# Semantic tree operations

D. Gueorguiev 1/6/2021

Notation:

– an -ary tree

- the set of all nodes of

– path from root of to node

– tree tuple of

– tree factor, an integer

- tuple of primitive factors

– primitive factor, an integer

– the set of all primitive factor tuples of

– the set of all arcs of

– the set of all weights of

## Summary and definitions

We are considering -ary tree (abbreviated with from now on), which is a tree in which each node has at most children. Let us introduce the *tree tuple* where represents a *tree node* which is a semantic particle or a *subtree* of nodes (i.e. semantic particles). We will denote by the set of all nodes which belong to the tree .

Here is a *tree factor* which encodes uniquely the position of the node or the root of the subtree in the parent tree . More precisely, the factor encodes uniquely the path from the root of the tree to the node or the root of subtree associated with . Each tree factor representing a node other than the root of can be decomposed into a *tuple* of *primitive* *factors* where is smaller or equal to the height of the tree. Here with dot-accented we denote a primitive factor i.e. a factor which cannot be represented by any other combination of primitive factors.

We will denote with the set of all tuples of primitive factors associated with tree . The position in of each of those primitive factors and their value encodes an *arc* from the set of arcs forming the path .

**Definition**: *Arc* of semantic tree

Let us define a tree with nodes as:

(1)

We denote with the tuple of the primitive factors for :

. (2)

Every arc of is represented by a tuple such that . However not every tuple represents an arc - the tuple associated with the root of does not correspond to an arc of . We will denote with the set of all arcs of . Obviously, .

If we substitute (2) in (1) we come up with the following simplified notation for represented in terms of its primitive factors which will be used throughout this discussion:

(3)

*Example*

/ | \

/ | \

/ / \ \

The root node is given with the tuple . Its children are given with , , and . Their children are accordingly , , , and . In a later paragraph we will introduce rules which will allow us to show that the following expansions take place:

Thus, the tree can be written as:

T = .

Notice that for the root of the tree we always have:

(4)

**Definition**: *Matching arcs of trees*

Let us consider two trees represented as:

and

If for some and we have then the arcs which correspond to and are said to be *matching*.

**Definition**: *Weighted semantic tree*

Let there exists a function which maps each tree factor to a real number which is the weight corresponding the arc associated with the specified tuple factor. So, the number associated with the arc corresponding to is the arc weight. With we will denote the set of all where .

**Definition**: *Weight of a tree path*

Let us evaluate the weight of the path from root to node . Let be the factor associated with and . Then the weight of the path is given with:

(5)

**Definition**: *Semantic significance vector of a semantic tree*

In various settings we will use a weight vector instead of scalar weight with semantic trees and semantic paths. Thus, the vector-valued function will map each tree factor to an element of some vector space . We will refer to each element of mapped to a tree arc or a path as *semantic* *significance vector*.

//TODO

## Semantic subtree expansion and node comparison

The following operations are defined for tree factors:

### Multiplication operation for semantic tuple factors

One possible implementation for the primitive factors is to define them as the digits greater than 0 of -nary number system such that . We define an operation `` denoting digit concatenation . Obviously,

for any pair

Note that the latter implies that

for any tuple where

Encoding a complete -ary tree of height with the algebraic notation above:

. Further we will assume that .

In general, we have:

where

Obviously, we have at most distinct terms which represent nodes i.e. semantic values.

### Tuple factor comparison operator

The expression for the tree also can be written as:

where and are the *node factors* given with . The node values are the values ordered in increasing order of . This order corresponds to *level order traversal* of the -ary tree. Note that with appropriately defined comparison operation `<` we can model different ways of traversing the -ary tree. For instance, if we define `<` as the comparison for the values of we will have ordering which corresponds to the *preorder traversal* of the tree.

*Example*

*Peter is Dimitar’s son.*

*Dimitar’s son has a friend in the neighborhood and his friend’s name is James.*

* *James is Peter’s friend*

*Peter is the son of Dimitar.*

*The son of Dimitar has a friend in the neighborhood and the name of his friend is James.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/ \ / \

\_\_\_\_\_\_\_\_

/ | \ / \ \ / \

/ | \ | / \ / \

/ \

/ \

/ \

Expressing with the algebraic notation discussed earlier:

which is expanded to:

Expressing with the algebraic notation yields:

+

which is expanded to:

## Semantic Tree Difference

**Definition**: *Semantic Tree Difference* – a metric (real valued function) which maps two semantic trees of the same kind into a real value.

Let us have two trees represented as:

and (6)

Here denote the nodes of the two trees (not subtrees) which are semantic particles with signatures . The sequences and denote the sequences of tuple factors which encode the position of each node in each of the two trees.

Let us assume that the two trees are weighted so that for each tree there is weight function which maps each tuple factor to a real number which is the weight corresponding the arc associated with the specified tuple factor. Let us denote by and the two weight functions corresponding to and . Generally, . If then and have the same weights on their matching arcs.

We have the same set of semantic particles but they are arranged differently in two trees. We would like to define metric how different are the two trees.

We will consider the following special cases – the trees which we would like to obtain a

Metric for calculating Semantic Property Tree Difference will be two kinds:

* Semantic Property Trees
* Semantic Particle Trees

First, we need to elaborate on a metric defining a semantic difference between two semantic property trees

### Semantic Difference between Semantic Property Trees

**Definition**: *Semantic difference between semantic properties*

Let us consider two semantic properties and given with their semantic signature matrices

and where the column vectors and are the semantic aspects of and accordingly. Without loss of generality we will assume that . Then the semantic difference between the properties and is given with

(7)

where and denote the semantic mass and energy of the aspect of while and denote the corresponding quantities for the of . The summation in the first summation term occurs over all pairs such that for a given the index is selected as that aspect index from which yields the smallest value of . The aspect indices of are ordered by decreasing values so we start with the aspect having the highest ratio of and attempt to match it with similar one from .

The second summation term in (7) represents the unmatched aspects from which obviously are .

**Definition**: *Semantic difference between semantic property trees*

Let us consider the property trees represented as:

and

Both trees share the same property set ,. As before we will denote by and the two weight functions corresponding to and .

We would like to evaluate how much different semantically are the two trees

//TODO