CS228 Logic for Computer Science 2020

Lecture 2: Propositional logic - unique parsing

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-01-15

Topic 2.1

Extra lecture slides: unique parsing

Matching parentheses

Theorem 2.1

Every $F \in \mathbf{P}$ has matching parentheses, i.e., equal number of '(' and ')'.

Proof.

base case:

atomic formulas have no parenthesis. Therefore, matching parenthesis

induction steps:

We assume $F, G \in \mathbf{P}$ has matching parentheses.
Let n_F and n_G be the number of '(' in F and G respectively.
Trivially, $\neg F$ has matching parentheses.
For some binary symbol $\circ$, the number of both '(' and ')' in $(F \circ G)$ is $n_F + n_G + 1$.

Due to the structural induction, the property holds. $\square$

Prefix of a formula

Theorem 2.2

A proper prefix of a formula is not a formula.

Proof.

We show a proper prefix of a formula is in one of the following forms.

1. strictly more '(' than ')',
2. a (possibly empty) sequence of $\neg$.

Clearly, both the cases are not in **P**.

base case:

A proper prefix of atomic formulas is empty string, which is the second case

...

Exercise 2.1

Give examples of the above two cases

Prefix of a formula II

Proof(contd.)

induction step:

Let $F, G \in \mathbf{P}$.

Consider proper prefix F' of $\neg F$. There are two cases.

- $F' = \epsilon$, case 2
- $F' = \neg F''$, where F'' is a proper prefix of F. Now we again have two subcases for F''.
  - If F'' is in case 1, F' belongs to case 1
  - If $F'' = \neg..\neg$, F' belongs to case 2

...

Prefix of a formula III

Proof(contd.)

By induction F and G have balanced parenthesis.

Consider proper prefix H of $(F \circ G)$, F' be prefix of F, and G' be prefix of G.

- If $H = (F \circ G$, H belongs to case 1 because H has one extra '('
- If $H = (F \circ G'$, H belongs to case 1(why?)

Similarly the following cases are handled

- $H = (F\circ$
- $H = (F$
- $H = (F'$
- $H = ($

$\square$

Exercise 2.2

Complete the (why?).

Unique parsing

Theorem 2.3

Each $F \in \mathbf{P}$ has a unique parsing tree.

Proof.

$\nu(F) \triangleq$ number of logical connectives in F. We apply induction over $\nu(F)$.
base case: $\nu(F) = 0$
F is an atomic formula, therefore has a single node parsing tree.
inductive steps: $\nu(F) = n$
We assume that each F' with $\nu(F') < n$ has a unique parsing tree.
case $F = \neg G$: Since G has a unique parsing tree, F has a unique parsing tree.
case $F = (G \circ H)$:
Suppose there is another formation rule such that $F = (G' \circ' H')$.
Since $F = (G \circ H) = (G' \circ' H')$, $G \circ H) = G' \circ' H')$.
Wlog, G is prefix of G'.
Since $G, G' \in \mathbf{P}$, G can not be proper prefix of G'. Therefore, $G = G'$.
Therefore, $\circ = \circ'$. Therefore, $H = H'$. Therefore, one way to unfold F.
F has a unique parsing tree. $\square$

Parsing algorithm

The previous proofs suggest a parsing algorithm to generate parsing tree.

Algorithm 2.1: PARSER

Input: F : a string over **Vars** and logical connectives
Output: parse tree if $F \in \mathbf{P}$, exception FAIL otherwise

```
if F = p or F = ⊤ or F = ⊥ then return ({F}, ∅) ;
if F = ¬G then
    (V, E) := PARSER(G);
    return (V ∪ {F}, E ∪ {(F, G)});
if F has matching parentheses and F = (F′) then
    G := smallest prefix of F′ where non-zero parentheses match or atomic
     formula
          after a sequence of '¬'s;
    o′H := tail(F′, len(G));
    if the above two match succeed then
        (V1, E1) := PARSER(G);
        (V2, E2) := PARSER(H);
        return (V1 ∪ V2 ∪ {F}, E1 ∪ E2 ∪ {(F, G), (F, H)});
Throw FAIL
```

Parse Algorithm

Exercise 2.3

Show the run of Algorithm 2.1 on the following formulas.

1. $\neg q \Rightarrow (p \oplus r \Leftrightarrow s)$
2. $(\neg(p \Rightarrow q) \wedge (r \Rightarrow (p \Rightarrow q)))$

End of Lecture 2