( \mathcal{K} \) of constraints over \( \mathcal{G} \) (usually \( \mathcal{G} \) and \( \mathcal{K} \) are finite). \( \diamond \)

Intuitively, a specification \( \mathcal{G} | \mathcal{K} \) is a graph \( \mathcal{G} \) under constraints \( \mathcal{K} \), so \( \mathcal{G} | \mathcal{K} \) can be read "\( \mathcal{G} \) such that \( \mathcal{K} \)". 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 \( \pi \& e \), where \( \pi \) 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.
DEDUCTION OVER GRAPHS UNDER CONSTRAINTS...

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

![Figure 4: Constraint Inference rules](image)

The table shows the constraint inference rules for $G\mathcal{K}$-computations and their preordering. The rules are as follows:

- **Inclusion**: $\pi K e$ (constraint) $\mathcal{K}$ $e$ (edge)
  $\pi : \text{src}(e) \rightarrow \text{tgt}(e)$

- **Reflexivity (and Identity)**: $\varphi\mathcal{K}\varphi$
  $\varphi : A \rightarrow B$ $\psi\mathcal{K}\psi$

- **Transitivity (and Product)**: $\mathcal{K} w + \psi K \theta$
  $\mathcal{K} w + \psi K \theta$

- **Addition**:
  $\mathcal{K} w + \psi K \theta$
  $\mathcal{K} w + \psi K \theta$

- **Parenthesis**:
  $\mathcal{K} w + \psi K \theta$

- **Zero**
  $\varphi = X$ (node)
  $\varphi = X$ (node)
  $\varphi = X$ (node)
  $\varphi = X$ (node)

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

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

Now, two \( G|K \)-computations \( \varphi \) and \( \varphi' \) are said to be equivalent, denoted \( \varphi \equiv_K \varphi' \), iff \( \varphi \leq_K \varphi' \) and \( \varphi' \leq_K \varphi \). We extend the relation \( \equiv_K \) as follows:

\[
\varphi \equiv_K \varphi' \equiv_K \text{id}(\text{src}(\varphi)). \varphi \equiv_K \varphi. \text{id}(\text{tgt}(\varphi)), \quad \text{for every } G|K\text{-computation } \varphi.
\]

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 \( \text{src}, \text{tgt}, \cdot, \text{zero} \) and \( \text{id} \) can be defined on equivalence classes of \( G|K \)-computations. These operations have the same properties as in Section 3. For example, from the addition rules, the operation \( \cdot + \cdot \) is idempotent and product is distributive with respect to addition. Moreover, the preordering \( \leq_K^* \) induces an ordering on parallel \( G|K \)-computations, denoted \( \\leq_K \), which has also the same properties as the ordering \( \leq \) defined earlier on \( G \)-computations. In particular, \([\varphi] \leq_K [\varphi']\) if and only if \([\varphi] \cdot [\varphi'] = [\varphi']\cdot [\varphi', \text{i.e. if and only if } \varphi \cdot \varphi' = \varphi' \cdot \varphi].\) In view of the above rules and definitions, Proposition 4 can be extended as follows:

**Proposition 4** (continued) The equivalence classes of \( G|K \)-computations equipped with the operations \( \text{zero}, \text{id} \) and \( \cdot \), and the ordering \( \leq_K \cdot \) form a semantic universe, denoted \( eG|K \).

The meaning of Computations under Constraints

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

**Definition 8** Given an interpretation \((C, II)\) of a graph \(G\), we say that
- \((C, II)\) satisfies the constraint \(\varphi \leq \varphi'\), denoted \(\varphi \leq \varphi'\), if \(\varphi\) and \(\varphi'\) are meaningful with respect to \((C, II)\) and \(\varphi|_m \leq \varphi'|_m\),
- \((C, II)\) satisfies a set \(K\) of constraints, denoted \(\varphi \vdash K\), if \((C, II)\) satisfies every constraint in \(K\), and
- \((C, II)\) is a model of the specification \(G|K\) if \(\varphi \vdash K\).

To simplify matters we shall drop the index \(C\) when no confusion is possible.

Clearly, in the absence of constraints, every \(C\)-interpretation of \(G\) is a \(C\)-model of \(G\). Note that if \(\pi = e_1e_2...e_n\) is a path then \(\pi\) is meaningful and \(\pi|_m = e_1|_m e_2|_m ... e_n|_m\). Thus, \((C, II)\) satisfies the edge constraint \(\pi \leq e\) if \(\pi|_m \leq e|_m\). A specification \(G|K\) has always at least one \(C\)-model, namely, any interpretation \(\pi\) such that \(\pi(e) = 0\text{src}(le), \text{tgt}(le)\) for every edge \(e\) present in \(K\), is a model of \(G|K\). Such \(C\)-models are called trivial \(C\)-models. From now on by \(C\)-model we mean a nontrivial \(C\)-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 \(K \vdash k\), if every interpretation which satisfies \(K\) also satisfies \(k\). Given two sets of constraints \(K\) and \(K'\) over \(G\), we say that \(K\) implies \(K'\), denoted \(K \vdash K'\), 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 \(G|K\), if \(k\) is a constraint over \(G\) then \(K \vdash k\) 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 \((C, II)\) be an interpretation which satisfies a set of constraints \(K\). Clearly the inclusion rule is sound. Since \(\varphi\) is an ordering in the semantic universe, reflexivity and transitivity rules are sound too. To prove the soundness of addition rule, assume \(\varphi, \psi\) and \(\theta\) to be \(G|K\)-computations and assume \(\varphi \leq \varphi\), \(\psi \leq \psi\) and \(\theta \leq \theta\) to be satisfied, and prove that \(\varphi \oplus \psi \leq \varphi \oplus \psi\) and \(\varphi \leq \varphi\) are satisfied. The assumption means that \(\varphi\), \(\psi\) and \(\theta\) are meaningful and \(\varphi|_m \leq \varphi|_m\) and \(\varphi|_m \leq \varphi|_m\). Thus, \(\varphi|_m \leq \varphi|_m\) and \(\varphi|_m \leq \varphi|_m\). Thus, it is enough to prove that \(\varphi|_m \leq \varphi|_m\) and \(\varphi|_m \leq \varphi|_m\). With earlier notations we can write \(\varphi|_m \leq \varphi|_m\) and \(\varphi|_m \leq \varphi|_m\). But \(\varphi|_m \leq \varphi|_m\) is the sum of paths in \(\Delta^*(\varphi+\psi)\). Similarly, \(\varphi|_m \leq \varphi|_m\) is the sum of paths in \(\Delta^*(\varphi+\psi)\). As \(\Delta^*(\varphi+\psi) = \Delta^*(\varphi) \cup \Delta^*(\psi)\) so \(\varphi|_m \leq \varphi|_m\) differs from \(\varphi|_m \leq \varphi|_m\) by a permutation of its terms and by repetition of some of its terms. But \(\oplus\) is idempotent, associative and commutative, so \(\varphi|_m \leq \varphi|_m\).
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 \( (I, G) \) which does not satisfy \( k \). We shall give such a model when \( \mathcal{C} \) is \( \mathcal{U} \mathcal{P} \), using the (enriched) Yoneda’s Lemma [MacL71] as follows:

Let \( X \) be a node in \( G \) and consider the graph morphism \( l_X : G \to \mathcal{U} \mathcal{P} \) defined by:

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

The graph morphism \( l_X \) is a \( \mathcal{U} \mathcal{P} \)-model. Obviously \( l_X \) is a nontrivial interpretation. Moreover, if \( pse \) is in \( K \) then for every \( u : X \to src(p) \) we can write \( u.p \) \( l_X(u) \) that is, \( l_X(p)(u) \) is \( l_X(e)(u) \). In other words \( l_X(p) \to l_X(e) \) which means that \( l_X \) satisfies \( pse \).

Now, assume that \( k = csc' \) and \( K \vdash k \). The model \( l_{src(c)} \) does not satisfy \( k \), otherwise we must have \( l_X(c) \to l_X(c') \). This means that

\[
l_{src(c)}(c)(u) \not\equiv l_{src(c)}(c')(u) \text{ or, equivalently, } u.c \not\equiv u.c', \text{ for every } u : src(c) \to src(c).
\]

Taking \( u = id(src(c)) \) we must have \( c \not\equiv c' \) which contradicts \( K \vdash k \). This completes 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 \( \varphi \) is a \( G|K \)-computation. That is, \( \varphi \) is a \( G \)-computation and \( \varphi \) can be obtained by a finite number of application of computation rules beginning with \( G \) and \( K \). Similarly, \( K \models \varphi \) means that \( \varphi \) is a \( G \)-computation and \( \varphi \) is meaningful with respect to every model \( (G, \mathcal{U} \mathcal{P}) \) of \( K \).

**Theorem 3** The inference computation rules of Figure 4 are sound and complete, i.e. for each set \( K \) of constraints over \( G \), and for each \( G \)-computation \( \varphi \), \( K \vdash \varphi \) 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 \( \varphi \) be a \( G \)-computation such that \( \varphi \) is meaningful with respect to every model \( (G, \mathcal{U} \mathcal{P}) \) of \( K \). We must prove that \( \varphi \) is a \( G|K \)-computation. Let \( X = src(\varphi) \) and consider the model \( (\mathcal{U} \mathcal{P}, l_X) \) constructed in the proof of Theorem 2. Thus, \( l_X(\varphi) \) is meaningful in \( \mathcal{U} \mathcal{P} \). In particular \( l_X(\varphi)(id(X)) = \varphi \) has a meaning in \( e_G|K \), that is \( K \vdash \varphi \). This completes the proof.
Corollary 1 If \((\mathcal{C}, \mathcal{I})\) is a model of the specification \(G|K\) then any two equivalent \(G|K\)-computations have the same meaning with respect to \((\mathcal{C}, \mathcal{I})\).

Proof if \((\mathcal{C}, \mathcal{I})\) is a model then, by Theorem 2, all \(G|K\)-computations are meaningful. Moreover, if \(\varphi \leq_{K} \psi\) then if \(\varphi \leq_{K} \psi\) and \(\psi \leq_{K} \varphi\) then we can write

\[ I\varphi \mathcal{I}_{m} \implies I\varphi \mathcal{I}_{m} \quad \text{and} \quad I\psi \mathcal{I}_{m} \implies I\varphi \mathcal{I}_{m} \quad \text{so} \quad I\psi \mathcal{I}_{m} \equiv I\varphi \mathcal{I}_{m}. \]

Otherwise, \(\psi\) has the form \((\varphi), \text{id.}\varphi \) or \(\varphi \cdot \text{id.}\). In all these cases we obviously have \(I\psi \mathcal{I}_{m} \equiv I\varphi \mathcal{I}_{m}\). 

Specifications are objects of a category \(\text{Spec}\). A morphism of \(\text{Spec}\) from \(G_{i}K_{1}\) to \(G_{2}K_{2}\) is a graph morphism \(F : G_{1} \to G_{2}\) such that for every constraint \(e_{1}e_{2}...e_{n}se\) in \(K_{1}\) the constraint \(Fe_{1}Fe_{2}...Fe_{n}sFe\) is in \(K_{2}\). Let us see a semantic universe \(\mathcal{C}\) as a specification (not necessarily finite) whose constraints are all \(\pi \equiv e\) 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 monotone, so there is a forgetful functor \(U : \text{Sem} \to \text{Spec}\). Let us denote by \(\text{Spec/C}\) the category of objects over \(UC\) and by \(\text{Spec/C}\) the category of objects over \(\mathcal{C}\). We denote by \(\text{C-Mod}\) the full sub-category of \(\text{Spec/C}\) whose objects are \(\mathcal{C}\) models. The functor \(U\) defines a functor \(U/\mathcal{C} : \text{Sem/C} \to \text{C-Mod}\). 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 \((G[K], \mathcal{I})\) of \(\text{C-Mod}\) with the object \((\mathcal{C}G[K], \mathcal{I}\mathcal{C})\) of \(\text{Sem/C}\) defines a functor \(\eta_{\mathcal{C}}G[K] : \text{C-Mod} \to \text{Sem/C}\) which is a left free adjoint to the functor \(U/\mathcal{C} : \text{Sem/C} \to \text{C-Mod}\).

Proof Since \(\mathcal{I}\mathcal{C}_{m}\) is a semantic function \(\mathcal{I}\mathcal{C}_{m}\) is a \(\mathcal{C}\)-model of \(\mathcal{C}G[K]\), so

\[ U/\mathcal{C}(\mathcal{C}G[K], \mathcal{I}\mathcal{C}_{m}) \] is an object of \(\text{C-Mod}\). Let \(\eta_{\mathcal{C}G[K]} : G[K] \to U(\mathcal{C}G[K])\) be the inclusion graph morphism. As \(U(\mathcal{I}\mathcal{C}_{m})\eta_{\mathcal{C}G[K]} = \mathcal{I}\mathcal{C}_{m}\), so \(\eta_{\mathcal{C}G[K]}\) is an arrow of \(\text{C-Mod}\) from object \((G[K], \mathcal{I})\) to \(U/\mathcal{C}(\mathcal{C}G[K], \mathcal{I}\mathcal{C}_{m}) = U/\mathcal{C}(\mathcal{C}G[K], \mathcal{I}\mathcal{C}_{m})\). We shall prove that \(\eta_{\mathcal{C}G[K]}\) is the unit of an adjunction (see Figure 5, with the same convention as in Figure 3).

Let \((\mathcal{C}', T)\) be an object of \(\text{Sem/C}\) and let \(F : \mathcal{C} \to \mathcal{C}'\) be an arrow of \(\text{C-Mod}\) from \((G[K], \mathcal{I})\) to \(U/\mathcal{C}(\mathcal{C}', T) = (U(\mathcal{C}'), U(T))\). Let \(\varphi\) be a \(G|K\)-computation with
canonical form $e^\varphi = p_1 + p_2 + ... + p_n$. As $\| \|$ is a model, $\varphi$ is meaningful, that is, $\| \varphi \|_m = \| p_1 \|_m \cdot \| p_2 \|_m \cdot ... \cdot \| p_n \|_m$ exists in $\mathcal{C}$. As $\varphi$ is obtained by the addition rule, there is a computation $\pi$ such that $\rho_1 \rho_2 \rho_3$ for all $i$, $1 \leq i \leq n$. But $F$ is an arrow of $\mathcal{C}$-Mod, so $F(\rho_i) \cdot F(\pi)$ for all $i$, $1 \leq i \leq n$. This implies that $\varphi^* = F(\rho_1) \cdot ... \cdot F(\rho_n)$ exists in $\mathcal{C}$. If we define $H(\varphi) = \varphi^*$ then $H$ is a semantic function and defines an arrow of $\text{Sem/C}$, as it satisfies $TH = \| \|_m$. Moreover, the equation $U(H)_{\|G\|_K} = F$ is satisfied in $\mathcal{C}$-Mod and $H$ is the unique arrow of $\text{Sem/C}$ satisfying this equation. This completes the proof. 

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

5 CONSISTENCY AND FIXPOINT SEMANTICS

Let $G[K]$ be a specification and let $\| \||$ be an interpretation of $G$ in some semantic universe $\mathcal{C}$. If $\| \||$ is not a model of $G[K]$ 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 $G[K]$, we say that an interpretation $(\mathcal{C}, \| \||)$ of $G$ is *consistent* with $K$ if every $G[K]$-computation is meaningful with respect to $\| \||$. 

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

**Proposition 5** Let $G[K]$ be a specification and let $(\mathcal{C}, \| \||)$ be an interpretation of $G[K]$. Then $(\mathcal{C}, \| \||)$ is a $\mathcal{C}$-model of $G[K]$ if and only if $(\mathcal{C}, \| \||)$ is consistent with $K$ and any two equivalent and meaningful $G[K]$-computations have the same meaning with respect 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 $G[K]$-computations have the same meaning. Let $\pi \epsilon e$ be an edge constraint in $K$. We can write $e_\varphi \epsilon e$ by reflexivity, and then by addition rules, $e + \pi$ is a $G[K]$-computation and $e + \pi = e_\varphi \epsilon e + \pi$. Thus, $e + \pi \Rightarrow e_\varphi \epsilon e$ and we have $\| e + \pi \|_m = \| e \|_m$. That is $\| e \|_m \cdot \| \pi \|_m = \| e \|_m$ or, equivalently, $\| \pi \|_m \Rightarrow \| e \|_m$. This means that $\| \||$ satisfies $\pi \epsilon e$. This completes the proof. 

LS 17
**Definition 10** Let $G\mathcal{K}$ be a specification. The derivative of a $G$-computation $\varphi$, denoted $\partial \varphi$, is defined recursively as follows:

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

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

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

**Lemma 1** If $\varphi$ is a $G\mathcal{K}$-computation then so is $\partial^0 \varphi$ and $\partial^i \varphi$, 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 $G\mathcal{K}$-computation and $\partial \varphi$. If $\varphi$ is an edge of $G$ and $\partial \varphi = zero(src(\varphi), tgt(\varphi))$ then $\varphi$ is the head of some constraint, $\partial \varphi$ is obtained by the addition rule and $\partial \varphi$. Now, let $\varphi$ be a derived $G\mathcal{K}$-computation such that, for each component $\varphi$ of $\varphi$, $\partial \varphi$ is a $G\mathcal{K}$-computation and $\partial \varphi$ (structural induction hypothesis). If $\varphi$ has the form $u \cdot v$ then $\partial u$ and $\partial v$ are $G\mathcal{K}$-computations and $\partial u \cdot vs_k \cdot u$ and $\partial v \cdot vs_k \cdot v$ by the induction hypothesis. So by augmentation $\partial u \cdot vs_k \cdot u \cdot v$ and $\partial u \cdot vs_k \cdot v$ by the induction hypothesis.

Thus, $\partial \varphi = u \cdot \partial v + \partial u \cdot v + \partial u \cdot \partial v$ is a $G\mathcal{K}$-computation and $\partial \varphi \cdot us_k \cdot u \cdot v = \varphi$, by the addition rule. Similarly, if $\varphi$ has the form $u + v$, where $\partial u$ and $\partial v$ are $G\mathcal{K}$-computations satisfying $\partial u \cdot us_k \cdot u$ and $\partial v \cdot vs_k \cdot v$, then $\partial u \cdot us_k \cdot v$ and $\partial v \cdot vs_k \cdot v$, so $\partial \varphi = \partial u + \partial v$ is a $G\mathcal{K}$-computation and $\partial \varphi \cdot u \cdot v = \varphi$ by addition rules. This completes the proof.

**Definition 11** An edge $e$ in a specification $G\mathcal{K}$ is said to be non recursive if there is an integer $k \geq 0$ such that $\partial^{(i+1)} e = zero(src(e), tgt(e))$; 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 $\partial^{(i+1)} e = zero(src(e), tgt(e))$, is called the depth of $e$, and is denoted $depth(e)$.
We stress that acyclicity refers to constraints $K$ and not to graph $G$. The way edges in $K$ depend on one another, may be represented as a graph $K$ called the dependency graph. This graph is defined as follows:

- The nodes of $K$ are the edges of $G$, and
- There is an edge from $f$ to $e$ in $K$ whenever $e$ is the head of a constraint in $K$ and $f$ is the body of that constraint.

Note that for every edge $e$ of $G$ the edges which appear in $d(e)$ are the immediate sons of $e$ in $K$. In particular, leaves of $K$ are those edges of $G$, which appear only in the body of constraints. This allows to state the following propositions and lemmas which express important properties of cyclic specifications.

**Proposition 6** If $G \subseteq K$ is cyclic if and only if its dependency graph is cyclic.

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

**Lemma 2** If $G \subseteq K$ 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 $K \vdash l_e \leq r_e \leq e$. Moreover, $l_e$ and $r_e$ are cycles of $G$, but $l_e$ and $r_e$ are not necessarily unique nor with positive length.

**Proof** Note that when we write $K \vdash l_e \leq r_e \leq e$, the paths $l_e$ and $r_e$ are cycles in $G$. Indeed, $l_e \leq r_e$ and $e$ are parallel so $\text{src}(l_e) = \text{src}(l_e \leq r_e) = \text{src}(e)$ and $\text{tgt}(l_e) = \text{src}(e)$ 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 $G$ such that the constraints $l_0 \leq e_0 \leq e, l_1 \leq e_1 \leq e, ..., l_n \leq e_n \leq e$ are in $K$, where $l_i$ or $r_i$ may be the empty path for $i = 0, 1, ..., n$. It follows that $K \vdash l_e \leq r_e \leq e$, where $r_e = r_{0n} \leq ... \leq r_{n-1} \leq r_n$ and $l_e = l_0 \leq ... \leq l_n \leq e$. 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 $G \subseteq K$ 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 $\mathcal{D}^{(l_e\mathbb{1})e}$. This completes the proof.
Lemma 3 In a linear cyclic specification $G \mathcal{K}$, for every recursive edge $e$, and for all integers $n$ and $k$, if $1 \leq n$ and $0 \leq k < \text{depth}(e)$ then
\[
\delta^{(n \times \text{depth}(e) + k)} e = l_e^n \delta^{(k)} e r_e^n
\] (*)

Proof We shall prove (*) by induction on $k$ and $n$. From the proof of Lemma 2 we can see that $\delta^{(\text{depth}(e))} e = l_e \cdot r_e$. This means that the equality (*) stands for $k=0$ and $n=1$. Assume that for a given $n \geq 1$, equality (*) stands for $0 \leq k < \text{depth}(e) - 1$ (induction hypothesis for $k$). As $l_e$ and $r_e$ are formed by non recursive edges, we can write
\[
\delta^{(n \times \text{depth}(e) + k + 1)} e = \delta^{(n \times \text{depth}(e) + k)} e l_e r_e \delta^{(k + 1)} e r_e.
\]
Similarly, assume that for a given $k$, $0 \leq k < \text{depth}(e)$, equality (*) stands for $n \geq 1$ (induction hypothesis for $n$). We can then write,
\[
\delta^{((n + 1) \times \text{depth}(e) + k)} e = \delta^{(n \times \text{depth}(e) + k)} e l_e r_e \delta^{(k)} e r_e = l_e^n \delta^{(k + 1)} e r_e.
\]
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
\[
T_i(e) = \sum_{k=0}^{i} \delta^{(k)} e
\]
is a $\mathcal{G}\mathcal{K}$-computation and $T_i(e) \mathcal{G}\mathcal{K} e$. Moreover,
\[
T_i(e) = T_{i-1}(e) + \int_{e}^{\lambda_i(e)} \delta \eta_i(e) e r_{\lambda_i(e)}
\]
where $\lambda_i(e) = i \div \text{depth}(e)$, $\eta_i(e) = i \mod \text{depth}(e)$.

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 $i \geq \text{depth}(e)$, the $\mathcal{G}\mathcal{K}$-computation $T_i(e)$ uses only $\delta^{(k)} e$, for $0 \leq k < \text{depth}(e)$. Moreover, since the specification is assumed to be linearly cyclic, the cycles $l_e$ and $r_e$ and all $\delta^{(k)} e$, $0 < k < \text{depth}(e)$, 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 $\mathcal{G}$-interpretations of a given specification $\mathcal{G}\mathcal{K}$ as follows:

1. $\mathcal{I} \leq \mathcal{I}'$ if and only if 1) $\mathcal{I} A = \mathcal{I}' A$ for every node $A$, and 2) $\mathcal{I} e \leq \mathcal{I}' e$ 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 $(u, u^2, ... , u^{n-1})$ is consistent.
and \( u^1 \leq u + u^2 + \ldots + u^{n-1} \) for every \( k \geq n \). The least \( n \) satisfying this inequality is called the \textit{height} of \( u \) and is denoted \textit{height}(\( u \)).

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

Let \( \mathcal{G} \mid K \) 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 \( \| \) which are consistent with \( K \) and \( \| l \| \leq 1 \). Each \( \| l \| \) in \( \mathcal{I} \) can be extended to all \( \mathcal{G} \mid K \)-computations. For simplicity, let us denote the extension of \( l \) also by \( l \). Then \( l(e \oplus \partial e) \) exists and is equal to \( l(e) \ominus l(\partial e) \), for every edge \( e \) in \( \mathcal{G} \).

Now, we define \( T : \mathcal{I} \rightarrow \mathcal{I} \) by:

- \( T(\| A \|) = \| A \| \), for every node \( A \), and
- \( T(\| e \|) = l(e) \ominus l(\partial e) \), for every edge \( e \).

Theorem 4 With the above hypotheses, \( \mathcal{I} \) is a wposet with least element \( \| 1 \| \), and the function \( T \) is continuous and has a least fixpoint \( \| e \| ^S \) which is a model of \( \mathcal{G} \mid K \).

Proof From the definition of the ordering of \( \mathcal{I} \), it is clear that \( \| 1 \| \) 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 \( l_1, l_2 \) be in \( \mathcal{I} \) and assume that \( \text{lub}(l_1, l_2) \) exists in \( \mathcal{I} \). This means that for every edge \( e \) in \( \mathcal{G} \), \( l_1(e) \ominus l_2(e) \) exists in \( \mathcal{C} \). Thus we have \( T(\text{lub}(l_1, l_2))(e) = l_1(e) \ominus l_2(e) \ominus l_1(\partial e) \ominus l_2(\partial e) = T(l_1(e)) \ominus T(l_2(e)) = \text{lub}(T(l_1), T(l_2))(e) \). 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 \( \| e \|, \| e \| \ominus \partial e, \| e \| \ominus \partial e \ominus \partial^2 e, \ldots \). Using the above notation we can write \( \| e \| ^S = \lim_\Psi \| T_\Psi(e) \| \).

We must show that \( \| e \| ^S \) can be computed in a finite number of steps i.e. we must show that \( \| e \| ^S \) is bounded by an arrow of \( \mathcal{C} \). Using the above Lemmas, if \( e \) is not recursive then \( \| e \| ^S = \| e \| \ominus \partial e \ldots \partial^{\text{depth}(e)-1} e \| \).

Otherwise, consider \( l_e \) and \( r_e \) as in Lemma 2 and denote

\[
\begin{align*}
\eta_e &= \max(\text{height}(\| l_e \|), \text{height}(\| r_e \|)), \\
\Lambda_e &= \| l_e \| \ominus l_e \ominus \ldots \ominus \| l_e \|^{n-1}, \\
\Lambda_e &= \| l_e \| \ominus l_e \ominus \ldots \ominus \| l_e \|^{n-1}.
\end{align*}
\]

Now, for all \( n \geq 0 \) and all \( k \), \( \text{Osk} < \text{depth}(e) \) we can write \( \| l_e \| ^l \| \partial^k e \| \| r_e \| ^l = l_e \| \partial^k e \| r_e \) for every \( l > 0 \).

(*)

From Lemma 4 we can write

\[
T_\Psi(e) = \sum_{k=0}^{\text{depth}(e)-1} \partial(\Psi)e \sum_{l=1}^{\text{depth}(e)-1} \partial(\Psi)e \sum_{k=0}^{\text{depth}(e)-1} l_e \| \partial^k e \| r_e.
\]

(**)
Now, from (*) and (**) we can write
\[ \| e \| \leq \Phi^{\text{depth}(e)}(k) \| \sigma(k) e \| \circ \mathcal{R}_e(\Phi^{\text{depth}(e)}(k) \| \sigma(k) e \| ) \leq. \]

This inequality prove that \( \| e \| \) is bounded which completes the proof.

In [LeSp93] we show that \( \| I \| ^S \) can be computed by an effective algorithm, in the semantic universe \( m.\text{Set} \).

**Fixpoint Semantics** Using Theorem 1, the model \( \| I \| ^S \) can be extended to \( \mathcal{G} \mathcal{K} \) as a semantic function \( \| I \| ^S_m \). This semantic function provides meaning for all computations under the constraints of \( \mathcal{K} \). Let us consider the functor \( U : \text{Sem} \rightarrow \text{Spec} \) seen earlier and let us consider the full sub-category of the category of \( U \)-objects under \( (\mathcal{G} \mathcal{K}, \| I \| ) \), whose object are consistent interpretations. The pair \( (\mathcal{G} \mathcal{K}, \| I \| ^S_m) \) 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 \( (\mathcal{G} \mathcal{K}, \| I \| ^S_m) \) the fixpoint semantics of \( (\mathcal{G} \mathcal{K}, \| I \| ) \).

### 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 specificaiton. 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


DEDUCTION OVER GRAPHS UNDER CONSTRAINTS...

[MacL71] S. Mac Lane, Categories for the Working Mathematician (Springer, Berlin, 1971)


