innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Syntax – Grammar, Parsing, and Parse Trees.

Propositional Logic – Well-Formed Formulas

- A propositional formula is formed – *and therefore known to be* ***well-formed*** – by the rules below and only by the rules below:
  i. every ***propositional atom*** **p** is well-formed
  ii. If φ is well-formed, then so is $\neg\varphi$
  iii. if φ and ψ are well-formed, then so is $\varphi \vee \psi$
  iv. if φ and ψ are well-formed, then so is $\varphi \wedge \psi$
  v. if φ and ψ are well-formed, then so is φ --> ψ

Propositional Logic – Well-Formed Formulas

- Rules for *well-formed formulas* in propositional logic:
  1. every ***propositional atom*** **p** is well-formed
  2. If φ is well-formed, then so is $\neg\varphi$
  3. if φ and ψ are well-formed, then so is $\varphi \vee \psi$
  4. if φ and ψ are well-formed, then so is $\varphi \wedge \psi$
  5. if φ and ψ are well-formed, then so is φ --> ψ
- Note that this definition is inductive:
  - i.e. larger (i.e. *structurally more complex*) formulas are formed out of smaller (i.e. simpler) formulas.

This form of *induction* (*on syntactic terms*) is referred to as a ***structural induction.***

Propositional Logic – Grammar

Well-formed formulas (φ, ψ) can be defined formally using a context free grammar:

1. **φ ---> p**
2. **φ ---> ¬φ**
3. **φ ---> φ ∨ ψ**
4. **φ ---> φ ∧ ψ**
5. **φ ---> φ --> ψ**

(Informal) Definition:

i. every ***propositional atom*** **p** is well-formed
ii. If φ is well-formed, then so is ¬φ
iii. if φ and ψ are well-formed, then so is φ ∨ ψ
iv. if φ and ψ are well-formed, then so is φ ∧ ψ
v. if φ and ψ are well-formed, then so is φ --> ψ

Example 1 - Parsing

Since φ and ψ denote well-formed formulas in this grammar

1. φ ---> p
2. φ ---> $\neg\varphi$
3. φ ---> $\varphi \vee \psi$
4. φ ---> $\varphi \wedge \psi$
5. φ ---> φ --> ψ

we can rewrite the rules to use only φ

Grammar **Gr-PropL-AMB:**

(φ is a well-formed formula):

1. φ ---> p
2. φ ---> $\neg\varphi$
3. φ ---> $\varphi \vee \varphi$
4. φ ---> $\varphi \wedge \varphi$
5. φ ---> φ --> φ

Example 1 - Parsing

Note that parsing refers to:

- *verifying that a given sentence (in this case **a formula**)*
    - *belongs to the language (defined by the **given grammar**)*

Grammar **Gr-PropL-AMB:**

(φ is a well-formed formula):

1. φ ---> p
2. φ ---> $\neg\varphi$
3. φ ---> $\varphi \vee \varphi$
4. φ ---> $\varphi \wedge \varphi$
5. φ ---> φ --> φ

Parse the following formula using the given grammar:

p1 $\wedge$ p2 --> p3 $\vee$ p4

Example 1 - Parsing

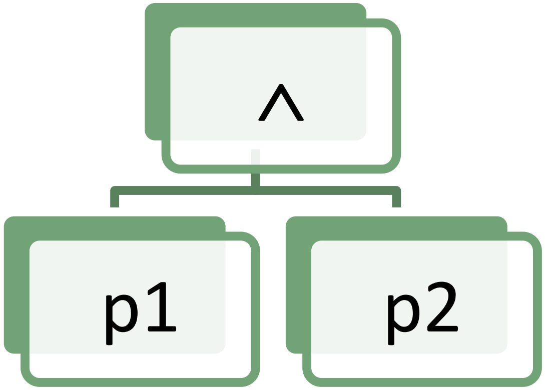
Parsing $p1 \wedge p2 \dashrightarrow p3 \vee p4$

Parsing Step	Rule
$p1 \wedge p2$ --> $p3 \vee p4$	
$\varphi \wedge p2$ --> $p3 \vee p4$	**R1**
$\varphi \wedge \varphi$ --> $p3 \vee p4$	**R1**
φ --> $p3 \vee p4$	**R4**
...	

and the parse tree so far:

Grammar:

1. φ ---> p
2. φ ---> $\neg\varphi$
3. φ ---> $\varphi \vee \varphi$
4. φ ---> $\varphi \wedge \varphi$
5. φ ---> φ --> φ



Example 1 - Parsing

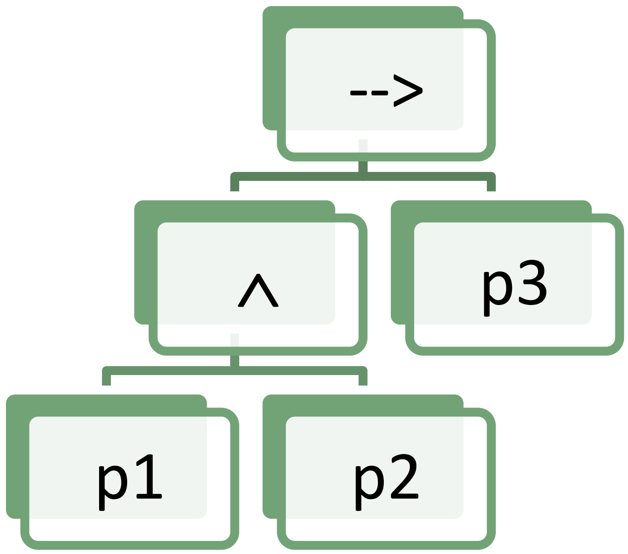
Parsing p1 ∧ p2 --> p3 ∨ p4

Parsing Step	Rule
p1 ∧ p2 --> p3 ∨ p4	
φ ∧ p2 --> p3 ∨ p4	R1
φ ∧ φ --> p3 ∨ p4	R1
φ --> p3 ∨ p4	R4
φ --> φ ∨ p4	R1
φ ∨ p4	R5
…	

Grammar:

1. φ ---> p
2. φ ---> ¬φ
3. φ ---> φ ∨ φ
4. φ ---> φ ∧ φ
5. φ ---> φ --> φ

and the parse tree so far:



Example 1 - Parsing

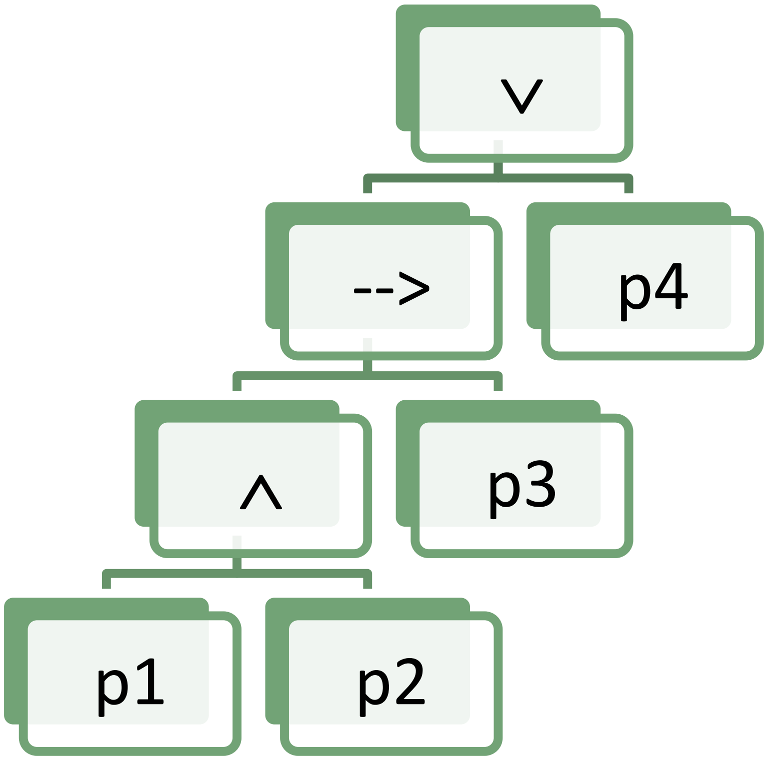
Parsing p1 ∧ p2 --> p3 ∨ p4

Parsing Step	Rule
p1 ∧ p2 --> p3 ∨ p4	
φ ∧ p2 --> p3 ∨ p4	**R1**
φ ∧ φ --> p3 ∨ p4	**R1**
φ --> p3 ∨ p4	**R4**
φ --> φ ∨ p4	**R1**
φ ∨ p4	**R5**
φ ∨ φ	**R1**
φ	**R3**

Grammar:

1. **φ ---> p**
2. **φ ---> ¬φ**
3. **φ ---> φ ∨ φ**
4. **φ ---> φ ∧ φ**
5. **φ ---> φ --> φ**

and the final parse tree :



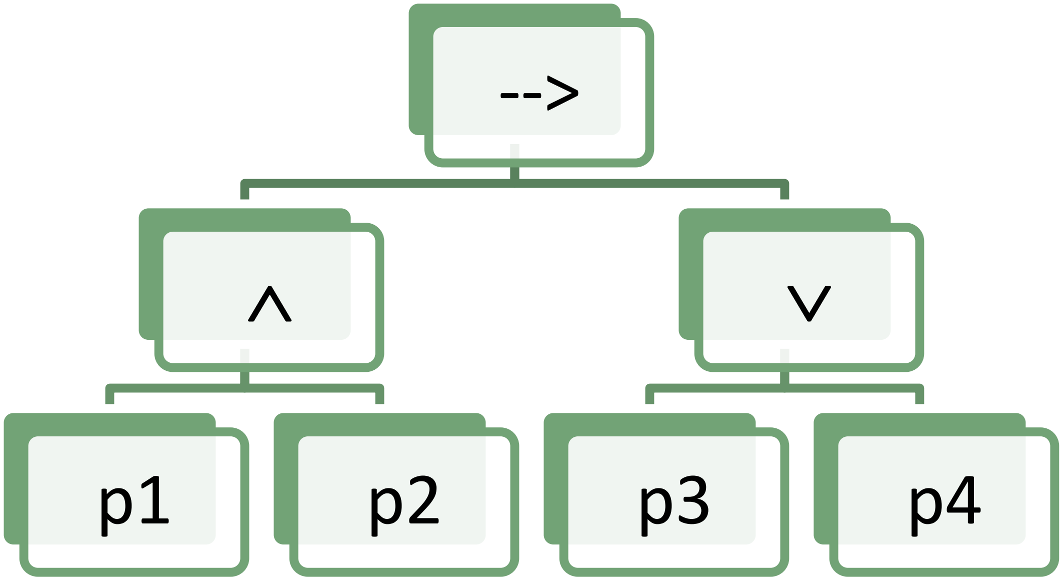
Example 1 – Parsing Alternative

Parsing p1 ∧ p2 --> p3 ∨ p4

Parsing Step	Rule
p1 ∧ p2 --> p3 ∨ p4	
φ ∧ p2 --> p3 ∨ p4	R1
φ ∧ φ --> p3 ∨ p4	R1
φ --> p3 ∨ p4	R4
φ --> φ ∨ p4	R1
φ --> φ ∨ φ	R1
φ --> φ	R3
φ	R5

Grammar:

1. φ ---> p
2. φ ---> ¬φ
3. φ ---> φ ∨ φ
4. φ ---> φ ∧ φ
5. φ ---> φ --> φ



and the alternative parse tree :

BITS Pilani
Pilani Campus

innovate achieve lead

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Syntax – Order of Evaluation

Propositional Logic – Parsing

- Example
  - Parse (i.e. verify) the following formula
    - $\neg p1 \wedge p2$
  - using the grammar ***Gr-PropL-AMB*** in two different ways and
  - draw the corresponding parse trees!

Propositional Logic – Order of Evaluation

- The grammar discussed (**Gr-PropL-AMB**) does not capture ***precedence rules*** (or *order of evaluation*) and therefore results in ambiguity:
  - e.g. $\neg$ p1 $\wedge$ p2 may be treated as
    - NOT (p1 AND p2) or as
    - (NOT p1) AND p2

Syntax: Order of Evaluation: Associativity

- Associativity:
  - Some operators are associative:
    - e.g. $x * (y * z)$ is the same as $(x * y) * z$
      - * is multiplication of integers;   OR
      - * is multiplication of matrices
    - e.g. $(p \wedge q) \wedge r$ is the same as $p \wedge (q \wedge r)$
  - but others are not:
    - e.g. (p --> q) --> r is not the same as p --> (q --> r)
      - Why not?

Order of Evaluation: Left-Associativity vs. Right-Associativity

- Example:
  - Assignment operator in C:
      - x = y = z + 2;
    - = is a right-associative operator
  - Division:
      - x/y/z
    - / is usually considered left-associative

Addressing ambiguity – Order of Evaluation

- One way of addressing such ambiguity in specification (i.e. in grammars):
  - *leave it to the user* (i.e. who is writing the formulas) to specify an order or evaluation
    - *for instance*, parenthesize all expressions to avoid ambiguity

Grammar

(Gr-PropL-OE-1):

ϕ ---> p

ϕ ---> '(' '$\neg$' ϕ ')'

ϕ ---> '(' ϕ '$\vee$' ϕ ')'

ϕ ---> '(' ϕ '$\wedge$' ϕ ')'

ϕ ---> '(' ϕ '-->' ϕ ')'

Exercises:

1. Specify the two forms of
   $\neg$ p1 $\wedge$ p2
   using this grammar.
   [**Hint:** *You need to add a pair of (outermost) parentheses that is redundant.* **End of Hint.]**

2. Show how those two forms can be parsed.

3. (Trivial) Question: What is the drawback of this approach?