# Computerlinguistik

WS 2018/2019

Greg Kobele

October 26, 2018

## 1 Trees

A tree is a formal object inspired by a, well, tree. (I sort of want to write 'hah hah, just kidding', but there's too much truth in that statement to completely dismiss!) All kidding aside, a tree is a formal object designed to represent *hierarchical structures.* Hierarchical structures naturally emerge following the arrow of time; geneology is an example, with ancestors being temporally (and causally) prior to their decendents. The crucial aspect of hierarchical structures formalized via trees is that there may be many entities *at the same level* in the hierarchy – they are neither superior nor inferior to each other. Siblings are an example of this in the geneological example, as are members of the armed forces holding the same rank (say, lieutenant) in a context where the hierarchy in question has to do with social standing rather than time.

There is a natural formal sense in which trees are a generalization of strings (and conversely, in which strings are a special case of trees): a string is a tree where each superior has at most one immediate subordinate. We will again provide two formalizations of the concept of trees, one generalizing the array perspective, the other the list perspective on strings.

### 1.1 Trees as arrays

The array perspective on strings views them as sets of positions, where these positions are *totally ordered.*[1] We represented the positions as consecutive

[1] A *partial ordering* of a set is a relation (which we will write as $\leq$) defined over elements of that set which is

**reflexive** for all elements $a$, $a \leq a$

**antisymmetric** for all distinct elements $a$ and $b$, if $a \leq b$ then **not** $b \leq a$

natural numbers, and *implicitly* used the less-than-or-equal-to relation as the total ordering. To pave the way for the generalization to trees, let us represent instead positions as *strings* of numbers. The first position will be the empty string $\epsilon$, the second position will be the one-element string 0, the third the two element string 00, etc, with the $n^{th}$ position being represented as the $n-1$ element string $0^{n-1}$ (0 repeated $n-1$ times).[2] Here, the total ordering between positions is given by the 'is-a-prefix-of' relation (with $a$ coming before $b$ iff $a$ is a prefix of $b$). Figure 1 presents the two different notations for positions, in the context of the string *abba*.

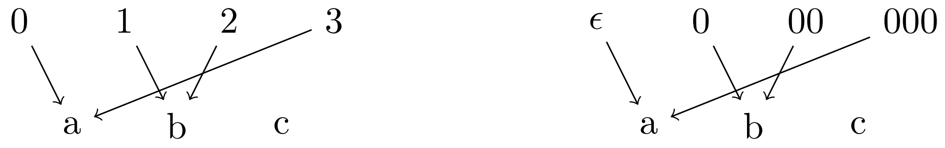

Figure 1: Two different notations for positions

Although numbers and sequences of the number 0 are interchangeable, moving to this alternative representation of positions makes a generalization salient which was not visible in the realm of numbers, one which allows us to represent the possibility of having *multiple* next positions. If a position is represented by the sequence $\sigma$, then $\sigma 0$ is the *first* successor of $\sigma$. We can represent the *second* successor of $\sigma$ as $\sigma 1$, the *third* as $\sigma 2$, and so on; in general, the $k+1^{th}$ successor of $\sigma$ is $\sigma k$. Figure 2 presents a side by side comparison of a tree (on the right) and our formal representation of it in terms of a function from positions.

Not just any set of strings of numbers represents the positions of a tree. A set of strings of numbers satisfying the following two conditions does, and is called a *tree domain*.

**Definition 1.** A **tree domain** is a set $D$ of sequences of natural numbers satisfying the conditions below.

**transitive** for all triplets of elements $a$, $b$, and $c$, if $a \leq b$ and $b \leq c$ then $a \leq c$

A *total ordering* is a partial ordering which satisfies the additional condition of being connected.

**connectedness** for all distinct elements $a$ and $b$, either $a \leq b$ or $b \leq a$

The natural numbers provide a familiar example of a totally ordered set (where the ordering is the usual 'less than or equal to').

[2]Recall that the $n^{th}$ position was represented by the number $n-1$, and so the string $0^k$ and the number $k$ represent the same position.

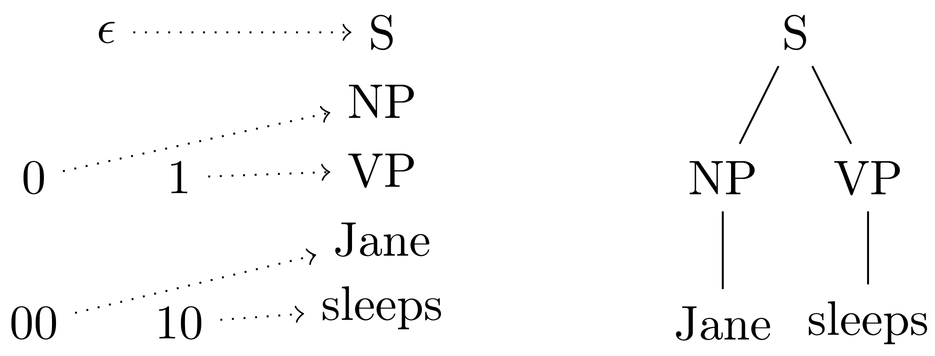

Figure 2: A position based representation of a tree

**prefix closure** for any sequences $\sigma$ and $\tau$, if $\sigma\tau \in D$, then $\sigma \in D$ as well

**sisterhood closure** for any sequence $\sigma$ and number $i$, if $\sigma i \in D$, then for every number $j \leq i$, it is also the case that $\sigma j \in D$

Some trees have an upper bound on the maximal number of daughters had by any node. (Every finite tree has such an upper bound.) Given a tree domain $D$, it's *width* is the least number $b + 1$ such that $b$ is used in some string in $D$, or 0 if $D = \{\epsilon\}$. Strings have a width of 1.

## 1.2 Trees as lists

According to the inductive definition of strings presented earlier, a string was either empty, or it consisted of a symbol, followed by the rest of the string. Continuing to follow our imperative of viewing strings as the special case of trees where each node has at most one successor, we see that we must generalize our definition to allow more than one possible 'rest of the string' to follow a symbol. We can do this by allowing not a single object, but rather a *sequence* thereof, to follow a symbol. This sets the stage for our inductive definition of trees.

**Definition 2.** Given an alphabet $\Sigma$, a tree over $\Sigma$ consists of a symbol $a \in \Sigma$ followed by a sequence $ts$ of trees over $\Sigma$:

$$a \lhd ts$$

The width of a tree (defined inductively) can be given as the maximal length of any sequence of siblings in a tree. We can define this function recursively as follows.

**Definition 3.**

$$\texttt{width}(a \lhd ts) := \texttt{max}(\{|ts|\} \cup \{\texttt{width}(ts(i)) : i < |ts|\})$$

The workings of this function can be understood by translating the definition; the width of a tree $a \lhd ts$ is the maximum number from amongst:

- the number of daughters of $a$ (that is: $|ts|$)
- the widths of the trees making up $ts$

The size of a tree, which we shall write $|t|$ on analogy with the length of a list, is given by the number of symbols it contains. The size of a tree $a \lhd ts$ is one greater than the sum of the sizes of the trees occurring in $ts$.

**Definition 4.**

$$|a \lhd ts| := 1 + \sum \{|ts(i)| : i < |ts|\}$$

Another measure of the size of a tree is its *depth*; the length of the longest path from root to a leaf.

**Definition 5.**

$$\texttt{depth}(a \lhd ts) := 1 + \texttt{max}(\{\texttt{depth}(ts(i)) : i < |ts|\})$$

A tree with an infinite number of nodes (i.e., $|t| = \infty$) is called *infinite*, otherwise it is *finite*. A tree is finite iff it has both a finite width and a finite depth.