Propositional formulas

Chapter 3, Section 3

Introduction

The main aim of this section is to prove that each formula of LP(σ) is associated to a unique parsing tree. When we analysed the formula $(p \to (\neg(\neg p)))$ in Section 3.1, we built a parsing tree for it. What has to be shown is that this method of analysis always works in a unique way, and it recovers the parsing tree that the formula is associated to.

By Definition 3.2.7 every formula ϕ of LP is the root label got by applying the compositional definition (3.22) to some parsing tree π. Looking at the clause of (3.22) used at the root, we see that ϕ is either atomic or a complex formula that can be written in at least one of the forms $(\neg\phi)$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ or $(\phi \leftrightarrow \psi)$, where ϕ and ψ are formulas (using Remark 3.2.8). In the complex cases the shown occurrence of $\neg$, $\wedge$, $\vee$, $\to$ or $\leftrightarrow$ is said to be a *head* of the formula. To show that each formula has a unique parsing tree, we need to prove that each complex formula has a unique head. In fact this is all we need prove, because the rest of the tree is constructed by finding the heads of the formulas on the daughter nodes, and so on all the way down. Our proof will give an algorithm, that is, a mechanical method of calculation, for finding the head.

Purely for this section, we call the symbols $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$ the *functors*. So a functor is a truth function symbol but not $\bot$.

Depth

Definition 3.3.1 An expression is a string $a_1 \cdots a_n$ of symbols and its length is n. A *segment* of this expression is a string

$$a_i \cdots a_j \text{ (with } 1 \leq i \leq j \leq n \text{)}$$

This segment is *initial* if *i* = 1; so the initial segments are

$$a_1 \quad a_1 a_2 \quad a_1 a_2 a_3 \quad \cdots \quad a_1 \cdots a_n$$

The *proper initial segments* of the expression are those of length $< n$. For each initial segment *s* we define the *depth* $d[s]$ to be the number of occurrences of '(' in *s* minus the number of occurrences of ')' in *s*. The *depth* of an occurrence a_j of a symbol in the string is defined as the depth of $a_1 \cdots a_j$.

Once again, remember that a propositional symbol is a single symbol. For example, we count p_{12047} as one symbol, not as a string of six symbols.

Example

Example 3.3.2 In the string $(p_0 \to (p_1 \to p_0))$ the depths of the initial segments are as follows :

Initial segment	Depth
$($	1
$(p_0$	1
$(p_0 \to$	1
$(p_0 \to ($	2
$(p_0 \to (p_1$	2
$(p_0 \to (p_1 \to$	2
$(p_0 \to (p_1 \to p_0$	2
$(p_0 \to (p_1 \to p_0)$	1
$(p_0 \to (p_1 \to p_0))$	0

Depth of a formula

Lemma 3.3.3

Let χ be any formula of LP. Then

(a) χ has depth 0, and every proper initial segment of χ has depth > 0;

(b) if χ is complex then exactly one occurrence of $\wedge$, $\vee$, $\to$, $\leftrightarrow$ or $\neg$ in χ has depth 1, and this occurrence is the unique head of χ.

Unique Parsing Theorem

Theorem 3.3.4 (Unique Parsing Theorem)

Let χ be a formula of LP. Then χ has exactly one of the following forms:

(a) χ is an atomic formula.

(b) χ has exactly one of the forms $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$, where ϕ and ψ are formulas.

(c) χ has the form $(\neg\phi)$, where ϕ is a formula.

Moreover in case (b) the formulas ϕ and ψ are uniquely determined segments of χ. In case (c) the formula ϕ is uniquely determined.

Example

Example 3.3.5

We parse $(p_1 \to ((\neg p_3) \vee \bot))$. It is clearly not atomic, so we check the depths of the initial segments in order to find the head. Thus

$$d[(] = 1 \quad d[(p_1] = 1 \quad d[(p_1 \to] = 1$$

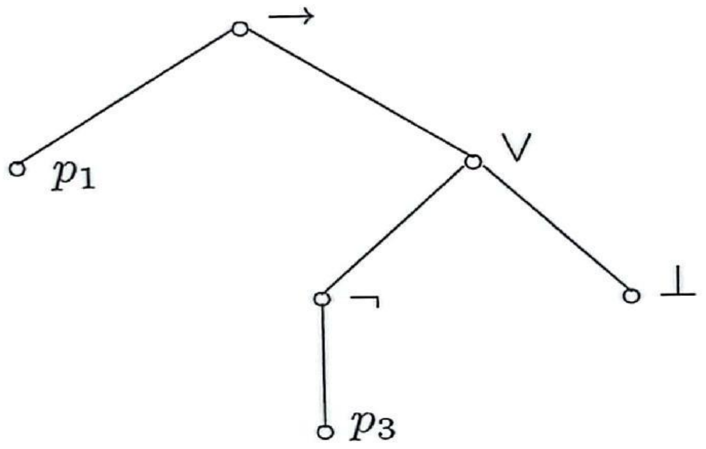
Found it! The third symbol is a functor of depth 1, so it must be the head. Therefore, the formula was built by applying this head to p_1 on the left and $((\neg p_3) \vee \bot)$ on the right. The formula on the left is atomic. A check for the head of the formula on the right goes

$$d[(] = 1, d[((] = 2, d[((\neg] = 2,$$
$$d[((\neg p_3] = 2, d[((\neg p_3)] = 1, d[((\neg p_3) \vee] = 1$$

so again we have found the head.

Starting at the top, we can draw the complete parsing tree :

(3.27)



Scanné avec CamScanner

Non-formula expression

Example 3.3.6

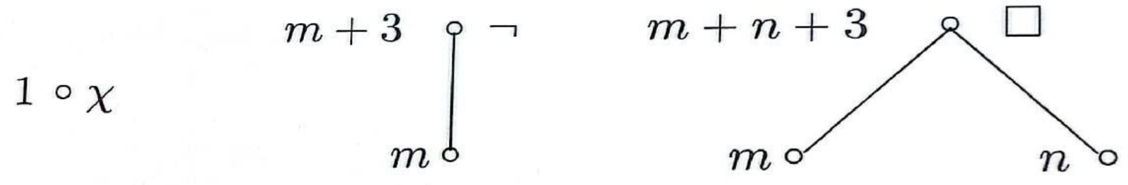
What happens if you try to use the algorithm above to find the parsing tree of an expression s that is not a formula? If you succeed in constructing a parsing tree, then s must be the associated formula of the tree, which is impossible. So you cannot succeed, but what happens instead? There is no question of going into an infinite loop; since the process breaks s down into smaller and smaller pieces, it has to halt after a finite time. What must happen is that you eventually hit an expression which is not an atomic formula and has no identifiable head. At this point the algorithm is telling you that s is not a formula.

For example, we try to parse $(p_0 \to p_1) \to p_2)$.

Recursive definition

The Unique Parsing Theorem often allows us to rewrite compositional definitions in a simpler form. For example, consider the definition :

(3.29)



where χ is atomic and $\Box \in \{\wedge, \vee, \to, \leftrightarrow\}$.

Recursive definition

The definition (3.29) shakes down to the equivalent definition of a function *f* giving the left labels:

$$f(\phi) = 1 \text{ when } \phi \text{ is atomic}$$

$$\textit{(3.30)} \qquad f((\neg\phi)) = f(\phi) + 3;$$

$$f((\phi \ \square\, \psi)) = f(\phi) + f(\psi) + 3 \text{ when } \square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}.$$

Definitions such as (3.30), that define some property of formulas by defining it outright for atomic formulas, and then for complex formulas in terms of their smaller subformulas, are said to be *recursive*.