University of Central Lancashire
UCLan

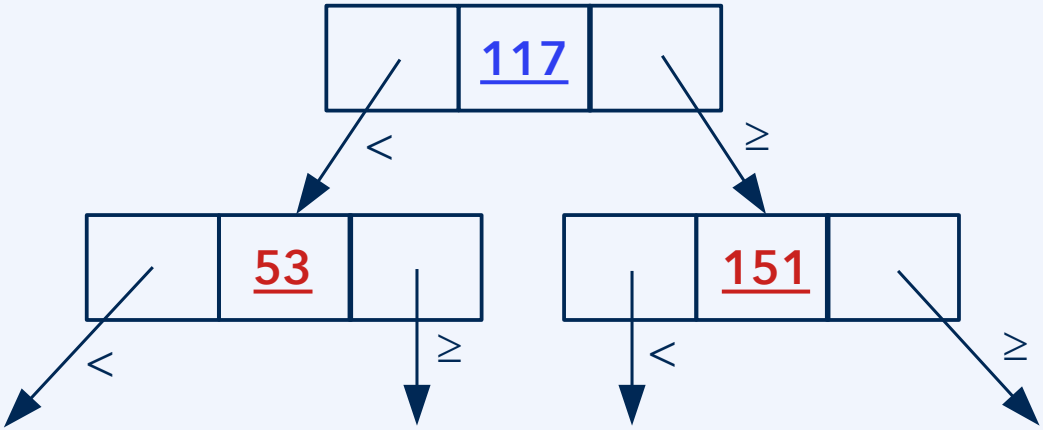
Parse trees: Tutorial 4.1 problem

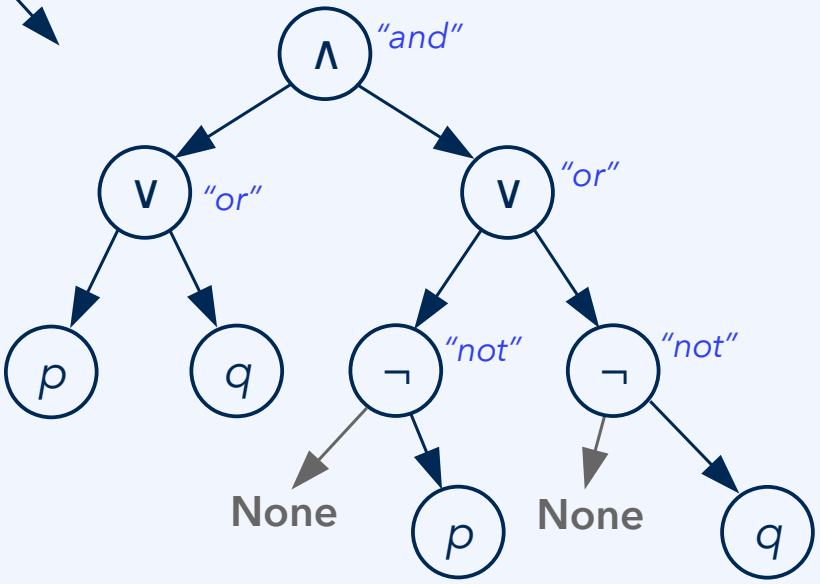
CO2412

8th March 2022

University of Central Lancashire
UCLan

Parse trees for propositional logic statements





see parse-tree-discussion notebook

Parse trees for propositional logic statements

Task 4.1b: **Create a Propositional object for the statement $((p \vee q) \wedge (\neg p \vee \neg q))$ and print its truth table.**

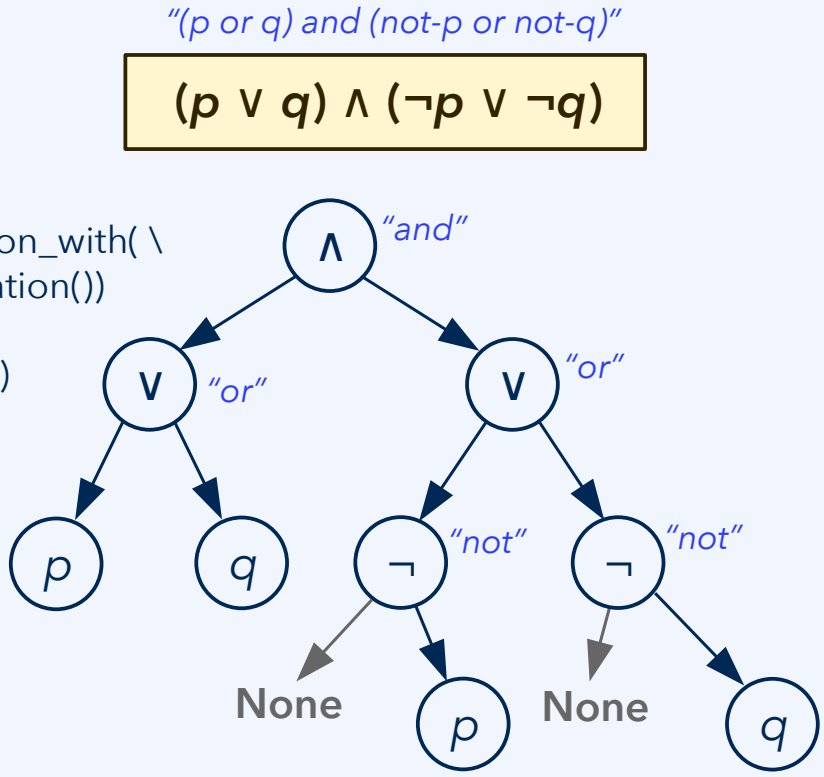
```
p_or_q = Propositional("p").disjunction_with( \
                                    Propositional("q"))
not_p_or_not_q = Propositional("p").negation().disjunction_with( \
                                    Propositional("q").negation())

statement_S = p_or_q.conjunction_with(not_p_or_not_q)
```

```
Truth table of ((p or q) and (not p or not q))

False(p)        False(q)        <>       False
False(p)         True(q)        <>       True
 True(p)        False(q)        <>       True
 True(p)         True(q)        <>       False

True 2 times, false 2 times, undefined 0 times
```



University of Central Lancashire
UCLan

Parse trees for propositional logic statements

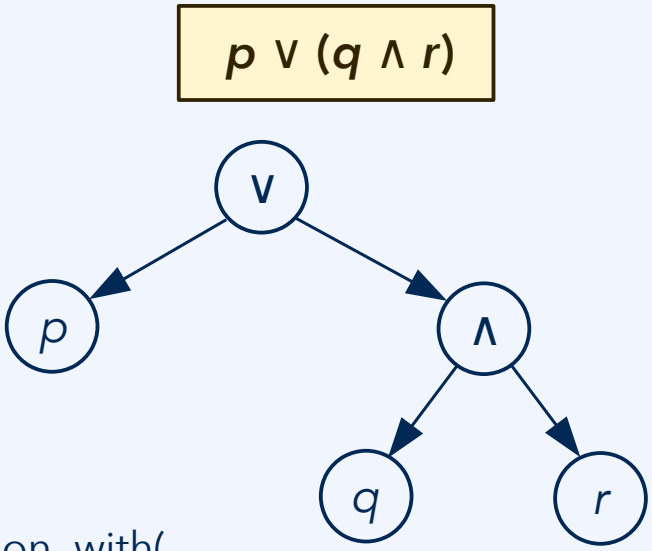
Task 4.1e: Create a Propositional object that generates the truth table …

p	*q*	*r*	specification	$q \wedge r$	$p \vee (q \wedge r)$
False	False	False	False	False	False
False	False	True	False	False	False
False	True	False	False	False	False
False	True	True	True	**True**	True
True	False	False	True	False	True
True	False	True	True	False	True
True	True	False	True	False	True
True	True	True	True	**True**	True

Parse trees for propositional logic statements

Task 4.1e: Create a Propositional object that generates the truth table …

p	*q*	*r*	*p* ∨ (*q* ∧ *r*)
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True




```
statement_T = Propositional("p").disjunction_with(
    Propositional("q").conjunction_with(Propositional("r"))
)
```

University of Central Lancashire
UCLan

Parse trees for propositional logic statements

Task 4.1c: What is the truth table for the statement $(p \wedge q) \leftrightarrow (q \wedge r)$?

p	q	r	$p \wedge q$	$q \wedge r$	$(p \wedge q) \leftrightarrow (q \wedge r)$
False	False	False	False	False	**True**
False	False	True	False	False	**True**
False	True	False	False	False	**True**
False	True	True	False	True	False
True	False	False	False	False	**True**
True	False	True	False	False	**True**
True	True	False	True	False	False
True	True	True	True	True	**True**

University of Central Lancashire
UCLan

Parse trees for propositional logic statements

Task 4.1d: Implement the logical equivalence (biconditionality) operator.

```
163
164         ############
165         # NEW CODE #
166         ############
167         #
168         elif self._item == "<->":
169             if left_evaluation == "undefined" or right_evaluation == "undefined":
170                 return "undefined"
171             else:
172                 return (left_evaluation == right_evaluation)
173
```

In [3]:

```
1  # Task 4.1c/d: What is the truth table for the statement (p ∧ q) ↔ (q ∧ r)?
2  #
3  p_and_q = Propositional("p").conjunction_with(Propositional("q"))
4  q_and_r = Propositional("q").conjunction_with(Propositional("r"))
5  statement_R = p_and_q.biconditional_with(q_and_r)
6
7  print("Truth table of", statement_R.to_string(), "\n")
8  (t, f, u) = statement_R.print_whole_truth_table()
9  print("\nTrue", t, "times, false", f, "times, undefined", u, "times")
```

```
Truth table of ((p and q) <-> (q and r))

False(p)        False(q)        False(r)        <>      True
False(p)        False(q)         True(r)        <>      True
False(p)         True(q)        False(r)        <>      True
False(p)         True(q)         True(r)        <>      False
 True(p)        False(q)        False(r)        <>      True
 True(p)        False(q)         True(r)        <>      True
 True(p)         True(q)        False(r)        <>      False
 True(p)         True(q)         True(r)        <>      True

True 6 times, false 2 times, undefined 0 times
```