Logique & Analyse **153-154** (1996), 153-164

ON RELATIONS BETWEEN WP-GRAPHS AND DATA STRUCTURES[1]

M.V. KRITZ

Abstract
The mathematical objects called *wp*-graphs were introduced to model biological information. They process, notwithstanding, several interesting properties not directly related to its original motivation. It is here shown that various of the most commonly used programming data structures can be recasted as special cases of *wp*-graphs and, thus, that these data-structures may be expressed in a common framework, which partially explains why *wp*-graphs are showing a wide range of potential applications. Although it is not claimed that this general form is good to use or implementation, it does reduce some constructive meta-concepts to the same level of the constructed data-structures, i.e., they are also *wp*-graphs.

1. *Introduction*

The mathematical objects called *wp*-graphs were introduced in [Kritz 1991], named there i-graphs, to mathematically model configurations that convey biological information, a new concept also introduced therein. However, a lot of other aspects and possible applications of these objects have thereafter been discovered. One to mention: is that they happen to generalize, in a sense to be made precise, many data-structures currently used in programming and computer science [R. Scialom, 1990].

Roughly speaking, biological information is conveyed by any group of parts and wholes from a universe, bearing somehow relations to one another. These configurations —of parts, wholes and their relations— can be mathematically modelled by *wp*-graphs. A *wp*-graph is recursively constructed using hypergraphs as building blocks and can be informally described as hypergraphs, whose nodes can be either hypergraphs or sets thereof; i.e. they can be also *wp*-graphs.

[1]This work was presented at the 2nd Gauss Symposium, 2-6 Aug. 1993, Munich, Germany.

By sticking to *plain* programming data structures —that is, to structures which definition involves neither environmental concepts nor special (meta-)rules and endomorphisms— it is possible to show that several of them can be recasted as *wp*-graphs. This is done by controlling the class of hypergraphs allowed to intervene in the construction of a *wp*-graph, leading thence to sub-classes of the *wp*-graph space. To clarify what plain structures are, consider stacks and queues. Elements of either class are a (sequential) list of objects endowed with the idea of inserting and extracting elements (parts) to and from it in a specific manner, and rules to define these processes. These classes differ just because they have distinct insertion/extraction rules. In this case, the insertion/extraction *idea* is the environmental concept and the definition rules, the special rules. Here, lists are the plain data-structures, since their definition involves just relational and organizational aspects among their parts.

The aim of the present work is to show that several *plain* data structures occurring in programming and computer science —among which one counts: lists, s-expressions, recursive trees, graphs (and, thus, non-recursive trees), (multi-dimensional) arrays, relations and so on— can be obtained as particular instances of *wp*-graphs, and how to do it. To achieve this I informally present, in the next section, some aspects of *wp*-graph definition relevant to the present work. In the third section, data structures are presented and shown to be *wp*-graphs. The last section contains some concluding remarks.

2. *About wp*-graphs

The stropped acronym *wp*-graph stands for *whole-part*-graphs.[2] They are mathematical objects recursively defined in a constructive manner, having hypergraphs or, more precisely, extended hypergraphs as their basic building blocks. A hypergraph is, quite simply, a family of subsets of a given set —the set of its nodes— such that the union of all elements in the family *do cover* the starting node set. Formally, a hypergraph is a pair [Berge 1970]:

$$\mathsf{h} = \{N, A \subset \wp\,(N)\}, \tag{1}$$

[2]The initial [Kritz, 1991] acronym i-graph stands for *information*-graphs. However, biological information is immaterial, being *carried* by whole-part configurations, the description of whole-part arrangements being a more fundamental characteristic of these mathematical objects.

where N is a non-empty finite set (of nodes), $\wp(N)$ is the set of all its subsets and the arcs $a_i \in A$, $i \in I$, I non-empty, are such that

$$\left.\begin{array}{rcl} a_i & \neq & a_j, (i \neq j), \\ a_i & \neq & \varnothing, (\forall i \in I), \\ \bigcup_{i \in I} a_i & = & N. \end{array}\right\} \tag{2}$$

To construct *wp*-graphs, however, hypergraphs need to be extended, by simultaneously restraining and enlarging its possible nodes. If $N(\mathsf{h})$ denotes the node set of h, an *extended* hypergraph is a hypergraph such that $N(\mathsf{h}) \subset \mathbf{U}_{\odot}$, being $\mathbf{U}_{\odot}$ the following set (of admissible nodes):

$$\mathbf{U}_{\odot} = \mathbf{U} \cup \mathbf{V} \cup \{\odot\} \tag{3}$$

Above, $\mathbf{U}$ is an universe set of (atomic) ground objects,

$$\mathbf{V} = \{v_1, v_2, v_3, \ldots\},$$

is an infinite set of (meta-)variables and $\odot$ is a special atomic element, used below to define a hierarchical assignment. Conforming to biological motivations, $\mathbf{U}$ is taken to be an admissible set in the sense of [Barwise 1975] —i.e., its elements may be either simple, atomic, elements or sets. As before, the node set satisfies $N(\mathsf{h}) \neq \varnothing$ # $N(\mathsf{h}) < \infty$ and the family $A(\mathsf{h})$ must satisfy (2), but for extended hypergraphs it is also required that $N(\mathsf{h}) \neq \{\odot\}$. (Strictly speaking, $N(\mathsf{h}) \not\subset \mathbf{U}$ should also be required. Nevertheless, this would only burden some proves below without changing the results.) The class of all extended hypergraphs will be called $\mathcal{H}$. The notation and definitions below will be useful further on:

$$\left.\begin{array}{rll} \text{if } \odot \in N(\mathsf{h}), & \text{define} & \mathsf{h} = \mathsf{h}^{\circ} \wedge \mathsf{h}^{\circ} \in \mathcal{H}^{\circ} \wedge \odot \ \mathbf{in}\ \mathsf{h}^{\circ}\ ; \\ \text{if } (\exists v \in \mathrm{V})\,[v \in N(\mathsf{h})], & \text{define} & \mathsf{h} = \mathsf{h}^{*} \wedge \mathsf{h}^{*} \in \mathcal{H}^{*} \wedge v\ \mathbf{in}\ \mathsf{h}^{*}\ ; \\ \text{if } \mathsf{h} \in \mathcal{H}^{\circ} \wedge \mathsf{h} \in \mathcal{H}^{*}, & \text{define} & \mathsf{h} = \mathsf{h}^{\bullet} \wedge \mathsf{h}^{\bullet} \in \mathcal{H}^{\bullet} = \mathcal{H}^{\circ} \cap \mathcal{H}^{*}. \end{array}\right\} \tag{4}$$

The whole construction of *wp*-graphs is based on the following (binding) operator defined for extended hypergraphs, which *identifies* an extended hypergraph of type h° to a v-node in an extended hypergraph of type h^{*} by "assigning" $\odot$ **in** h° to a v **in** h^{*}. Namely,

$$\hookleftarrow : \ (\mathsf{h}^{*}, \mathsf{h}^{\circ}) \quad \mapsto \quad \mathsf{h}' = \mathsf{h}^{*} \hookleftarrow \mathsf{h}^{\circ} \tag{5}$$

$$\mathsf{h}^{*}\ \mathbf{ni}\ v \quad \text{"="} \quad \odot\ \mathbf{in}\ \mathsf{h}^{\circ}, \tag{6}$$

where "=" means that the whole h° in right hand side is to be interpreted as an atom and taken as a node of h^*, "occupying the place of v", or "filling" it. Likewise, if s is a set of $\mathsf{h} \in \mathcal{H}$, the binding $\mathsf{h}' = \mathsf{h}^* \;\lrcorner\; \mathsf{s}$ can be also defined by letting v "=" s, v **in** h^*, in the usual renaming sense, since set nodes are allowed (3). Moreover, u **in** h' if either u **in** h^* or u **in** h° (or $(\exists \mathsf{h} \in \mathsf{s})\,[u \textbf{ in } \mathsf{h}]$).

Observe that, if $\mathsf{h}^* = {}_m\mathsf{h}^* \in \mathcal{H}^*$ is such that $N({}_m\mathsf{h}^*) \cap \mathbf{V} = \{v_1, ..., v_m\}$ and $n \le m$, we can extend the above binding operator $\lrcorner$ collaterally to

$$ {}_m\mathsf{h}^* \;\lrcorner\; <\mathsf{h}^\circ_1, ..., \mathsf{h}^\circ_n>, \qquad (7)$$

just by requiring that

$$ {}_m\mathsf{h}^* \textbf{ ni } v_i \text{ "=" } \odot \textbf{ in } \mathsf{h}^\circ_i \, ,$$

and, analogously, to any group of h°s and sets of hs. Thence, we can define *wp*-graphs to be the members of a class ***wp-Γ*** constructed in the following way [Kritz 1991]:

Def 2.1 An object γ *is an wp*-graph, *that is,* $\gamma \in$ ***wp-Γ***, *if and only if*
1. $\gamma \in \mathcal{H}$
2. $\gamma = \{\gamma_1, ..., \gamma_k\}$, *where* $\gamma_i \in$ ***wp-Γ***, $\forall 1 \le i \le k$,
3. $\gamma \cong {}_m\mathsf{h}^* \;\lrcorner\; <\gamma^\circ_1, ..., \gamma^\circ_n>,\ n \le m,$

where $\cong$ *reads "is obtained as", "is identified to", or "is of the form of"; and either* $\odot$ **in** γ°_i *or* γ°_i *is a set like in case 2,* $\forall 1 \le i \le n$.

Note that the predicate **in** can be naturally extended to ***wp-Γ***, and, if γ is not a set, it has always a recursive upmost h, that is, an "upmost whole", which is called its *seed*. It may be assumed above that $\odot$ **in** γ_i means that $\odot$ is a node of seed of γ_i.

3. *About wp*-graphs *and data structures*

For organizational purposes, recursive and non-recursive structures are distinguished below, being the latter analysed first. Observe that, while non-recursive structures have the number of its parts —parametrically or explicitly— fixed by their definition, this is however not the case among recursively defined structures, whose number of parts is left undetermined by their definitions. Notwithstanding, some data structures can be defined in either way.

3.1. *A priori bounded data structures*

In this subsection, structures defined without recursive constructions are analysed, amongst which (undirected) graphs, trees, sets, functions, arrays and relations may be found.

To begin with, consider graphs and sets. A (directed) graph is defined in the literature in various ways, eventually with small differences that are latter sharped out. They are often described as a pair of sets [Berge 1976], holding their nodes and arcs, or as any subset of a cartesian product [Abe & Papavero 1992]. An undirected graph is often said to be a graph where the arcs' directions are left unspecified [Berge 1976]. Graphs are here considered as:

Def 3.1 A graph $g = \{N,A\}$ *is a pair of sets, with* $A \subset N \times N$, *satisfying* $(\forall n \in N)\ (\exists n' \in N)$ *such that either* $(n,n') \in A$ *or* $(n',n) \in A$.

This rules out multi-edges and isolated nodes, but allows for loops.

Lemma 3.1 (Graphs)
*Any graph g, as such, is a hypergraph and, so, an wp-*graph.

Proof Since the direction is unspecified, perforce, for each $(n_1,n_2) \in A$, either $(n_2,n_1) \notin A$ or they have been made equivalent. In either case, an edge between a pair of nodes can be described by the set $\{n_1,n_2\}$ instead. Thereafter, $A \subset \wp_2\,(N) \subset \wp\,(N)$, where $\wp_2\,(N) = \{s \subset N \mid \#\,(s) \leq 2\}$ and hence, as A satisfies (2), it's clear that g belongs to $\mathcal{H}$ and thus to ***wp-Γ***, by definition. □

With respect to sets, it is indeed true that **U**, through each of its elements, can be immersed in ***wp-Γ***.

Lemma 3.2 (Sets)
If $u \in \mathbf{U}$, *it can be associated to an element of* ***wp-Γ***. *So, any set* s $\subset \mathbf{U}$ *can be associated to an wp-*graph.

Proof For each $u \in \mathbf{U}$, consider the hypergraph $\mathsf{u} = \{\{u\},\{u\}\}$. Since it satisfies all conditions imposed on extended hypergraphs, u belongs to $\mathcal{H}$ and, thus, to ***wp-Γ***. Moreover, if $\mathsf{h} \in \mathcal{H}$ is such that $[\mathsf{N}\ (\mathsf{h}) \subset \mathbf{U} \wedge \#\ (\mathsf{N}\ (\mathsf{h})) = 1]$, then $\#\ (\mathsf{A}\ (\mathsf{h})) = 1$ and $\mathsf{h} = \{\{u\},\{u\}\}$, for some $u \in \mathbf{U}$. Therefore, any set whose elements belong to **U** can be turned into a set of *wp-*graphs as in case 2 of definition 2.1. They are, thus, *wp-*graphs as well. □

All other data-structures examined in this section can, in one way or another, be recasted as graphs and, of them, only *arrays* need concepts specific to *wp*-graphs. The following is indeed a special type of graph.

Lemma 3.3 (Trees)
*A (non-recursive) tree is an wp-*graph.

Proof A *tree* is an acyclic connected graph [Knuth, 1973]. Then, in the sense of Lemma 3.1, is a *wp*-graph. □

Amongst the most frequently used data structures are tables. However, although functions are (computationally) more often associated with the procedure or algorithm concept, tables are really a roll of argument-value pairs of functions. Here, their arguments will always lay in a domain (D) and their values in a co-domain (cD). These sets, for the time being, are taken as subsets of **U**.

Lemma 3.4 (Functions and Tables)
Let $f : \mathsf{D} \longrightarrow \mathsf{cD}$ *be a function, where* $\#(\mathsf{D})$ *and* $\#(\mathsf{cD})$ *are finite. Then, the object f can be associated with an wp-*graph.

Proof Let $\mathsf{g} = \{N, A\}$, where $N = \mathsf{D} \cup \mathsf{cD}$ and $A = \{\{x, f(x)\}, x \in \mathsf{D}\} \cup (\mathsf{cD} \setminus f(\mathsf{D}))$. Since f is a function, A satisfies (2) and g is a hypergraph (effectively obtained from the graph of f), hence an *wp*-graph. Furthermore, if g is a hypergraph such that $\mathbf{U} \supset N(\mathsf{g}) = M_1 \cup M_2$, $A(\mathsf{g}) \subset \wp_2(N(\mathsf{g}))$ and the arc set satisfies

$$(\forall a \in A)\,[a \cap M_1 \neq \emptyset \Rightarrow a \cap M_2 \neq \emptyset]$$

and

$$(\forall x \in M_1)\,(\exists a \in A)\,[x \in a]$$

; then g represents a function such that $\mathsf{D} = M_1$, $\mathsf{cD} = M_2$ and $f(x) = y$, where $[x \in a \cap M_1] \wedge [y \in a \cap M_2]$, whenever $a \cap M_1 \neq \emptyset$. □

Another important programming data structure is the fixed-size hash tables [Aho et alli 1983]. They can be simply seen as multi-valued functions F : Keys $\longrightarrow$ Set_of_Objects, satisfying $[k \neq k' \Rightarrow F(k) \cap F(k') = \emptyset]$; which forces its inverse to be a function, the *hash* function. That these objects can be seen as *wp*-graphs is a trivial consequence of the above lemma.

Lemma 3.5 (Multi-valued Functions and Hash Tables)
If $F : \mathsf{D} \longrightarrow \mathsf{cD}$ *is a multi-valued function, that is,* $\mathsf{cD} \subset \wp(B)$, *for some set B, it can be represented as an wp-*graph.

Proof It is enough to observe that, in the above lemma, cD and M_2 may contain sets as elements, as sets can be elements of **U**. □

Arrays are essentially functions between *indices* and *values* [Tennent 1981]. How indices are defined depends on the special programming language considered; but generally they are lattice-like structures, obtained as cartesian products of linearly ordered sets. There is, however, a fundamental difference now: array values can also be references or even quite complex structures (but, usually, not a mixture of them) and the correspondence is always assumed bijective. So,

Lemma 3.6 (Arrays)
Let $\mathscr{A} : \mathsf{D} \longrightarrow \mathsf{cD}$, *where* $\mathsf{D} = \prod_{i=1}^{k} J_i$, $J_i = \{\underline{l}_i, \ldots, \bar{l}_i\}$, $\underline{l}_i \leq \bar{l}_i \in \mathbf{Z} \subset \mathbf{U}$, *be an array, assuming that either* $\mathsf{cD} \subset \mathbf{U}$, $\mathsf{cD} \subset \mathbf{V}$ *or* $\mathsf{cD} \subset \boldsymbol{wp}\text{-}\boldsymbol{\Gamma}$. *Arrays of this sort can be associated to wp-*graphs *in a way similar as above.*

Proof If $\mathsf{cD} \subset \mathbf{U}$, Lemma 3.4 is the proof. If $\mathsf{cD} \subset \mathbf{V}$, given an array $\mathscr{A}$ consider $\mathsf{g} = \{N, A\}$, where $N = \mathsf{D} \cup \{v_1, \ldots, v_p\}$, $p = \#(\mathsf{D})$, and $A = \{\{\nu, v_\nu = \mathscr{A}[\nu]\}, \nu \in \mathsf{D}\}$. Clearly, $N(\mathsf{g}) \subset \mathbf{U}_\odot$ and $A(\mathsf{g})$ satisfies (2), thus $\mathsf{g} \in \boldsymbol{wp}\text{-}\boldsymbol{\Gamma}$. Conversely, if g is a hypergraph where $N(\mathsf{g}) = I \cup V \subset \mathbf{U}_\odot$, $I = \prod_{i=1}^{k} J_i$, $V \subset \mathbf{V}$, $\#(V) = \#(I)$ and $A(\mathsf{g}) \subset \wp_2(N(\mathsf{g}))$, satisfying $(\forall a \in A)\,[a \cap I \neq \emptyset \wedge a \cap V \neq \emptyset]$, it defines an array $\mathscr{A}$ by letting $\mathscr{A}[\nu] = v$, for each $\{\nu, v\} \in A$.

If $\mathsf{cD} \subset \boldsymbol{wp}\text{-}\boldsymbol{\Gamma}$, let p be as above and $V_a = \{v_\nu \in \mathbf{V}, \nu \in \mathsf{D}\}$. It is true that $\mathsf{cD} = \{\gamma_\nu = \mathscr{A}[\nu], \nu \in \mathsf{D}\}$, $p = \#(V_a) = \#(\mathsf{cD})$ and $\mathsf{A} : \mathsf{D} \longrightarrow V_a$, such that $\mathsf{A}[\nu] = v_\nu$ is an array. Denoting by G_a the *wp-*graph associated to A by the previous construction, $\mathscr{A}$ is then associated to $\mathsf{G}_a \lrcorner <\gamma_1, \ldots, \gamma_p>$, where γ_ν binds to v_ν. □

In the last step of the proof above, cD may contain *any wp-*graph. It's always possible, however, to assume that cD is contained in a *sub-class* of $\boldsymbol{wp}\text{-}\boldsymbol{\Gamma}$, such as the ones defined in this work.

More generally than the description of functions and arrays, *wp-*graphs happen to also encompass descriptions of relations among objects. A relation is here taken simply as a function R from a cartesian product to $\{0,1\}$, and the objects $x_i \in X_i$, $i = 0, \ldots, n$ are related if, and only if, $R(x_1, \ldots, x_n)$

= 1. Or, equivalently, as a subset R of $\prod_{i=1}^{n} X_i$, the two forms being related by $\mathsf{R} = \{(x_1, ..., x_n) \in \prod_{i=1}^{n} X_i \mid R(x_1,...,x_n) = 1\}$.

Lemma 3.7 (Relations)
Let $\mathsf{R} \subset \prod_{i=1}^{n} X_i$ *be a n-ary relation where* $X_i \subset \mathbf{U}, \forall (1 \leq i \leq n)$. *It can be then associated to an wp-*graph.

Proof Let $\mathsf{h} = \{N,A\}$, being $N(\mathsf{h}) = \bigcup_{i=1}^{n} X_i$ and

$$A(\mathsf{h}) = \{a = \{y_1, ..., y_n\} \mid (\forall 1 \leq i \leq n)\, [y_i \in X_i] \wedge R(y_1,..., y_n) = 1\} \cup A^{\circ}$$

, where $P_i(\mathsf{R})$ is the i-th canonical projection and $A^{\circ} = \{\{x\} \mid x \in (\bigcup_1^n (X_i \setminus P_i(\mathsf{R})))\}$. So defined $\mathsf{h} \in \boldsymbol{wp\text{-}\Gamma}$, since $A(\mathsf{h})$ conforms to (2). Now, take $\gamma \in \mathcal{H} \subset \boldsymbol{wp\text{-}\Gamma}$, with $N(\gamma) = \bigcup_{i=1}^{n} X_i \subset \mathbf{U}$, such that, for all $a \in A(\gamma)$: either

$$a = a^* = \{x_1, ..., x_n\}, x_i \in X_i$$

or

$$a = \{x\}, \text{ where } x \in N(\gamma) \setminus \bigcup_{j=1}^{r} a_j^*$$

. In this case, γ represents the relation $\mathsf{R} \subset \prod_{i=1}^{n} X_i$, where $(x_1, ..., x_n) \in \mathsf{R} \Leftrightarrow (\exists 1 \leq j \leq r)\, [\{x_1, ..., x_n\} = a_j^*]$. □

3.2. *A priori unbounded data structures*

Recursively defined data-structures may have any number of elements or parts, so the analysis hereinbelow will, by means of (generally accepted) definitions of these data-structures, recast them as *wp*-graphs through the recursion scheme in definition 2.1. As definitions usually vary somehow, the ones supporting this discussion are those contained in the directly referred material. This section has the following structure. Sub-classes of ***wp-Γ*** will be singled out by analogous definitions and then "matched" to equivalent classes of usual data-structures, by means of the structural similarity of their recursive constructions and a slight variation of the method used in Lemma 3.2 to immerse **U** in ***wp-Γ***. So, this proof act on definitions syntax rather than on the analysed objects themselves.

The subsequent denotations and assumptions for intervening extended hypergraphs will be useful. Naming by $\mathsf{h}(m)$ the elements of $\mathcal{H}$ for which $\#(N(\mathsf{h}) \setminus (\mathbf{V} \cup \{\odot\})) = m$, assume that

$$\mathsf{h}^{\circ}(m) \quad \Rightarrow \quad (\forall 1 \leq i \leq m)\,[\{\odot, n_i\} \in A\,(\mathsf{h}^{\circ}(m))], \tag{8}$$
$$_m\mathsf{h}^{*}(1) \quad \Rightarrow \quad (\forall 1 \leq i \leq m)\,[\{n, v_i\} \in A\,(_m\mathsf{h}^{*}(1))], \tag{9}$$
$$_m\mathsf{h}^{*}(0) \quad \Rightarrow \quad \{v_1,\ldots, v_m\} \in A\,(_m\mathsf{h}^{*}(0)), \tag{10}$$

in any reference to them henceforth, unless otherwise stated. Also, **U** shall match the "atoms" class of any usual definition.

The simplest recursive data-structures are linear lists or sequences, the class of which is denoted by $\mathcal{S}$. In [Bauer & Wössner 1982], four definition cases are distinguished: left and right sequences that allow or not for empty sequences. Their classes will be here denoted by $\mathcal{S}_l$ and $\mathcal{S}_r$ or $\mathcal{S}_l^{\lozenge}$ and $\mathcal{S}_r^{\lozenge}$, respectively, but note that sequences in the latter two are sequences in the former two, respectively, pre- or ap-pended with $\lozenge$. The sequence-like elements of ***wp-Γ*** are:

*Def 3.2 (Left wp-*graph *Sequences) An element of* ***wp-Γ*** *is a left sequential wp-*graph, $\mathsf{h}_s \in \mathbf{S}_l$, *if and only if*

- $\mathsf{h}_s = \mathsf{h}^{\circ}(1)$, *where* $\mathsf{h}^{\circ}(1) \notin \mathcal{H}^{*}$, *or*
- $\mathsf{h}_s = {}_1\mathsf{h}^{\circ}(1) \dashv \mathsf{s}$, *for some* $\mathsf{s} \in \mathbf{S}_l$.

and

*Def 3.3 (Right wp-*graph *Sequences) An element of* ***wp-Γ*** *is a right sequential wp-*graph, $\mathsf{h}_s \in \mathbf{S}_r$, *if and only if*

- $\mathsf{h}_s = {}_1\mathsf{h}\,(1)$, *where* ${}_1\mathsf{h}\,(1) \notin \mathcal{H}^{\circ}$, *or*
- $\mathsf{h}_s = \mathsf{s} \dashv {}_1\mathsf{h}^{\circ}(1)$, *for some* $\mathsf{s} \in \mathbf{S}_r$.

By adding an atom to **U**, say $\lozenge$, representative of void, and letting $\mathsf{h}\,(\lozenge)$ denote a $\mathsf{h}\,(1)$ type hypergraph which only element not in $\mathbf{V} \cup \{\odot\}$ is $\lozenge$, the *wp-*graph counterparts of $\mathcal{S}_l^{\lozenge}$ and $\mathcal{S}_r^{\lozenge}$ are obtained by exchanging $\mathsf{h}^{\circ}(1) \notin \mathcal{H}^{*}$, by $\mathsf{h}^{\circ}(\lozenge) \notin \mathcal{H}^{*}$, in definition 3.2, and ${}_1\mathsf{h}\,(1) \notin \mathcal{H}^{\circ}$, by ${}_1\mathsf{h}\,(\lozenge) \notin \mathcal{H}^{\circ}$, in definition 3.3, which is to say that these changes occur in $\mathbf{S}_l$ and $\mathbf{S}_r$, respectively.

Paraphrasing [Siklóssy 1976], Sexes, or, more commonly *s-expressions*, whose class is here denoted by $\mathcal{S}$ex, are the LISP data-structures par excellence. A counterpart class Sexi, of sex-*wp-*graphs, is defined as:

*Def 3.4 (wp-*graph *Sexes) An wp-*graph $\sigma \in$ ***wp-Γ*** *belongs to* Sexi *only in the following two cases:*

- $\sigma = \mathsf{h}^\circ(1) \notin \mathcal{H}^*$,
- $\sigma = {}_2\mathsf{h}^\circ(0) \lrcorner <\sigma_1, \sigma_2>$; *where* σ_1 *and* σ_2 *belong to* Sexi.

Sexes are indeed quite similar to binary trees, their difference being essentially that sexes have labels only on their leaves while binary trees have them on branching points [Bauer & Wössner 1982]. The *wp*-graph class, **binT**, analogous to binary trees is defined as follows:

*Def 3.5 (wp-*graph *Binary Trees) A* $\beta \in$ ***wp-Γ*** *belongs to its subclass* **binT** *only if either:*

- $\beta = {}_2\mathsf{h}^\bullet(1)$,
- $\beta = {}_2\mathsf{h}^\bullet(1) \lrcorner <\beta_1, \beta_2>$, *where* β_1 *and* β_2 *belong to* **binT**.

And not just binary trees counterparts are to be found in ***wp-Γ***, general trees too, the class of which is hereinbelow denoted by **T**.

*Def 3.6 (wp-*graph *Trees) A* $\beta \in$ ***wp-Γ*** *belongs to its subclass* **T** *only if either:*

- $\tau = {}_k\mathsf{h}^\bullet(1)$,
- $\tau = {}_k\mathsf{h}^\bullet(1) \lrcorner <\beta_1, ..., \beta_k>$, *where* $\tau_1, ..., \tau_k \in \mathbf{T}$ *and* $k \geq 2$.

It is to be noted that for trees, to prevent arcs at the same level, it is required that $A\,({}_k\mathsf{h}^\bullet(1)) = \{\{\odot, n\}, \{n, v_1\}, ..., \{n, v_k\}\}, k \geq 2$. Also, if k do not vary from a recursion step to another, i-e. $k = k_0$ throughout the construction, definition 3.6 produce $\boldsymbol{T}_{k_0}$, the class of trees that have a fixed number of sub-trees at each branching.

Proposition 3.1 (Recursive Structures)
For all above defined **wp-Γ** *sub-classes, there is a one to one correspondence with the elements of the equivalent data-structure class as usually defined.*

Proof Note that, for each of definitions from 3.2 to 3.6, the defined objects are obtained in the very same way as the corresponding definitions in [Bauer & Wössner 1982, sect. 2.9] or [Tennent 1981, sect. 3.2]. That is, a structure is the outcome either of a single particle from a universe or results from the application of a constructive *meta-operator* to structures pertaining to its class. Thus, they are all built from single particles in each class and, as the corresponding definitions have the same rules, except for the modelling usual data-structures' meta-elements by *wp*-graphs of the forms (8) thru (10).

It is hence enough to match elements of **U** to these particles to obtain the correspondence between elements of any corresponding classes. Recall that $N(\mathsf{h}(m)) \subset \mathbf{U}\odot$ and note that for all $\mathsf{h}(m)$ appearing in the previous definitions $m \leq 1$. Thus, whenever an atom appearing in a data-structure belongs to **U**, it can be taken as the unique element in $\mathsf{h}(m) \cap \mathbf{U}$, where $\mathsf{h}(m)$ intervenes in the corresponding ***wp-Γ*** definition, by the same 'lifting' argument of Lemma 3.2. The result then follows from recursive induction. □

4. *Conclusion*

The following must be observed about *wp*-graphs as above described.

As defined in section 2, *wp*-graphs are slightly less general than in [Kritz 1991]. The distinction is, however, subtle being of no relevance to the present work. Anyway, as here defined, they are indeed particular cases of the original ones and the inclusions remain valid. Also, the restriction $\# N(\mathsf{h}) < \infty$ (as well as $N(\mathsf{h}) \neq \{\odot\}$), present in that work too, is more of biological than mathematical nature, being at ease removed. Even so, ***wp-Γ*** does encompass programming structures, that are often considered finite due to computability requirements. Moreover, as seen in section 3.2, ***wp-Γ*** contains *wp*-graphs with a number of parts as large as pleased and indeed infinite in the limit. So, the above constrain is not that restrictive. Furthermore, note that hypergraphs can be made *directed* and this property hierarchically inherited by *wp*-graphs, what largely extends the data-structure classes that may be mapped into these mathematical objects, or greatly simplify the way of doing it. This is, however, beyond the scope of this note.

With the exception of arrays, the assertions and proves in section 3 refer to structures over **U**, but a slightly more careful inspection of the ***wp-Γ*** definition shows that any appearance of **U** elements therein could indeed be substituted by *wp*-graphs. This is a consequence of ***wp-Γ*** being "closed" under the binding operator $\lrcorner$, and the fact that all the class constructors involved in the foregone analysis are indeed particular cases of it. Also, the fork meta-concept of [Bauer & Wössner 1982] appearing in each non-sequential recursive structure is realized by an *element* of ***wp-Γ***. Therefore, no outside concept is involved in the definition of the *wp*-graph-counterparts of data-structures, as $\lrcorner$ is effectively the constructor of ***wp-Γ***. The proves of Lemma 3.7 and Proposition 3.1 above have been made quite terse, since the available space prevents a throughout discussion of all possible cases. However, the proof of Proposition 3.1 has essentially a metalinguistic character, associating the structures' formation rules and atoms.

Moreover, it collapses linguistic and meta-linguistic tokens in just one description level.

Finally, I must mention that, thanks to Bauer and Wössner, I've recently learned about Pratt's recursive graphs [Pratt 1969]. It is still soon to fully grasp all implications of some key differences between the two definitions: namely the use of hypergraphs instead of graphs and the employ of **V** and $\odot$ to establish a communication path between immediate recursion levels. This will be treated in future work.

The author heartily thanks the 2nd Gauss Symposium organizers for the warm reception and the opportunity to get in touch with so many new research fields.

e-mail:kritz@lncc.br

REFERENCES

J.M. Abe & N. Papavero, *Teoria Intuitiva dos Conjuntos*. Makron Books, Rio de Janeiro, 1992.

F.L. Bauer & H.Wöessner, *Algorithmic Language and Program Development*. Springer-Verlag, Berlin, 1982.

C. Berge, *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973. (transl. and rev. ed. from Graphes et Hypergraphes, Dunod, Paris, 1970.)

D.E. Knuth, *The Art of Computer Programming, v.1: Fundamental Algorithms, 2nd ed.*. Addison Wesley, Reading, 1973.

M.V. Kritz, *On Biology and Information*. P&D Report #025/91, LNCC/CNPq, Rio de Janeiro, Dec. 1991.

T.W. Pratt, *A Hierarchical Graph Model of the Semantics of Programs*. Proc. AFIPS Spring Joint Computer Conference 1969, 1969, pp. 813–825.

R. Scialom, *Personal Communication*. Dec. 1990.

L. Siklóssy, *Let's Talk LISP*. Prentice-Hall, Englewood Cliffs, 1976.

R.D. Tennent, *Principles of Programming Languages*. Prentice-Hall, Englewood Cliffs, 1981.