PUC

Série: Monografias em Ciência da Computação

Nº 17/77

# A RELATIONAL MODEL TOWARDS THE SYNTHESIS OF DATA STRUCTURES

by

Roberto Lins de Carvalho
Antonio Luz Furtado
Atendolfo Pereda Bórquez

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225 — ZC 19
Rio de Janeiro — Brasil

Sẽrie: Monografias em Ciência da Computação

Nº 17/77

# A RELATIONAL MODEL TOWARDS THE SYNTHESIS OF DATA STRUCTURES*

by

Roberto Lins de Carvalho
Antonio Luz Furtado
Atendolfo Pereda Bõrquez

Series Editor: Michael F. Challis Nov. ,1977

* Work Partially supported by the Brazilian Governement Agency FINEP

ABSTRACT:

Data structures are defined by means of relations, and the allowable operations on them are expressed in terms of a few "well-structured" primitive operations.

KEY WORDS:

Data structures, relations.

RESUMO:

Define-se estruturas de dados por meio de relações, e as operações permissíveis sobre aquelas são expresas em termos de umas poucas operações primitivas "bem-estruturadas".

PALAVRAS-CHAVE:

Estrutura de dados, relações.

## CONTENTS

# 1. INTRODUCTION

To understand the essential aspects of what constitutes a particular class of data structures, we must abstract these aspects from our intuitive ideas about it. This abstraction must be unambiguous, modelling the semantics of the data structure in the most adequate form possible. Because of reasons such as the possibility of establishing the correctness of the implementation of a specific data abstraction, and the need for precision in the communication medium, these abstractions must be formalized.

In this work, a relational model was chosen as the general base for the specification of data abstractions. Each particular kind of data structure can be obtained from the elements of the domain of the model by adding some restrictions. These restrictions characterize the static aspects of the data structure being specified and they must be expressed in a precise form: for example, using axioms in a first order language with equality.

The operations on each data structure give its dynamic aspects. The operations are defined using a set of "well-structured" primitive operations on the elements of the domain.

In chapter 2 the basic definitions are presented. In chapter 3 the potential of the model is illustrated with examples of the characterization of some well-known data structures.

## 2. BASIC CONCEPTS

We will consider a finite family $\{D_i\}_{0<i<n}$ of objects such that $D_i \cap D_j = \phi$ for $i \neq j$ and $D_0 = \{*\}$. The union of the objects of this family will be called the universe of data and will be denoted by $D = \bigcup_{i=0}^{n} D_i$. We will also consider a finite set $L = \{\ell_1, \ell_2, \ldots, \ell_m\}$ of objects called function labels and $L_t$ of temporary labels. $L_a$ is to be a considered as a linearly ordered set.

Let D and $L$ be the sets defined above. We use T to denote a relation $T \subseteq D \times L$, called a data relationship.

Definition 1.1. We define the following functions:

$$\underline{pr}_i: T \to D \quad i = 1,2$$

$$\underline{pr}_3: T \to L$$

by $\underline{pr}_i\,(<x_1, x_2, x_3>) = \underline{pr}_i(\underline{x}) = x_i, \quad i = 1,2,3$

$$\underline{pr}_{i,j} = <\underline{pr}_i, \underline{pr}_j>, \quad 1 \leq i, j \leq 3$$

$$T_{i,j} = \underline{pr}_{i,j}(T) \text{ (the image of T under } \underline{pr}_{ij})$$

$$T^0_{i,j} = \phi$$

$$T^1_{i,j} = T_{i,j}, \quad T^{n+1}_{i,j} = T^n_{i,j} \cdot T_{i,j}$$

where the operator . is the product of relations.

Lemma 1.1. There is an integer N such that

$$T^N_{i,j} = T^n_{i,j} \quad \text{for some } n<N$$

N is called the depth of $T_{i,j}$.

Proof  It is a consequence of D and L being finite. Then $T_{i,j}$ is finite because it is a subset of the finite sets DxD, DxL or LxD. Consider that we have calculated $T^m_{i,j}$, $m \geq 1$; it is clear that, if in $T^m_{i,j}$ there are not two pairs $<x_i,y_i>$ and $<x_j,y_j>$ with $y_i = x_j$, then $T^{m+1}_{i,j} = \emptyset$. When the previous condition does not apply, then the sequence $T^1_{i,j}$ , $T^2_{i,j}$,... is periodic, i.e. there exists some m such that $T^{m+1}_{i,j} = T^n_{i,j}$ for $m+1 > n$.

Definition 1.2. $\hat{T}_{i,j} = \bigcup_{k=1}^{N} T^k_{i,j}$ where N is the depth of $T_{i,j}$.

Definition 1.3. A data work space is a 5-tuple

$$D_w = <D, D_f, L, T, *>$$

where:

D is a universe of data

L is a set of labels

T is a data relationship, $T \subseteq D \times D \times L$

$D_f$ is the subset of D defined by

$D_f = \{y \in D \mid \forall \underline{x} (\underline{x} \in T \Rightarrow \underline{pr}_2(\underline{x}) \neq y)\} - \{*\}$

i.e. $D_f$ is our available space for obtaining new "nodes",

* is the only object in $D_0$, which is called the root.

and such that

(i) For all $\underline{x},\underline{y}$ T

if $\underline{pr}_{1,3}(\underline{x}) = \underline{pr}_{1,3}(\underline{y})$ then $\underline{x} = \underline{y}$ i.e. the second component of $\underline{x}$ is equal to the second component of $\underline{y}$.

(ii) For all $y \in ((D-D_f) - \{*\})$, there is $\underline{x} \in \hat{T}_{1,2}$ such that $\underline{pr}_1(\underline{x}) = \{*\}$ and $\underline{pr}_2(\underline{x}) = \{y\}$, i.e. for all y,y it is reachable from the root.

Definition 1.4. Let $D = <D,D_f,L,T,*>$ be a data work space. Then we define the relational data space, associated with $D_w$ to be the data work space $<D-D_f,\phi,L,T,*>$

Example 1.1.

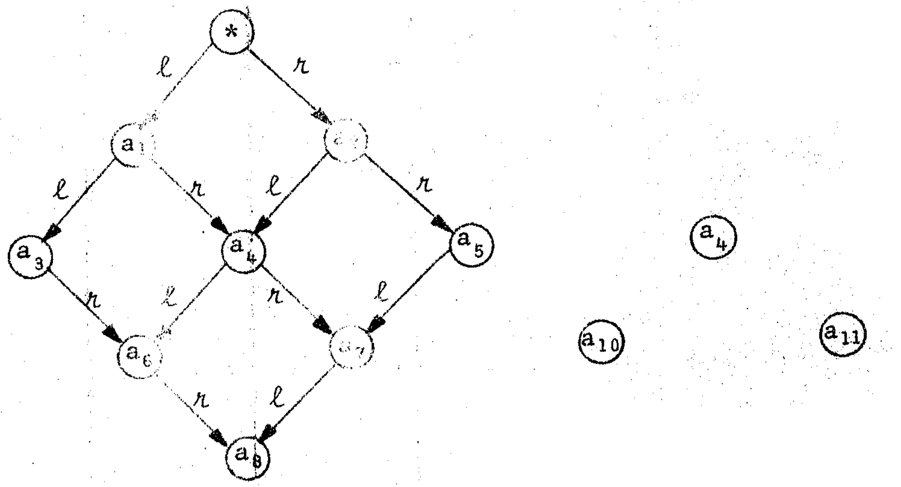
$D = \{*, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$

$D_f = \{a_9, a_{10}, a_{11}\}$

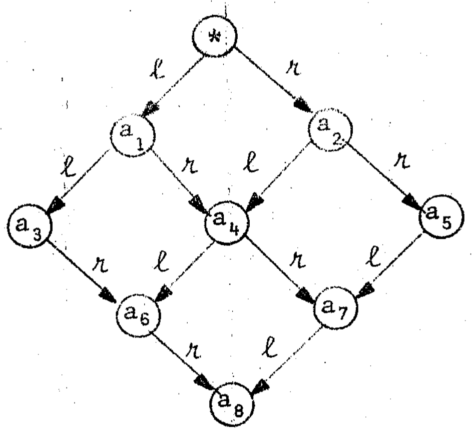
$L = \{\ell, r\}$

$T = \{(*, a_1, \ell), (*, a_2, r), (a_1, a_3, \ell), (a_1, a_4, r),$
$(a_2, a_4, \ell), (a_2, a_5, r), (a_3, a_6, r), (a_4, a_6, \ell),$
$(a_4, a_7, r), (a_5, a_7, \ell), (a_6, a_8, r), (a_7, a_8, \ell)\}$

Using a labelled digraph representation, the data work space is



and the associated relational data space is



$$T^1_{1,2} = T_{1,2} = \langle \underline{pr}_1, \underline{pr}_2 \rangle (T)$$

$$= \{(*,a_1), (*,a_2), (a_1,a_3), (a_1,a_4), (a_2,a_4), (a_2,a_5), (a_3,a_6), (a_4,a_6), (a_4,a_7), (a_5,a_7), (a_6,a_8), (a_7,a_8)\}$$

$$T^2_{1,2} = T^1_{1,2} \cdot T^1_{1,2} = \{(*,a_3), (*,a_4), (*,a_5), (a_1,a_6), (a_1,a_7), (a_2,a_6), (a_2,a_7), (a_3,a_8), (a_4,a_8), (a_5,a_8)\}$$

$$T^3_{1,2} = T^2_{1,2} \cdot T^1_{1,2} = \{(*,a_6), (*,a_7), (a_1,a_8), (a_2,a_8)\}$$

$$T^4_{1,2} = T^3_{1,2} \cdot T^1_{1,2} = \{(*,a_8)\}$$

$$T^5_{1,2} = 0$$

Then, the depth N, for this example is equal to 4.

<u>Definition 1.5.</u> Let $D = \langle D\text{-}D_f, \phi, L, T, *\rangle$ be a relational data space. We define the set of triples:

a) $T_\ell = \underline{pr}_{1,2}\,(T_\ell \cap (D^2 \times \{\ell\})),\ \ell \in L_a$

i.e. $T_\ell$ is the set of triples such that the first and second components of each triple are related by the label $\ell$ in T.

b) $T^1_\ell = T_\ell,\ T^{n+1}_\ell = T^n_\ell \cdot T_\ell, \ell \in L_a$

c) $T^n_{a,\ell} = (\{a\} \times D) \cap T^n_\ell\ ,\ a \in D$

i.e. $T^n_{a,\ell}$ is the set of triples such that the first component is $a$, and the second component is related to $a$ by a path formed by a sequence of n labels $\ell$.

<u>Lemma 1.2.</u> There is an integer $N_a$ such that $T^{N_a}_{a,\ell} = T^n_{a,\ell}$ for $n<N_a$ or

$$T^{N_a+1}_{a,\ell} = \emptyset$$

<u>Proof:</u> it is similar to the proof of Lemma 1.1.

Definition 1.6. Let $\sigma$ be a sequence of labels. We define $T_\sigma$ by:

if $\sigma = \ell$ then $T_\sigma = T_\ell$
if $\sigma = \sigma'\ell$ then $T_\sigma = T_{\sigma'\ell} = T_{\sigma'} \cdot T_\ell$

for the sequence $\lambda$ of length zero we define $T_\lambda = \{< *, *>\}$.

Definition 1.7. Let $\sigma$ be a sequence of labels. A sequence of labels $\sigma_1$ is an initial sub-sequence of $\sigma$, denoted $\sigma_1 < \sigma$, iff

$$\sigma_1 = \ell_{i1}\ell_{i2} \ldots \ell_{ik},\ \sigma = \ell_{j1}\ \ell_{j2}, \ldots, \ell_{jm} \quad \text{such that}$$

$$\ell_{i1} = \ell_{j1}, \ell_{i2} = \ell_{j2}, \ldots, \ell_{ik} = \ell_{jk}, \text{ and } \quad m>k.$$

Definition 1.8. $T_{*,\sigma} = (\{*\} \times D) \cap T_\sigma$

i.e $T_{*,\sigma}$ is the set of couples such that the first component is the root $*$, and the second component is related to $*$ by the label path $\sigma$.

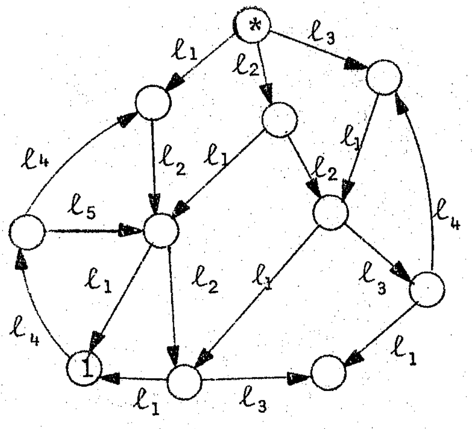
Fact: $T_{*,\sigma}$ is a singleton, because of the first condition on data work spaces.

Definition 1.9. Let $D = <D-D_f, \emptyset, L, T, *>$ be a relational data space. We define the set $S^d$ of useful sequences of labels(useful paths) from the root to $d \in (D-D_f)$ by

$$S^d = \{\sigma \in T_{*,\sigma} \mid \underline{pr}_2\ (T_{*,\sigma}) = \{d\} \text{ and}$$

$$\forall_{\sigma_1}\ (\sigma_1 < \sigma \Rightarrow \underline{pr}_2\ (T_{*,\sigma_1}) \neq \{d\})\}$$

Example 2.0:



$$S^1 = \{\ell_1\ell_2\ell_1,\ \ell_2\ell_1\ell_1,\ \ell_2\ell_1\ell_2\ell_1,\ \ell_2\ell_2\ell_1\ell_1,\ \ell_1\ell_2\ell_2\ell_1,\ \ell_3\ell_1\ell_1\ell_1\}$$

An example of a path that is not a useful path is $\ell_2\ell_1\ell_1\ell_4\ell_5\ell_1$.

<u>Definition 1.10</u>. Let $D = \langle D\text{-}D_f, \emptyset, L, T, *\rangle$ be a relational data space. S is defined by

$$S = \Big(\bigcup_{d \in D-D_f} S^d\Big) \cup \{\lambda\}$$

<u>Definition 1.11</u>. Let D be a universe of data and L a set of labels. We define the initial relational data space over D and L by

$$D_{in} = \langle \{*\},\ D - \{*\},\ L,\ \emptyset,\ *\rangle$$

We denote by $T_{D,L}$ the family of all relational data spaces defined over D and L.

## 2.1. <u>PRIMITIVE PREDICATES</u>

The family $T_{D,L}$ of relational data spaces is to be considered as a dynamic entity. Initially we have only $D_{in}$ , and we will form a new relational data space to be incorporated in $T_{D,L}$ by the use of certain primitive operations, to be defined. Before we do this, we will introduce certain predicates, or conditions over the useful paths of a relational data space $\langle D, D_f, L, T, *\rangle$.

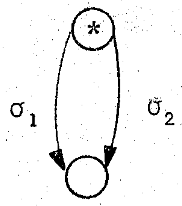
a) <u>end</u> $(\sigma, T)$ means that the element $d \in D$, reachable via the useful path $\sigma$ from the root $*$ of the relational data space is not the beginning of any path.
Formally:

$$\underline{\text{end}}\ (\sigma\ ,\ T) \text{ iff } \underline{\text{pr}}_2\ (T_{*,\sigma}) \subseteq \underline{\text{pr}}_1(T)$$

$$\begin{tikzcd} * \arrow[d, "\sigma"] \\ d \end{tikzcd}$$

b) alt $(\sigma_1, \sigma_2, T)$ means that the element of D, reachable by the path $\sigma_1$, is reachable by the path $\sigma_2$.

alt $(\sigma_1, \sigma_2, T)$ iff pr $(T_{*,\sigma_1}) =$ pr $(T_{*,\sigma_2})$

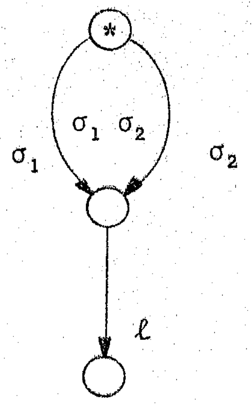


c) confl $(\sigma_1, \sigma_2, T)$ means that $\sigma_1$ and $\sigma_2$ are useful paths from the root, sharing at least the last label.
Formally:

confl $(\sigma_1, \sigma_2, T)$ iff $\underline{pr}_2(T_{*,\sigma_1}) = \underline{pr}_2(T_{*,\sigma_2})$ and

$\sigma_1 = \sigma_1' \ell$ and $\sigma_2 = \sigma_2' \ell$

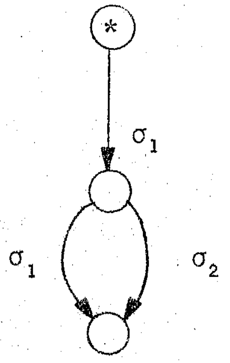
and $\underline{pr}_2(T_{*,\sigma_1'}) = \underline{pr}_2(T_{*,\sigma_2'})$



d) byp $(\sigma_1, \sigma_2, T)$ means that the element of D, reachable by the path $\sigma_1, \sigma_2$, can be found from the point reachable by $\sigma_1$ following a path distinct from $\sigma_2$.

byp $(\sigma_1, \sigma_2, T)$ iff there is a path $\sigma_3$ such that $\sigma_3 \neq \sigma_2$ and no subpath of $\sigma_3$ is the same as a subpath of $\sigma_2$ and

$$\underline{pr}_2\,(T_{*,\sigma_1\,\sigma_2}) = \underline{pr}_2\,(T_{*,\sigma_1\,\sigma_3})$$



The definition of <u>alt</u> $(\sigma_1,\sigma_2,T)$ does not exclude the possibility that $\sigma_1$ and $\sigma_2$ share at least the last label, that is for the same $\sigma_1$ and $\sigma_2$, <u>confl</u> $(\sigma_1,\sigma_2,T)$ could be true.

## 2.2. PRIMITIVE OPERATIONS

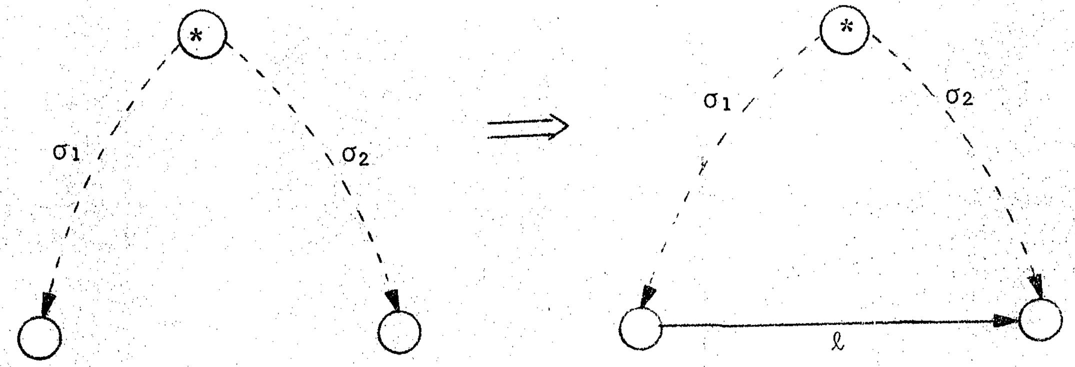
Now let us define the primitive operations of our system.

a) $\underline{ins}_1 : S \times S \times L \times T_{D,L} \rightarrow P\,(D \times D \times L)$

defined by:

$$\underline{ins}_1(\sigma_1,\sigma_2,\ell,T) \begin{cases} T \cup (\underline{pr}_2\,(T_{*,\sigma_1}) \times \underline{pr}_2(T_{*,\sigma_1}) \times \{\ell\}) \text{ if} \\ \qquad \underline{pr}_2\,(T_{*,\sigma_1}) \not\subseteq \underline{pr}_1\,(T_\ell) \\ T \quad \text{otherwise} \end{cases}$$

Intuitively, this operation can be represented by:

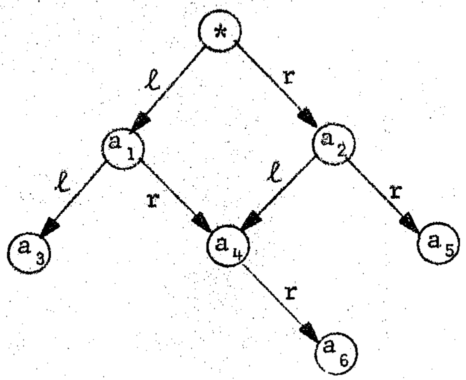


noting that the condition ensures that there will be no two edges with the same label leaving any node.

Example 2.1.

With T as follows, we will create a new link between "nodes" $a_5$ and $a_6$.

$T = \{<*, a_1, \ell>, <*, a_2, r>, <a_1, a_3, \ell>, <a_1, a_4, r>, <a_2, a_4, \ell>, <a_2, a_5, r>, <a_4, a_6, r>\}$.



$\sigma_1 = r.r$

$\sigma_2 = \ell.r.r$

$T_\ell = \underline{pr}_{1,2}\ (T \cap (D^2 \times \{\ell\})) = \{<*, a_1>, <a_1, a_3>, <a_2, a_4>\}$

$T_r = \underline{pr}_{1,2}\ (T \cap (D^2 \times \{r\})) = \{<*, a_2>, <a_1, a_4>, <a_2, a_5>, <a_4, a_6>\}$

$$T_{\sigma_1} = T_r \cdot T_r$$

$$T_{\sigma_1} = \{<*, a_5>, <a_1, a_6>\}$$

$$T_{\sigma_2} = T_\ell \cdot T_r$$

$$T_{\sigma_2} = T_\ell \cdot \{<*, a_5>, <a_1, a_6>\} = \{<*, a_6>\}$$

$$T_{*,\sigma_2} = (\{*\} \times D) \cap T_{\sigma_2} = \{<*, a>\}$$

$$T_{*,\sigma_1} = (\{*\} \times D) \cap T_{\sigma_1} = \{<*, a_5>\}$$

$$\underline{pr}_1 (T_\ell) = \{*, a_1, a_2\}$$

$$\underline{pr}_2 (T_{*,\sigma_1} = \{a_5\}$$

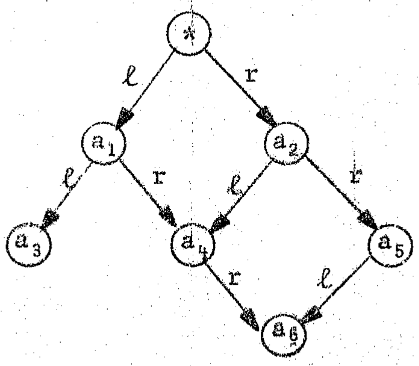
Thus

$$\underline{pr}_2 (T_{*,\sigma_1}) \subseteq \underline{pr}_1 (T_\ell)$$

then

$$\underline{ins}_1 (\sigma_1, \sigma_2, \ell, T) = T \cup (\underline{pr}_2 (T_{*,\sigma_1}) \times \underline{pr}_2 (T_{*,\sigma_2}) \times \{\ell\}) = T \cup \{<a_5, a_6, \ell>\}$$

that is, we obtain the "structure"

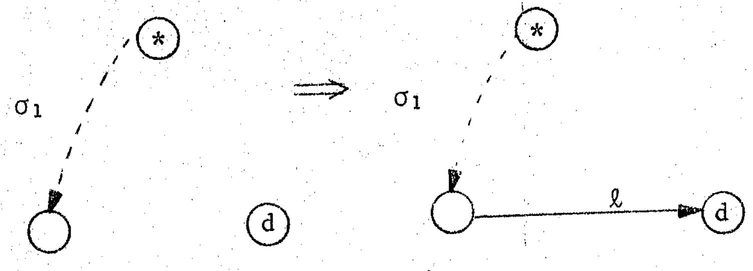


b) $\underline{ins}_2$: $S \times D_f \times L \times T_{D,L} \to P(D \times D \times L)$

defined by

$$\underline{ins}_2(\sigma_1, d, \ell, T) = \begin{cases} T \cup (\underline{pr}_2(T_{*,\sigma_1}) \times \{d\} \times \{\ell\}) & \text{if} \\ \quad \underline{pr}_2(T_{*,\sigma_1}) \not\subseteq \underline{pr}_1(T_\ell) & \\ T & \text{otherwise} \end{cases}$$

Intuitive representation:

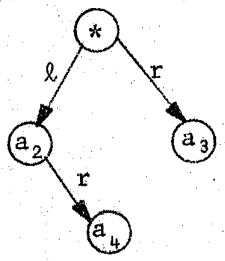


The condition is the same as for $ins_1$. The difference between the two operators is that, in $ins_2$, the destination node belongs to $D_f$ before the operation takes place.

Exemple 2.2. We will insert a new node, b, to the "right" of node $a_4$.

$T = \{<*, a_2, \ell>, <*, a_3, r>, <a_2, a_4, r>\}$

$D_f = \{b\}$



$\sigma_1 = \ell r$

$T_{\sigma_1} = T_\ell \cdot T_r$

$= T_\ell \cdot \{<*, a_3>, <a_2, a_4>\}$

$= \{<*, a_2>\} \cdot \{<*, a_3>, <a_2, a_4>\}$

$= \{<*, a_4>\}$

and

$T_{*,\sigma_1} = T_{\sigma_1}$

$\underline{pr}_1\ (T_r) = \{*,\ a_2\}$

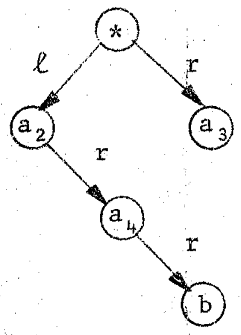
$\underline{pr}_2\ (T_{*,\sigma_1}) = \{a_4\}$

because $\underline{pr}_2\ (T_{*,\sigma_1}) \not\subseteq \underline{pr}_1\ (T_r)$ then

$\underline{ins}_2\ (\sigma_1, b,\ r,\ T) = T \cup (\underline{pr}_2\ (T_{*,\sigma_1}) \times \{b\} \times \{r\})$

$= T \cup \{\langle a_4,\ b,\ r\rangle\}$

i.e.



c) $\underline{del}$: $S \times L \times T_{D,L} \to P(D \times D \times L)$

defined by:

$$\underline{del}\ (\sigma_1,\ \ell,\ T) = \begin{cases} T - (\underline{pr}_2\ (T_{*,\sigma_1}) \times \underline{pr}_2 (T_{*,\sigma_1\ell}) \times \{\ell\}) & \text{if} \\ \quad \text{end } (\sigma_1\ell,\ T) & \\ \text{or exists } \sigma_2 \text{ s.t. if } \underline{alt}\ (\sigma_1\ell, \sigma_2, T) & \\ \quad \text{then it is not the case that} & \\ \quad \underline{confl}\ (\sigma_1\ell, \sigma_2,\ T) & \\ \text{or } \underline{byp}\ (\sigma_1\ ,\ \ell,\ T) & \\ T \text{ otherwise} & \end{cases}$$

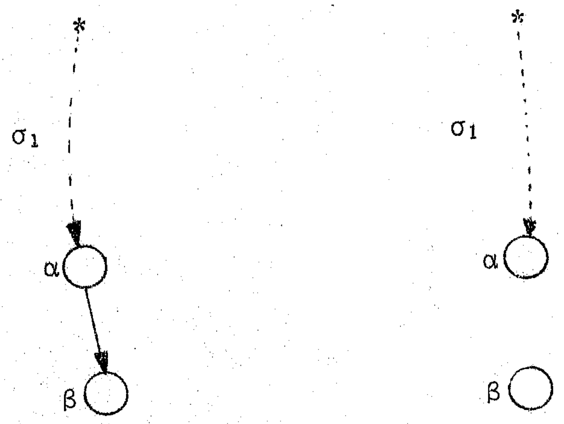
Intuitively <u>del</u> is applicable in the following cases, where $\alpha$ is the node reachable from * through $\sigma_1$, and $\beta$ the node reachable from * through $\sigma_1 \ell$(from $\alpha$ through $\ell$):

1. if there are no edges running from $\beta$;

2. if $\beta$ is also reachable from * through some $\sigma_2$ and $\sigma_2$ does not go through $\alpha$;

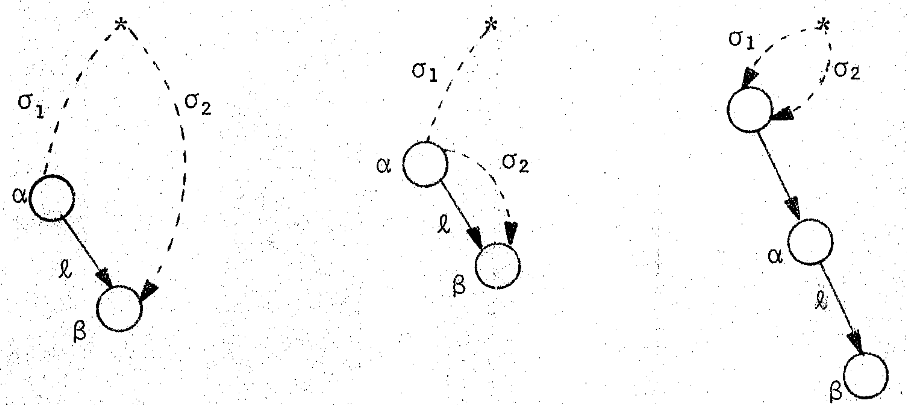
3. if between $\alpha$ and $\beta$ there is some path other than $\ell$.

The operation can be represented as follows:



In case 1, $\beta$ may become an element of $D_f$, which happens if there is no alternative path to it, and thus <u>del</u> is the inverse of <u>$ins_2$</u> , otherwise <u>del</u> is the inverse of <u>$ins_1$</u>.

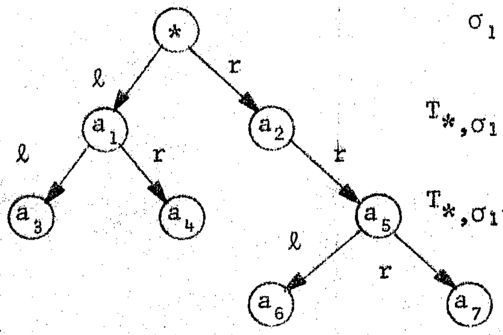
If there are edges running from $\beta$, the operation would be dangerous, because it might unwittingly cause other nodes to become unreachable from *, which in turn could violate the requirement that when a node is unreachable from * then there is no edge running from or into the node (the node becomes an element of $D_f$). However this unwanted situation will not arise if there is some alternative way to reach $\beta$ from * not involving the edge $\ell$ being deleted; cases 2 and 3 where this happens are shown below, the rightmost picture showing a violation of case 2:



<u>Example 2.3.</u>

We will delete the edge r from $a_5$ to $a_7$ in the following structure:

$T = \{<*, a_1, \ell>, <*, a_2, r> , <a_1,a_3,\ell> , <a_1,a_4, r> ,<a_1,a_5,r> ,$
$<a_5, a_6, \ell> ,<a_5, a_7, r>\}$

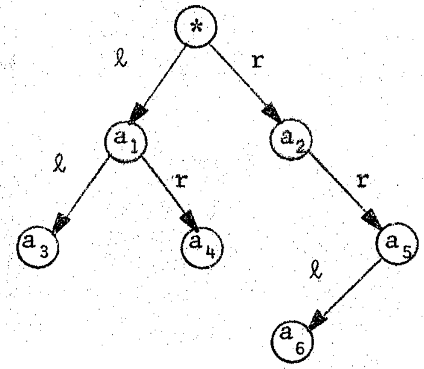


$\sigma_1 = rr$

$T_{*,\sigma_1} = \{<*, a_5>\}$

$T_{*,\sigma_1 r} = \{<*, a_7>\}$

$\underline{del}(\sigma_1,r, T) = T - (\underline{pr}_2(T_{*,\sigma_1}) \times \underline{pr}_2(T_{*,\sigma_1 r}) \times \{r\})$
$= T - \{<a_5,a_7, r>\}$



## 3. Examples

We will now present some examples, showing the application of the model in the description of data structures. To do this, we must restrict the relational data space so that it obeys certain specific characteristics; these are given using a first order language (at metalinguistic level), to describe them in a precise form. The operations on the data structures are defined using the three primitive operations del, $\underline{\text{ins}}_1$ and $\underline{\text{ins}}_2$. The application of these operations is illustrated by some examples.

### 3.1. Chains

These are relational data spaces with the additional characterizations below:

Axioms

(i) $\forall \underline{x}\ \forall \underline{y}\ (\underline{x}\varepsilon T \wedge \underline{y}\varepsilon T \rightarrow \underline{pr}_3\ (\underline{x}) = \underline{pr}_3\ (\underline{y}))$

Intuitively: there is only one edge label, which, by the way, implies that a chain cannot "branch"; if the single edge label is specified, say $\ell$, we have:

(i') $\forall \underline{x}\ (\underline{x}\ \varepsilon\ T \rightarrow \underline{pr}_3\ (\underline{x}) = \ell)$

(ii) $\exists \underline{y}\ (\underline{y}\ \varepsilon\ T \rightarrow \forall \underline{x}\ (\underline{x}\ \varepsilon\ T \rightarrow \underline{pr}_1\ (\underline{x}) \neq \underline{pr}_2\ (\underline{y}))$

Intituitively: it does not form a cycle.

Operations

(i) insertion at the end of a chain; the primitive operation $\underline{\text{ins}}_2$.

(ii) deletion at the end of a chain; the primitive operation del.

(iii) internal insertion of node x:

$$\underline{\text{ins}}_{ch}\ \begin{matrix}(\sigma,x,T) = \underline{\text{del}}\ (\lambda,\ell_0,\underline{\text{ins}}_1\ (\sigma\ell,\ell_0,\ell,\underline{\text{ins}}_2 \\ (\sigma,x,\ell,\ \ \underline{\text{del}}\ (\sigma,\ell,\ \underline{\text{ins}}_1\ (\lambda,\sigma\ell,\ell_0,T)))))\end{matrix}$$

Where $\sigma$ is the path to the node that will be before the node to be inserted, and $\ell_0$ is an auxiliary label.
Intuitively: an "auxiliary" edge $\ell_0$ is created first, linking $*$ to the node (call it $\beta$) following the one reached by $\sigma$(call it $\alpha$): now the $\ell$ edge from $\alpha$ to $\beta$ can be safely deleted because $\beta$ (and the nodes following it) are reachable though $\ell_0$; then $\alpha$ is connected to x (from $D_f$ the node to be inserted) and this node is in turn linked to $\beta$; finally the $\ell_0$ edge is discarded.

(iv) internal deletion

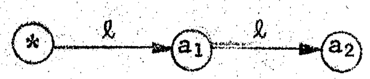
$$\underline{\text{del}}_{ch}\ (\sigma,T) = \underline{\text{del}}\ (\lambda,\ell_0,\underline{\text{ins}}_1\ (\sigma,\ell_0,\ell,\ \underline{\text{del}}\ (\sigma,\ell,\underline{\text{del}}\ (\sigma\ell,\ell,\text{ins}_1\ (\lambda,\sigma\ell\ell,\ell_0,T)))))$$

where $\sigma$ is the patch to the node previous to the node to be deleted, $\ell_0$ is an auxiliary label.
Intuitively: a strategy very similar to the one above is used.

Example 3.1.1.

Internal insertion of node b at right of node $a_1$ in the chain T given by

$T = \{<*,\ a_1\ ,\ \ell>\ ,\ <a_1,a_2,\ell>\}$



$$\underline{\text{ins}}_{ch}\ (\ell,b,T) = \underline{\text{del}}\ (\lambda,\ell_0,\underline{\text{ins}}_1\ (\ell\ell,\ell_0,\ell,\underline{\text{ins}}_2\ (\ell,b,\ell,\ \underline{\text{del}}\ (\ell,\ell\ ,\underline{\text{ins}}_1\ (\lambda,\ell\ell,\ell_0,T)))))$$

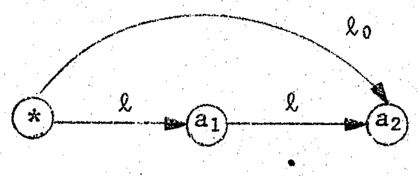
1st step: $\underline{\text{ins}}_1\ (\lambda,\ell\ell,\ell_0,T)$

$T_{*,\lambda} = \{<*,*>\}$

$T_{*,\ell\ell} = \{<*,a_2>\}$

then $\underline{\text{ins}}_1\ (\lambda,\ell\ell,\ell_0,T) = T \cup \{\ <*,a_2,\ell_0>\} = T'$

this procedure the chain



2nd step: <u>del</u> $(\ell,\ell,T')$

$= T' - \{<a_1, a_2, \ell>\} = T''$

i.e.

$$\begin{tikzcd} * \arrow[r, "\ell"] \arrow[rr, bend left, "\ell_0"] & a_1 & a_2 \end{tikzcd}$$

3rd step: <u>ins</u>$_2$ $(\ell, b, \ell, T'')$

$= T'' \cup \{<a_1, b, \ell>\} = T'''$

$$\begin{tikzcd} * \arrow[r, "\ell"] \arrow[rrr, bend left, "\ell_0"] & a_1 \arrow[r, "\ell"] & b & a_2 \end{tikzcd}$$

4th step: <u>ins</u>$_1$ $(\ell\ell, \ell_0, \ell, T''')$

$= T''' \cup \{<b, a_2, \ell>\} = T''''$

$$\begin{tikzcd} * \arrow[r, "\ell"] \arrow[rrr, bend left, "\ell_0"] & a_1 \arrow[r, "\ell"] & b \arrow[r, "\ell"] & a_2 \end{tikzcd}$$

5th step: <u>del</u> $(\lambda, \ell_0, T'''')$

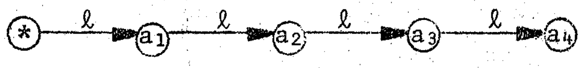
$= T'''' - \{<*, a_2, \ell_0>\} = T'''''$

$$\begin{tikzcd} * \arrow[r, "\ell"] & a_1 \arrow[r, "\ell"] & b \arrow[r, "\ell"] & a_2 \end{tikzcd}$$

Example 3.1.2.

We will delete the node $a_2$ in the chain T given by

$T = \{<*,a_1,\ell>\ ,\ <a_1,a_2,\ell>\ ,\ <a_2,a_3,\ell>,<a_3,a_4,\ell>\ \}$

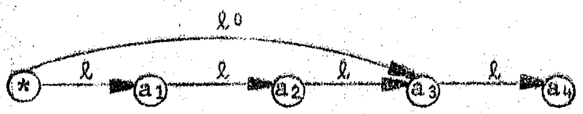


$\underline{\text{del}}_{ch}\ (\ell,T) = \underline{\text{del}}\ (\lambda,\ell_0,\ \underline{\text{ins}}_1\ (\ell,\ell_0,\ell,\ (\underline{\text{del}}\ (\ell,\ell,\ \underline{\text{del}}\ (\ell\ell,\ell,\ \underline{\text{ins}}_1$

$(\lambda,\ell\ell\ell,\ell_0,T)))))$

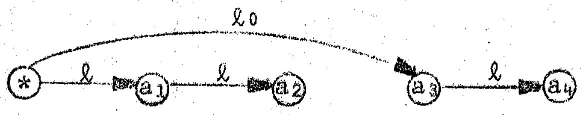
1st step: $\underline{\text{ins}}_1\ (\lambda,\ell\ell\ell,\ell_0,T)$

$= T \cup \{<*,a_3,\ell_0>\} = T'$



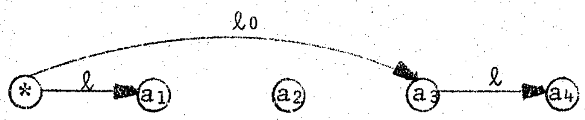
2nd step: $\underline{\text{del}}\ (\ell\ell,\ell,T')$

$= T' - \{<a_2,a_3,\ell>\} = T''$



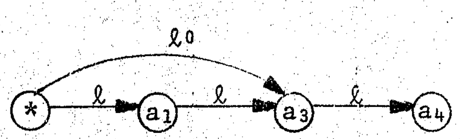
3rd step: $\underline{\text{del}}\ (\ell,\ \ell,T'')$

$= T'' - \{<a_1,a_2,\ell>\} = T'''$



4th step: $\underline{ins}_1\ (\ell, \ell_0, \ell, T''')$

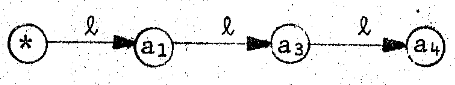
$= T''' \cup \{<a_1, a_3, \ell>\} = T''''$



5th step: $\underline{del}\ (\lambda, \ell_0, T'''')$

$= T'''' - \{<*, a_3, \ell_0>\} = T'''''$

$T''''' = \{<*, a_1, \ell>\ ,\ <a_1, a_3, \ell>\ ,\ <a_3, a_4, \ell>\}$



Note: In fact, operations (iii) and (iv) are sufficient; the reader should convince himself that $\underline{ins}_{ch}$ and $\underline{del}_{ch}$ also work for insertion and deletion, respectively, at the end of a chain. The main point is that the prerequisite conditions for applying the primitive operators in order to manipulate the auxiliary edge will fail, and thus the corresponding steps will not be executed.

## 3.2. Trees

These are relational data spaces with the additional characterizations below:

Axiom

(i) $\forall \underline{x}\ \forall \underline{y}\ ((\underline{x} \in T \wedge \underline{y} \in T) \rightarrow \underline{pr}_2\ (\underline{x}) \neq \underline{pr}_2\ (\underline{y}))$

Operations

(i) Insertion of a new node x
$\underline{ins}(\sigma, x, \ell, T) = \underline{ins}_2(\sigma, x, \ell, T)$

(ii) Deletion of an edge $\ell$, from a node accessed by $\sigma$

$$\underline{\text{delt}}_{tr}\ (\sigma,\{\ell\},T) = \begin{cases} T \text{ if } \underline{\text{end}}\ (\sigma,T) \\ \text{otherwise} \\ \underline{\text{del}}\ (\sigma,\ell,\ \underline{\text{del}}_{tr}\ (\sigma\ell,L_a,T))\quad (*) \end{cases}$$

Letting <u>sup</u> select the largest element of a set, we can write:

$$\underline{\text{del}}_{tr}\ (\sigma\ell,L,T) = \underline{\text{del}}_{tr}\ (\sigma\ell,\{\underline{\text{sup}}\ (L)\}\ ,\ \underline{\text{del}}_{tr}\ (\sigma\ell,L - \{\text{sup}\ (L)\},T))$$

where L is a set of cardinality greater than one, $L \subseteq L_a$.
In the case of binary trees, thus having only two labels, we would have

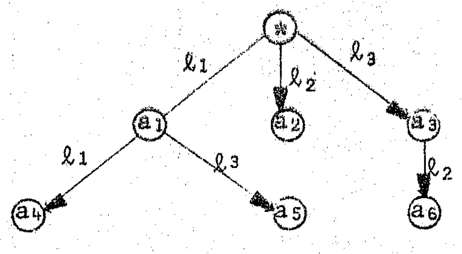
$$\underline{\text{del}}_{tr}\ (\sigma\ell,L,T) = \underline{\text{del}}_{tr}\ (\sigma\ell,\ell_1,\ \underline{\text{del}}_{tr}\ (\sigma\ell,\ell_2,T))$$

Note that in (*) we have as second argument $L_a$ (set of active labels), which is a linearly ordered <u>finite</u> set (which means that we simply have recursion on a finite set).

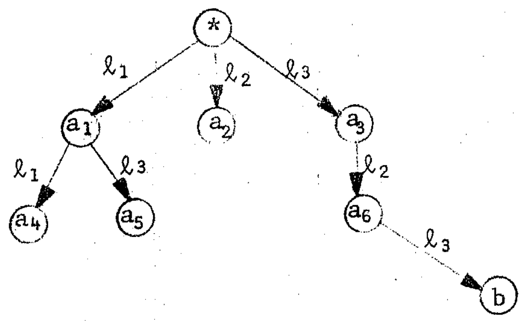
<u>Example 3.2.1.</u>

We will insert the node b at the $\ell_3$ edge of node $a_6$ of the tree:

$$T = \{<*,a_1,\ell_1>\ ,\ <*,a_2,\ell_2>\ ,\ <*,a_3,\ell_3>\ ,\ <a_1,a_4,\ell_1>\ ,\ <a_1,a_5,\ell_3>\ ,\ <a_3,a_6,\ell_2>\}$$



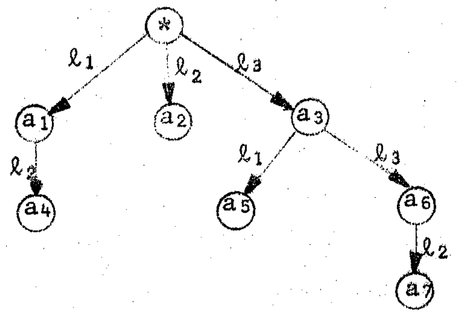
$$\underline{\text{ins}}\ (\ell_3\ell_2,b,\ell_3,T) = \underline{\text{ins}}_2\ (\ell_3\ell_2,b,\ell_3,T)$$
$$= T \cup \{<a_6,b,\ell_3>\}$$



Example 3.2.2.

We will delete the label $\ell_3$, from the node $a_3$ to node $a_6$ in the tree:

$T = \{<*,a_1,\ell_1> , <*,a_2,\ell_2> , <*,a_3,\ell_3> , <a_1,a_4,\ell_2> , <a_3,a_5,\ell_1> ,$
$<a_3,a_6,\ell_3> , <a_6,a_7,\ell_2>\}$



$La = \{\ell_1,\ell_2,\ell_3\}$ witch the order $\ell_1>\ell_2>\ell_3$

$\underline{del}_{tr}\ (\ell_3,\{\ell_3\},T) = \underline{del}\ (\ell_3,\ell_3,\underline{del}_{tr}(\ell_3\ell_3,La,T))$

$\underline{del}_{tr}\ (\ell_3\ell_3,La,T) = \underline{del}_{tr}\ (\ell_3\ell_3, \{\underline{sup}\ (La)\}, \underline{del}_{tr}\ (\ell_3\ell_3,$
$La - \{\underline{sup}\ (La)\},T))$

$\underline{del}_{tr}\ (\ell_3\ell_3,La - \{\underline{sup}\ (La)\},T) = \underline{del}_{tr}\ (\ell_3\ell_3,\{\ell_2,\ell_3\},T)$

$\underline{del}_{tr}\ (\ell_3\ell_3,\{\ell_2,\ell_3\},T) = \underline{del}_{tr}\ (\ell_3\ell_3,\{\ell_2\}, \underline{del}_{tr}\ (\ell_3\ell_3,\{\ell_3\},T))$

$\underline{del}_{tr}\ (\ell_3\ell_3,\{\ell_3\},T) = \underline{del}\ (\ell_3\ell_3,\ell_3, \underline{del}_{tr}\ (\ell_3\ell_3\ell_3,La,T))$

$= T$

$\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_2,\ell_3\},T) = \underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_2\},T)$

developing $\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_2\},T)$ we arrive at

$\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_2,\ell_3\},T) = \underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\ell_2,T)$

$\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_2,\ell_3\},T) = T - (<a_6,a_{11},\ell_2>) = T'$

$\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,La - \{\underline{\text{sup}}\ (La)\},T) = T'$

$\underline{\text{del}}_{tr}$ $(\ell_3\ell_3,La,T) = \underline{\text{del}}_{tr}$ $(\ell_3\ell_3,\{\ell_1\},T')$

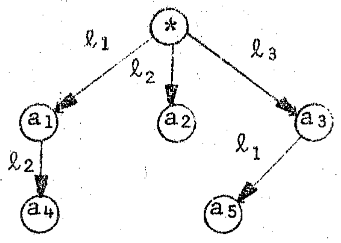
$= T''$

$\underline{\text{del}}_{tr}$ $(\ell_3,\{\ell_3\},T) = \underline{\text{del}}$ $(\ell_3,\ell_3,T'')$

$= T'' - \{<a_3,a_6,\ell_3>\}$

$= T'''$

$T''' = \{<*,a_1,\ell_1> , <*,a_2,\ell_2> , <*,a_2,\ell_2> , <a_1,a_4,\ell_2> , <a_3,a_5,\ell_1>\}$



## 3.2.1. Binary Trees

In addition to the axiom for general trees, we have

$$\forall \underline{x}\ \forall \underline{y}\ \forall \underline{z}\ (\underline{x}\in T \wedge \underline{y}\in T \wedge z\in T \rightarrow \underline{pr}_3\ (\underline{x}) = \underline{pr}_3\ (\underline{y}) \vee \underline{pr}_3\ (\underline{x}) = \underline{pr}_3\ (z) \vee \vee\ \underline{pr}_3\ (\underline{y}) = \underline{pr}_3\ (\underline{z}))$$

## 4. CONCLUSIONS

We define a general framework based on the relational properties of the elements of our universe of data, as a first step in the modelling of different kinds of data structures. In the description of each data structure we presented a set of axioms given in a first order language. These axioms characterize the static aspects of the structure, i.e., the structural attributes of the elements of that particular domain. In each description we also defined certain allowable operations over this domain. These operations describe the dynamic aspects of each type of data structure.

Using the terminology of Liskov and Zilles, our model uses a "fixed discipline" for the specification of data abstractions. It has a wide range of applicability in the description of different types of data structures. The use of a relational model facilitates the design of a language to prove properties about the model, for instance we can consider a first order language whose variables would be the relations, and the functionals and predicate symbols would be interpreted as being the operations and relations over that domain of relations. Another important point in favor of the model is its conceptual simplicity; any data type is defined in terms of a single collection of triples.

As for all specification technique, the main difficulty in describing a data abstraction is the requirement of precision in the characterization, i.e., the choice of axioms. In addition, the definition of the operations over a particular type of data abstraction must be such that the results of these operations obey the specifications of characterization, (i.e. the axioms). In other words, the operations must be safe, closed with respect to the axioms.

## Acknowledgement

The authors are grateful to Prof. F. Tompa, for his helpful discussions and valuable sugestions.

## BIBLIOGRAPHY

1. BURSTALL, R.M. "Some Techniques for proving correctness of programs which alter data structures". Machine Intelligence, 7, 1972.

2. EARLY, J. "Relational level data structures for programming languages". Acta Informatica, 2, 14: 239-309, 1973

3. FURTADO, A.L. "Characterizing sets of data structures by the connectivity relation". International Journal of Computer and Information Science. Vol. 5, Nº 2, 1976.

4. KOWALTOWSKI, T. "Correctness of program manipulating data structures". Berkeley, Univ. of California, Eletronics Research Laboratory, 1973. Mem. 404

5. LISKOV, B., ZILLES, S. "An Introduction to Formal specifications of Data Abstractions". Current trends in Programming methodology. Vol. I, R. YEH ed.

6. MAJSTER, M.E. "Extended Directed Graphs, a Formalism for Structured Data and Data Structures". Acta Informatica, Vol. 8, Fasc. 1, 1977

7. MAJSTER, M.E. "N$^{TH}$ - Level Data Structures". München Technishe Universitat. München, TUM - IFO - 7707, 1977.