2.2. Parsing trees

In this section, we make precise the ideas of Section 2.1.
We will define the formulas of $LP(\sigma)$ in terms of their parsing trees.

Scanné avec CamScanner

Definition 2.2.1

A (planar) tree is an ordered pair (N, D) where

(a) N is a finite non-empty set whose elements are called nodes;

(b) D is a function that takes each node μ in N to a sequence (possibly empty) of distinct nodes

$$D(\mu) = (\nu_1, \dots, \nu_n), \tag{2.17}$$

the nodes $\nu_1, \dots, \nu_n$ are called the daughters of μ, and μ is called the mother of $\nu_1, \dots, \nu_n$;

(c) every node except one has exactly one mother; the exception is a node called a root, in symbols $\sqrt{}$, which has no mother;

(d) there are no cycles, that is sequences

$$\nu_1, \nu_2, \dots, \nu_k \quad (k > 1) \tag{2.18}$$

where $\nu_k = \nu_1$ and each ν_i, $1 \leq i < k$, has mother ν_{i+1}.

Scanné avec CamScanner

To draw a tree (N, D), we first draw a node for its root $\sqrt{}$. If $D(\sqrt{}) = (\mu_1, \dots, \mu_n)$, then below the root we draw nodes $\mu_1, \dots, \mu_n$ from left to right, joined to $\sqrt{}$ by lines. Then we put the daughters of μ_1 below μ_1, the daughters of μ_2 below μ_2, etc., and we carry on downwards until all the nodes are included. This will happen because N is finite; and if we start from any node and go to its mother, the mother of its mother and so on, then by (d) we must eventually reach a node with no mother, which by (c) must be $\sqrt{}$.

The route from a node to the root is unique, since a node has at most one mother.

Scanné avec CamScanner

Definition 1.2.2

(a) In a tree, an *edge* is an ordered pair of nodes (μ, ν), where μ is the mother of ν. (So the lines in a tree diagram represent the edges.) We describe a node as a *leaf* of a tree if it has no daughters.

(b) The number of daughters of a node is called its *arity*. (So the leaves are the nodes of arity 0.)

(c) We define a *height* for each node of a tree as follows. If μ is a node with daughters $\nu_1, \ldots, \nu_n$, then the height of μ is

$$\max\{\text{height}(\nu_1), \ldots, \text{height}(\nu_n)\} + 1. \tag{2.19}$$

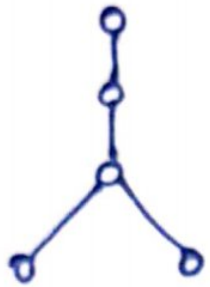
The *height of a tree* is the height of its root.

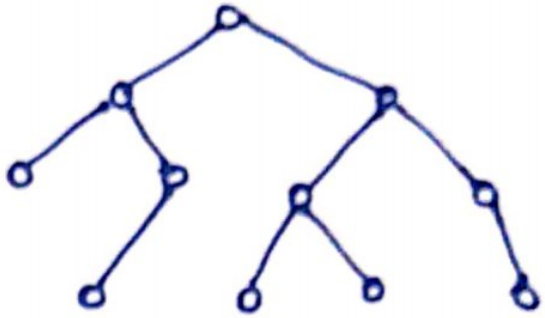
(d) A *path* from node ν to node μ is a set of nodes $\{\nu_0, \ldots, \nu_k\}$ where ν_0 is ν, ν_k is μ and for each $i<k$, ν_i is the mother of ν_{i+1}. A path from the root to a leaf μ is called a *branch* (to μ).

Scanné avec CamScanner

Here are three examples of trees :

(2.20)





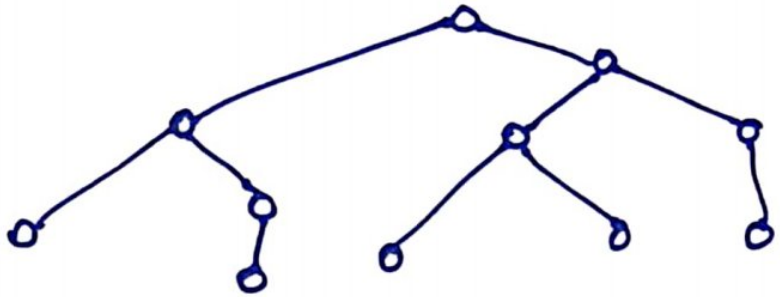
- The left-hand tree has one leaf and no non-leaves. Its height is 0.
- The centre tree has two leaves, two nodes of arity 1 and one node of arity 2 ; the root has height 3, so the height of the tree is 3.
- The right-hand tree has five leaves, two nodes of arity 1 and four nodes of arity 2. The height of the tree is 3.

Scanné avec CamScanner

Identifying two trees

The tree

(2.21)



is strictly not the same tree as the third tree in (2.20), because the two trees on different pages; also the angles are different in the two diagrams. But there is a unique correspondence between the nodes of one tree and the nodes of the other, and between the edges of one tree and those of the others. So, we will 'identify' the two trees, that is, count them as the same tree. (See exercise 2.2.6.)

Scanné avec CamScanner

Definition 2.2.3

A labelling of a tree is a function f defined on the set of nodes. We make it a right labelling by writing $f(v)$ to the right of each node v, and a left labelling by writing $f(v)$ to the left of v. A labelled tree is a tree together with a labelling; likewise we talk of left-labelled trees, and right-labelled trees.

Scanné avec CamScanner

Definition 2.2.4

A *parsing tree* for $LP(\sigma)$ is a right-labelled tree where

- every node has arity $\leqslant 2$;
- every leaf is labelled with either $\bot$ or a symbol from σ ;
- every node of arity 1 is labelled with $\neg$;
- every node of arity 2 is labelled with one of $\wedge, \vee, \rightarrow, \leftrightarrow$.

Scanné avec CamScanner

Now, we can easily check that all the parsing trees of Section 2.1 are parsing trees in the sense of Definition 2.2.4.
So we can define the formulas of $LP(\sigma)$ by saying how they are constructed from parsing trees. The method that we use, working up from leaves to the root as in Section 2.1, will have many applications. For example, we use it to set up alternative notations, and to define properties of formulas, and to assign meanings to formulas. To make it precise, we make the following definition.

Definition 2.2.5

A <u>compositional definition</u> δ is a set of rules that tell us how to put a left labelling on any parsing tree, in such a way that the left label on any node μ - write it $\delta(\mu)$ - is determined by just two things:

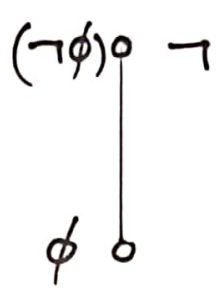
- the right-hand label on μ, and
- the sequence of values $(\delta(\nu_1), \dots, \delta(\nu_n))$, where μ has daughters $\nu_1, \dots, \nu_n$ from left to right.

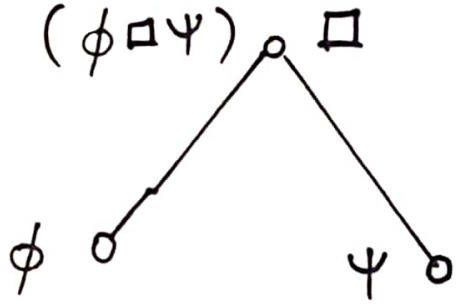
The rules must always determine $\delta(\mu)$ uniquely from this data, so that they define a unique left labelling for any parsing tree π; the label on the root of π is called the <u>root label</u>, in symbols $\delta(\pi)$.

Example 2.2.6

The rules that we used in Section 2.1 for recovering a formula from its parsing tree form a compositional definition. We can write it

$$\chi \circ \chi \qquad (2.22)$$





where χ is $\bot$ or a propositional symbol in σ, and $\square$ is a truth function symbol $\wedge$, $\vee$, $\rightarrow$ or $\leftrightarrow$.

This is three instructions. The first says: at a leaf, copy the right-hand label on the left. The second says: at a node with right-hand label $\neg$, write $(\neg\phi)$ where ϕ is the left label on the daughter. The third tells you what to do at a node with two daughters.

Definition 2.2.7

(a) If π is a parsing tree for $LP(\sigma)$, then the *formula associated to* π is $\delta(\pi)$, where $\delta(\pi)$ is the compositional definition (2.22). We say that π is a *parsing tree for* $\delta(\pi)$. The *formulas of* $LP(\sigma)$ are the formulas associated to parsing trees of $LP(\sigma)$. A *formula of* LP is a formula of $LP(\sigma)$ for some signature σ.

(b) The formula $\bot$ and propositional symbols are called *atomic formulas*. (These have parsing trees with just one node.) All other formulas are said to be *complex*.

(c) A formula has *complexity* k if it is associated to a parsing tree of height k. (So atomic formulas are those with complexity 0.)

For example, using the default signature, the following are atomic formulas :

$$\bot \qquad p_0 \qquad p_{2002} \qquad p_{999999999999}$$

On the other hand the following expressions

$$(\neg p_1) \qquad (p_0 \to (p_1 \to p_0)) \qquad ((p_1 \wedge (\neg p_0)) \leftrightarrow p_5)$$

are complex formulas. (draw their parsing trees)

Remark.

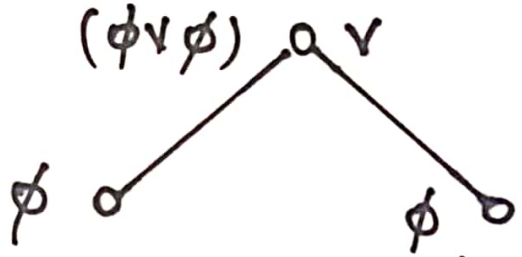
We will see in Section 2.3 that the notion of complexity is well defined.

Remark 2.2.8

Let π be a parsing tree and v a node of π. Then removing all the nodes of π which are not v and cannot be reached from v by going downwards along edges, we get a smaller parsing tree. The left label on v given by (2.22) is the label on the root of this smaller tree, so it is itself a formula. Thus all the left labels on nodes of a parsing tree given by (2.22) are formulas.

Imagine we are travelling up a parsing tree using (2.22) & attach formulas on the left sides of nodes. Nothing gets thrown away: if we write a formula ϕ against a node, the ϕ reappears in the label of its mother nodes In fact we can say what part of the label on the mother node is this ϕ, and we call this the trace of ϕ. Take, for example, a part of a parsing tree:

(2.23)



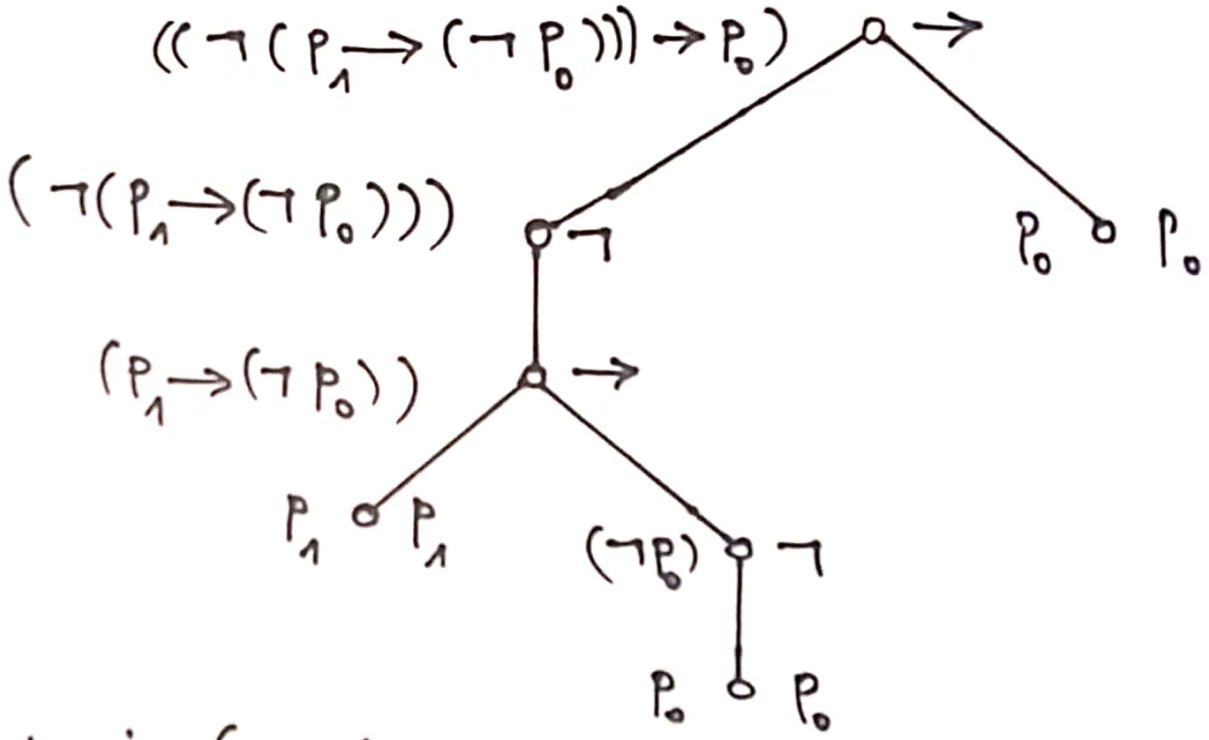
We labelled both of the two lower nodes ϕ. Then the upper node gets the label $(\phi \vee \phi)$. The left-hand ϕ is the trace of the ϕ on the bottom left node, and the right-hand ϕ is the trace of the ϕ on the bottom right node. We say in this case that there are two occurrences of ϕ in the formula $(\phi \vee \phi)$; each trace is a separate occurrence.

Definition 2.2.9

Let ϕ be the associated formula of a parsing tree P. Then the <u>subformulas</u> of ϕ are the traces in ϕ of the left labels of all the nodes in P.

For example, the formula $((\neg(P_1 \to (\neg P_0))) \to P_0)$ has the parsing tree

(2.24)

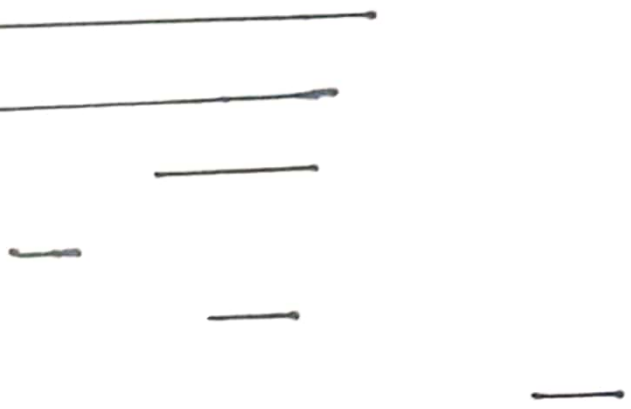


as we calculated in (2.14).

The tree has seven nodes in its parsing tree, and hence the formula has seven subformulas, illustrated as follows :

$$\underline{((\neg(P_1 \to (\neg P_0))) \to P_0)}$$

(2.25)



Note that the last two subformulas are the same formula, namely P_0, but occurring in different places.