REPRESENTING TREE STRUCTURES BY NATURAL NUMBERS

Carmen Luengo, Luis Fernández, Fernando Arroyo

***Abstract**: Transformation of data structures into natural numbers using the classical approach of gödelization process can be a very powerful tool in order to encode some properties of data structures. This paper presents some introductory ideas in order to study tree data structures under the prism of Gödel numbers, and presents a few examples of using this approach to trees.*

***Keywords**: Data structures, Trees, Gödelization*

ACM Classification Keywords: *E.1 Data Structures - Trees*

Introduction

In [Luengo, 2008] was introduced a very interesting method in order to describe a Transition P system as a natural number. The process is based on the classical and very well known gödelization method. That paper described the way to find out the corresponding Gödel number to a transition P systems. A very important part of the process is to represent the membrane systems, a tree structure, as a natural number. In that paper the membrane structure was represented in a very naïf way, which it has been demonstrated to be not enough in order to recover the tree structure from the natural number that represents it. Following with this idea, in this paper it is depicted a sketch for assigning natural numbers to tree structures, which permits recover the tree from the natural number.

Tree definitions

A tree is an undirected simple graph ***T = (V, E)*** satisfying that it is connected and has no cycles.

A tree ***T*** is said to be directed if it is a directed graph which would be a tree if directions on edges are not considered.

A tree ***T*** is said to be rooted if one vertex is designed as root, in which case the edges have a natural orientation, towards or away from the root.

In rooted trees, the parent of a vertex is the vertex connected to it on the path to the root. Every vertex on the tree has a unique parent except for the root. A child of a vertex v is a vertex of which v is the parent. And finally, a leaf is a vertex without children.

Rooted trees are the key element to define tree data structure and, on this context, vertexes are usually denoted as node and edges are usually denoted as links.

In Membrane Systems literature, a membrane structure is defined by a language *MS* over the alphabet *{ [,] }*, as follows:

1. $[\,] \in MS$
2. if $\mu_1, \dots, \mu_n \in MS$ for some $n \geq 1$, then $[\mu_1, \dots, \mu_n] \in MS$

Once the language is established, in order to properly define membrane structure is needed to consider the following relation over the elements of *MS*: $m_1 \sim m_2$ if and only if we can write the two strings in the form $m_1=\mu_1\mu_2\mu_3\mu_4$, $m_2=\mu_1\mu_3\mu_2\mu_4$, for $\mu_1\mu_4 \in MS$ and $\mu_2, \mu_3 \in MS$. That is, if two neighbouring pairs of parentheses are interchanged, together with their contents, then we obtain the same membrane structure. If we represent by ~* the transitive and reflexive closure of the relation ~. Clearly we have defined an equivalence relation over MS and membrane structures are equivalence classes.

It is very easy to translate this concept into graph theory. A membrane structure is a rooted tree, where the nodes are called membranes, the root is called skin and the leaves are called elementary membranes.

Finally, a recursive definition of membrane structure coming from graph theory is given:

A membrane structure is:

1. $\mu_0 = [\ ]_0 \in MS$
2. $([\ ]_0, \mu_1, \dots, \mu_n)$ where $\mu_1, \dots, \mu_n \in MS$ for some $n \geq 1$

That is a membrane –the external one or skin- encloses a set of membrane structures. It is considered the same membrane structure every permutation of the different μ_i belonging to *MS*. Moreover, there is a unique link v_i connecting the membrane $[\ ]_0$ with the membrane structure $\mu_i = ([\ ]_i, \mu^i_1, \dots, \mu^i_n)$ where $\mu^i_1, \dots, \mu^i_n \in MS$ for some $n \geq 0$.

Now μ_0 is the ancestor of $\mu_1, \dots, \mu_n$ and each one of them are connected by the link v_i. Obviously, the definition is extended to the other internal membrane structures in a natural way.

Let us say that every membrane system considered here has a finite number of membranes, $m \geq 1$.

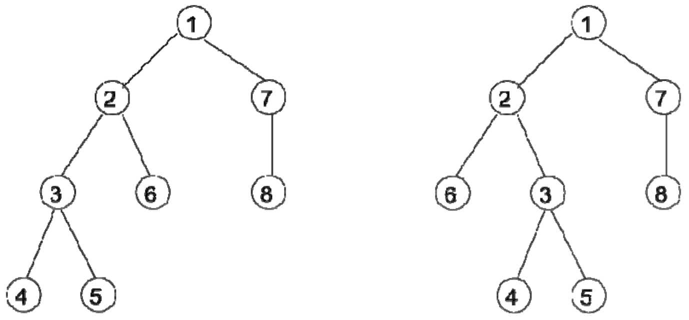

Figure 1: Two equivalent membrane structures

Tree Numbering Process

The purpose of the numbering process is to assign a natural number to each one of the different trees with m nodes or membranes and given a natural number associated to a tree with m nodes to recover the tree structure.

In [Luengo, 2008] we defined in a very simple way to assign natural numbers to membrane structure. The process was the following:

Let ***T* = (*N, E*)** be a tree data structure with m nodes labelled in $L=\{1,2, \dots, m\}$, and let $P_m = \{p_1, p_2, \dots, p_m\}$ be the set of the m first natural number starting in 2.

For the set of nodes of the tree ***T***, it is defined the following map:

$$\begin{aligned} G_m &: N \to P_m \\ n_k &\to p_k, \forall k \in L \end{aligned} \quad (1)$$

From this map it was defined the number associated to a tree *T* with labels in *L* as follows:

$$G_m(\mathrm{T}) = \prod_{k=1}^{m} G_m(n_k), \forall k \in L \tag{2}$$

It is very easy to see that the process is not useful to extract the tree structure. In fact, it is possible to have several trees to which the map G_m gives the same natural number.

If we want to recover the tree structure from a natural number provide by a G_m map, it is needed to consider some particularities of the tree structure such as: number of branches, depth of each one of the branches, etc. Following with these ideas, we are going to consider branches and depth in order to assign natural number to nodes of the tree.

First of all we are going to map links instead of nodes as follows:

Let ***T = (N, E)*** be a tree data structure with m nodes labelled in *L={1,2, …, m}*, and let $P_{m-1} = \{p_1, p_2, \ldots, p_{m-1}\}$ be the set of the m-1 first natural number starting in 2. Every edge or link is numbered in pre-order starting from left to right in depth. This process is shown in figure 2.

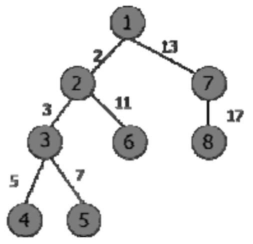

Figure 2: Numbering vertexes

As we did before, for the set of vertexes of the tree T, we define the map:

$$\begin{aligned} &G_m : E \to P_{m-1} \\ &e_k \to p_k, \forall k \in L - \{m\} \end{aligned} \tag{3}$$

And then, for the set of nodes of the tree T, it is defined the following map:

$$\begin{aligned} &G_m : N \to P_{m-1} \\ &\quad n_k \to G(n_k), \quad \forall k \in L \text{ where :} \\ &G_m(n_k) = \begin{cases} p_k \text{ if } n_k \text{ is a direct descendent of the skin or root node} \\ p_k p_i G_m(n_j) \text{ if } n_k \text{ is a direct descendent of } n_j \text{ by the link } e_k \\ \qquad \text{and } n_k \text{ is a node of the subtree } \mu_i \end{cases} \end{aligned} \tag{4}$$

Figure 3 shows this process.

This process permits to embedded information about tree structure into each one of the number associated to nodes, in particular, each one of the leaves of the tree have encoded the information corresponding to the branch to which they belong and the depth of each one of them.

Finally, it is possible to assign a natural number to the tree taking into account the leaves nodes of the tree as follows:

Let $T=(N, V)$ be a tree with m nodes and the set of leaves $L_{eaves} = \{l_1, l_2, ..., l_s\}$, $1 \le s \le m$ then the natural number associated to T is defined be the expression:

$$G_m(T) = \prod_{l_k \in L_{eaves}} G_k(l_k) \tag{5}$$

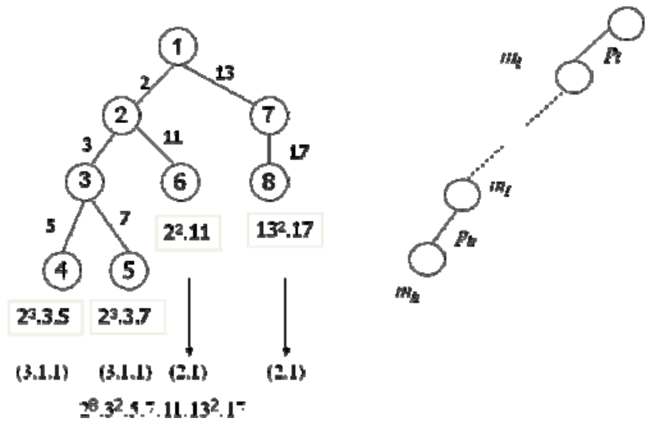

Figure 3: numbering nodes

This process permits to have some information about the tree structure, that is:

- For each one of the leave nodes, the last prime factor is the label corresponding to the link which connects the node with its father.
- The exponent of the minimum prime corresponds to the depth of the leave in the tree.
- Two equivalent trees have different natural number associated, which permits recover tree information independent of the equivalent relation defined before.

However when all the information is collected into only one unique natural number for the tree, there is some lost of information which is necessary in order to recover tree structure in the inverse process. In particular it is needed not only the number of branches the tree has, but also it is needed to know the depth of each one of them.

Figure 4 show different trees with four leaves with different structures depending on the number of nodes and branches.

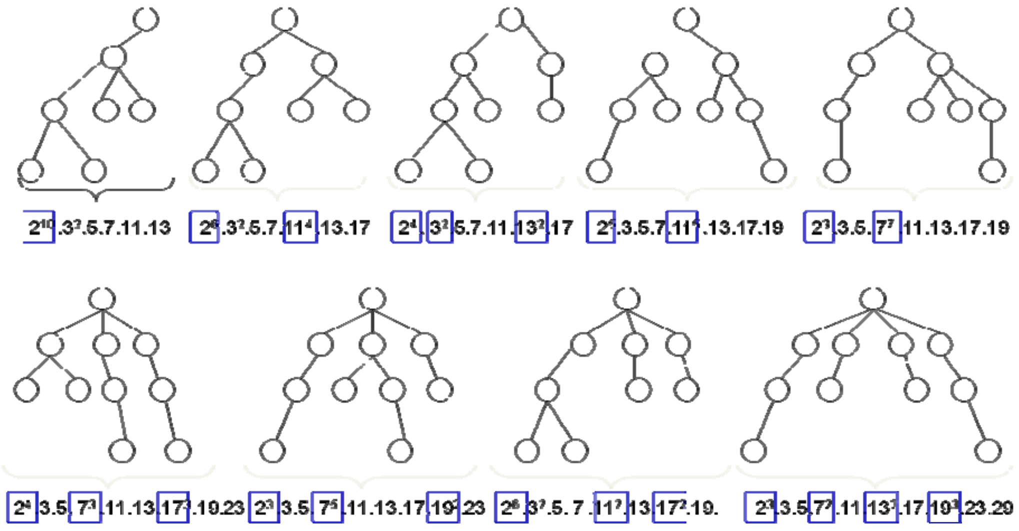

Figure 4: Trees whit 4 leaves

As it was said before, what is important for recovering tree structure information is the number of leave nodes on the tree and, also, the depth of each one of them. If we incorporate these two data into the natural number associated to the tree, then it is possible to recover the tree structure from that number; and it is not very difficult.

Incorporating the information of number of leaves and their corresponding depth into expression (5), we have:

Let *T=(N, E)* be a tree with m nodes, let *L={l₁, l₂, ..., lₛ}*, $1 \le s \le m$ be the set of leaves, let *D={d₁, ...,dₛ}* be the corresponding depth for each one of the branches of the tree *T*, let P_m the set of the m-1 first prime number, and let p_0 a prime number such as $p_0 \notin P_{m-1}$; then the natural number associated to *T* is defined be the expression:

$$G^{p_0}{}_m(T) = p_0^{p_1^{d_1} \cdots p_s^{d_s}} \prod_{l_k \in L} G_m(l_k) \qquad (6)$$

Where G_m for nodes is defined by expression (4)

Recovering tree structure

In order to build a tree from a natural number, there is needed that the number satisfied some constrains, that is:

- Its factorization into prime numbers has to have m different prime factors
- p_0 has to be known

Let $G^{p0}{}_m$ be a natural number which encodes a tree *T* with m-1 vertexes and m nodes, let p_0 be the prime which define the tree structure and let P_{m-1} be the rest of primes appearing in the factorization of $G^{p0}{}_m$; then this number can be also expressed by:

$$G^{p_0}{}_m(T) = p_0^{p_1^{d_1} \cdots p_s^{d_s}} \prod_{i \in \{1,\ldots,m-1\}} p_i^{e_i} \qquad (7)$$

From this prime factorization, it is possible to recover the tree structure as follows:

- Primes $\{p_i \mid 1 \le i \le m-1\}$, are ordered in an increasing way.
- Exponent of the p_0 factor encodes the number of leaves in the tree and the depth of each one of them.
- For each one of the rest of prime factor the exponent of the factor tell us about its position on the tree, factors having exponent equals to one are links to leaves nodes on the tree, or links to intermediate nodes without branches, because they appear only once; the rest of factor tell us the number of times the link is counted in order to encode the tree structure.

Let us to show some examples about these facts, for instance, considering the following number:

$$G_6{}^{13}(T) = 13^{2^3 3^3} . 2^6 . 3 . 5 . 7 . 11$$

The number establish the tree structure as follows: the prime factor 13 determines that the tree has 2 leaves with 3 nodes each one. The rest of prime factors explicit the rest of the tree structure as follows:

- The root node is connected directly by a link labelled with the prime number 2 to the next node by the left. This is the first step in order to form the first path from the node to the leave which is two more levels below. In order to build the path it is needed two more nodes connected by one more link labelled with the prime 5. This path consumes 3 times the number 2, 1 time the number 3 and 1 time the number 5. Hence we have now 3 times prime number 2, 0 times numbers 3 and 5 and 1 time numbers 7 and 11 in order to build the second path from the root to the second leave of the tree.
- The second path starts in the root node and how we have 3 times the number 2, so the path has also to pass by the node connected to the root by the link labelled with 2 in order to consume the rest of 2’s we

have. Now, the path cannot pass by the link labelled with number 3, we have no more 3’s, so we need to open a second way and we consume one new prime, the next one in the factorization which is 7, and finally, we arrive to the last node of the tree trough a new link labelled with the 11 prime number. On this process we have consumed the rest of primes we have in the factorization, ending the building process of the tree.

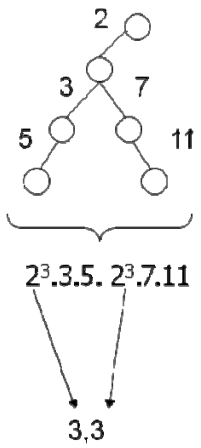

Figure 5: The built tree associated to G6 13(T)

Next figure shows some other examples of this process of recovering tree structures from a given natural number. In that figure it is considered that there are three leaves which are at depth of 4, 4 and 2 from the root node. In that figure is possible to see how important the multiplicity of the prime factors is in order to recover tree structures.

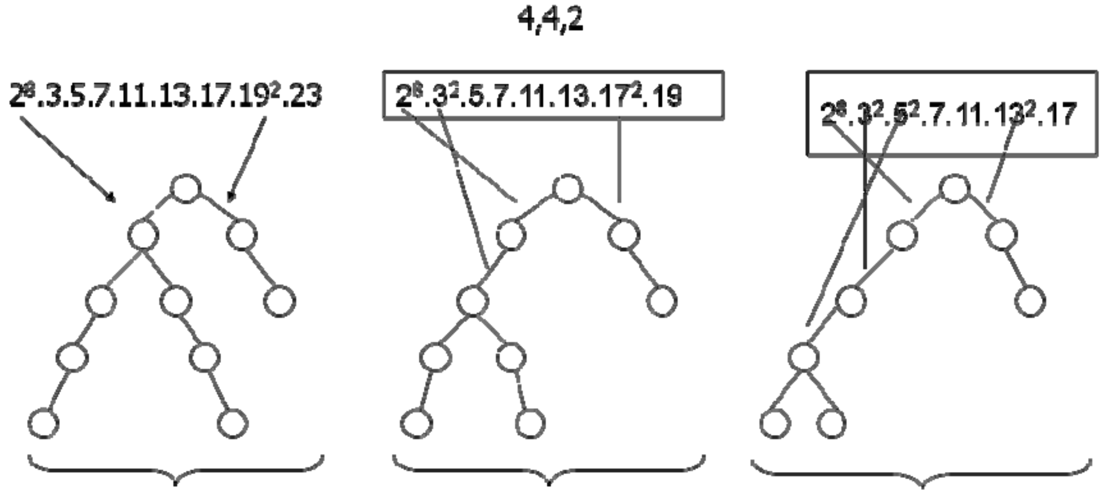

Figure 6: Several tree structures with 3 leaves and same depth

In terms of determining the general process for building a tree from a given natural number $G^{p0}{}_m$ like in (7) we propose the following algorithmic description:

G^{p0}_m provides the following data:

- s = number of leaves in the tree.
- $D = \{d_1, ..., d_s\}$ the set of number determining the depth for each one of the leaves.
- $P = \{(p_1, e_1), ..., (p_{m-1}, e_{m-1})\}$ the set of prime factor with their multiplicities ordered in an increasing way.

Then we consider the initial tree T to be built defined by:

$$T = (N=\{1\},L=\{\})$$

The idea is to add nodes linked by edges labelled by paired of natural number, the corresponding to the initial node and the final node as it is shows in Figure 7.

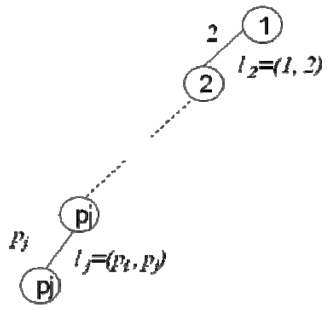

Figure 7: labeling nodes and links in the building process

The algorithm for building the tree from the natural number G^{p0}_m is the following:

```
for (i=1; i ≤ s; i++) {
        n_pos = 1;
        for (j=1 ; j ≤ d_s ; j++) {
                take ( p_j , e_j) ∈ P;
                if ( (n_pos , p_j) ∉ L)
                        then {
                                N = N ∪ {p_j};
                                L = L ∪ {(n_pos , p_j)};
                        }
                e_j --;
                n_pos  = p_j ;
        }       //end for_j
        P_m-1 = P_m-1 – { (p_i , 0) | 1 ≤ m-1 };
}       //end for_i
```

Conclusions

This paper describes a way in order to transform tree structures into natural number. The process is based on the very well known gödelization process. Moreover, the process can be inverted in order to generate a tree from a natural number accomplishing several constrains. However, the paper is only introductory in this field; it is needed to follow up with the study in order to determine the real power of this approach in the study of tree structures. There are several open questions such as:

- Is it possible to determine that two trees are equivalent when they are represented by natural numbers?
- What happens with insertion / deletion of nodes on the tree on the natural number?
- Etc.

One more problem comes from dealing with big natural numbers. We are aware that the whole process is supported by natural number with a big digit number, and that is a problem itself in many cases.

This paper is only one preliminary study of coding tree by numbers, and we hope to continue our work with significant results and to apply to membrane computing and other natural computing fields in which such data structures are relevant.

Bibliography

[Luengo 2008] Carmen Luengo, Luís Fernández, Fernando Arroyo; P systems Gödelization, International Book Series Number 1, pp 29 -34, ISSN 1313-0455, Bulgaria

[Păun 2002] Gh. Păun, Membrane Computing. An Introduction, Springer-Verlag, Berlin, 2002

[Suzuki 2000] Y. Suzuki, H. Tanaka, On a LISP Implementation of a Class of P Systems, Romanian J. of Information Science and Technology, 3, 2 (2000), 173-186.

[Turing 1936] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 2-42, (1936-7), 230-265

[P system Web page] http://ppage.psystems.eu/ (july 2010)

Authors' Information

Carmen Luengo Velasco – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid; Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: cluengo@eui.upm.es*

Luis Fernández Muñoz – *Dpto. Lenguajes, Proyectos y Sistemas Informáticos de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid; Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: setillo@eui.upm.es*

Fernando Arroyo Montoro - *Dpto. Lenguajes, Proyectos y Sistemas Informáticos de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid, Ctra. Valencia, km. 7, 28031 Madrid (Spain); e-mail: farroyo@eui.upm.es*