

Signed Formula

- ▷ **Signed formula**: an expression $A = b$, where A is a formula and b a boolean value.
- ▷ A signed formula $A = b$ is **true** in an interpretation I , denoted by $I \models A = b$, if $I(A) = b$.
- ▷ If $A = b$ is true in I , we also say that I is a **model of $A = b$** , or that I **satisfies $A = b$** .
- ▷ A signed formula is **satisfiable** if it has a model.

Some properties

1. For every formula A and interpretation I **exactly one** of the signed formulas $A = 1$ and $A = 0$ is true in I .
2. A formula A is **satisfiable** if and only if so is the signed formula $A = 1$.

How to find a model of a signed formula?

Example: $(A \rightarrow B) = 1$.

Operation table for \rightarrow :

	\rightarrow	1	0
1		1	0
0		1	1

So $(A \rightarrow B) = 1$ if and only if $A = 0$ OR $B = 1$.

Likewise, $(A \rightarrow B) = 0$ if and only if $A = 1$ AND $B = 0$.

Branch Expansion Rules

$$(A_1 \wedge \dots \wedge A_n) = 0 \rightsquigarrow A_1 = 0 \mid \dots \mid A_n = 0$$

$$(A_1 \wedge \dots \wedge A_n) = 1 \rightsquigarrow A_1 = 1, \dots, A_n = 1$$

$$(A_1 \vee \dots \vee A_n) = 0 \rightsquigarrow A_1 = 0, \dots, A_n = 0$$

$$(A_1 \vee \dots \vee A_n) = 1 \rightsquigarrow A_1 = 1 \mid \dots \mid A_n = 1$$

$$(A_1 \rightarrow A_2) = 0 \rightsquigarrow A_1 = 1, A_2 = 0$$

$$(A_1 \rightarrow A_2) = 1 \rightsquigarrow A_1 = 0 \mid A_2 = 1$$

$$(\neg A_1) = 0 \rightsquigarrow A_1 = 1$$

$$(\neg A_1) = 1 \rightsquigarrow A_1 = 0$$

$$(A_1 \leftrightarrow A_2) = 0 \rightsquigarrow A_1 = 0, A_2 = 1 \mid A_1 = 1, A_2 = 0$$

$$(A_1 \leftrightarrow A_2) = 1 \rightsquigarrow A_1 = 0, A_2 = 0 \mid A_1 = 1, A_2 = 1$$

Branch Closure Rules

If a branch contains both $p = 0$ and $p = 1$ for some atom p , then the branch is marked as **closed**.

If a branch contains either $\top = 0$ or $\perp = 1$, then the branch is marked as **closed**.

A Semantic Tableau

$$(\neg(q \vee p \rightarrow p \vee q)) = \mathbf{1}$$

A Semantic Tableau

$$(\neg(q \vee p \rightarrow p \vee q)) = 1$$

|

$$(q \vee p \rightarrow p \vee q) = 0$$

A Semantic Tableau

$$(\neg(q \vee p \rightarrow p \vee q)) = 1$$

|

$$(q \vee p \rightarrow p \vee q) = 0$$

|

$$\begin{cases} (q \vee p) = 1 \\ (p \vee q) = 0 \end{cases}$$

A Semantic Tableau

$$(\neg(q \vee p \rightarrow p \vee q)) = 1$$

|

$$(q \vee p \rightarrow p \vee q) = 0$$

|

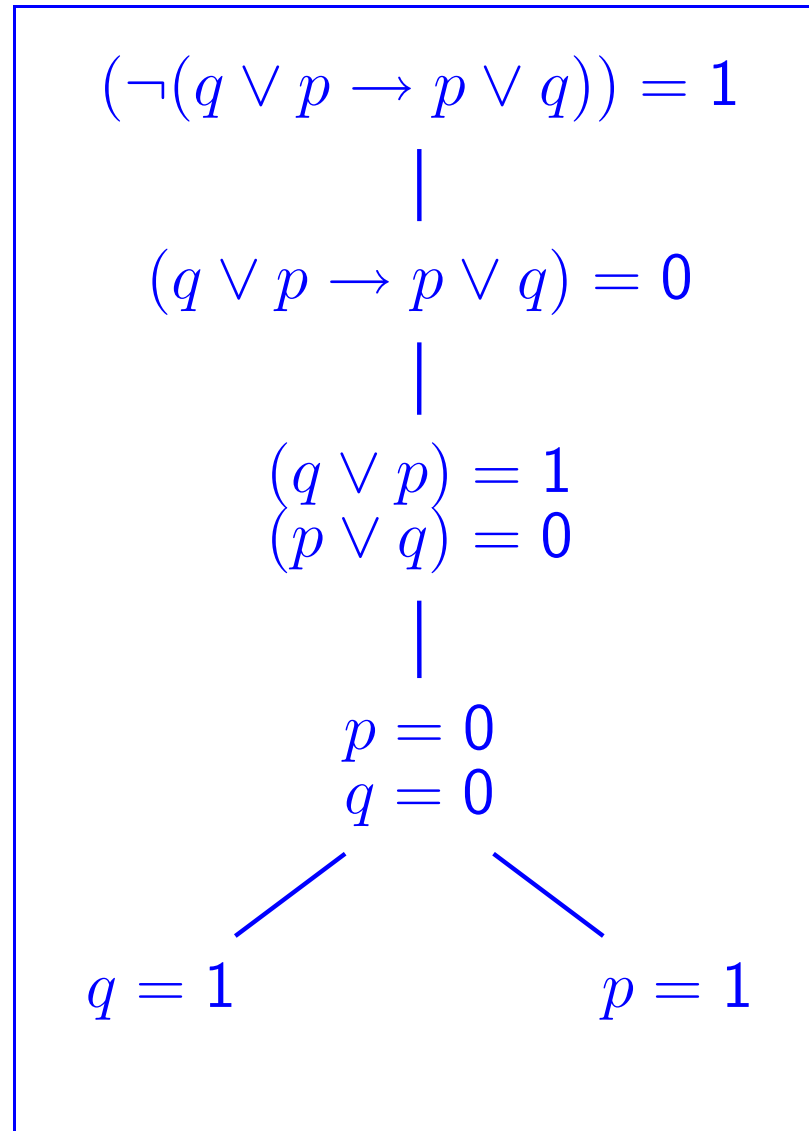
$$\begin{cases} (q \vee p) = 1 \\ (p \vee q) = 0 \end{cases}$$

|

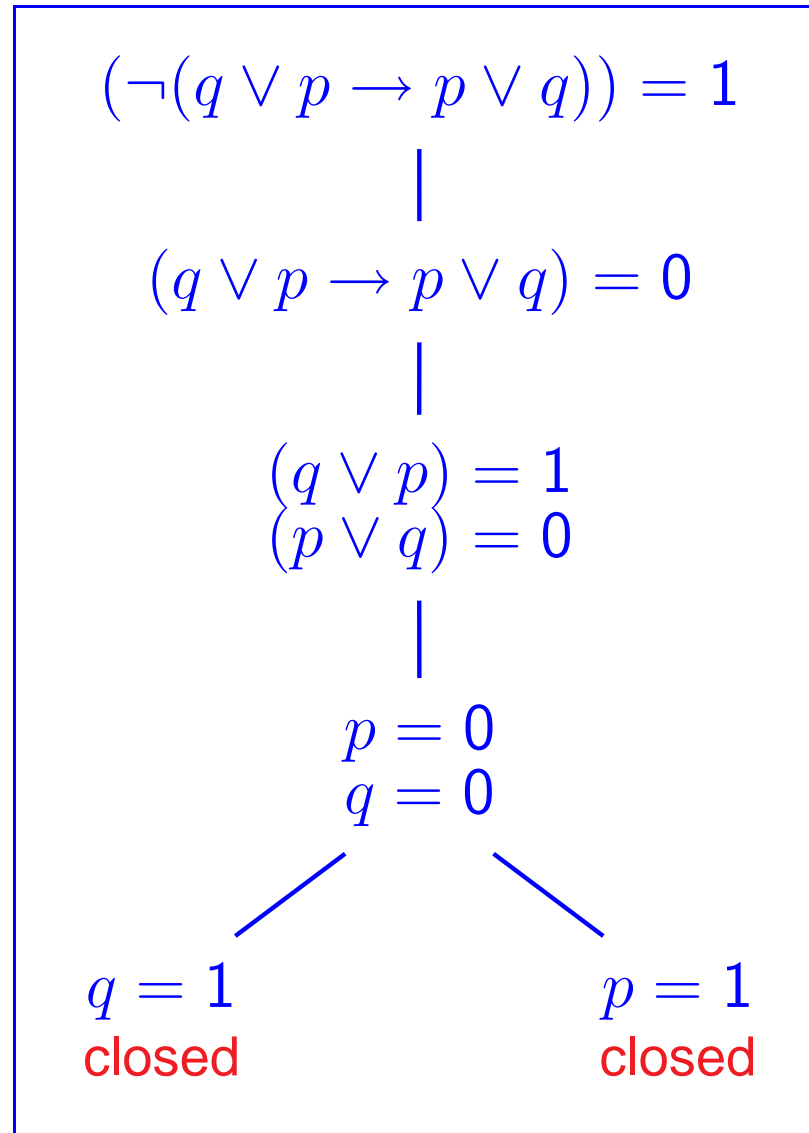
$$p = 0$$

$$q = 0$$

A Semantic Tableau



A Semantic Tableau



Checking satisfiability, equivalence etc.

Terminated tableau: if either all rules have been applied of all branches are closed.

- ▶ A formula F is **satisfiable** if and only if every (some) terminated tableau for $F = 1$ has an open branch.
- ▶ A formula F is **valid** if and only if in every (some) terminated tableau for $F = 0$ all branches are closed.
- ▶ Formulas F_1 and F_2 are **equivalent** if and only if in every (some) terminated tableau for $(F_1 \leftrightarrow F_2) = 0$ all branches are closed.

Data Structures for Large Propositional Formulas

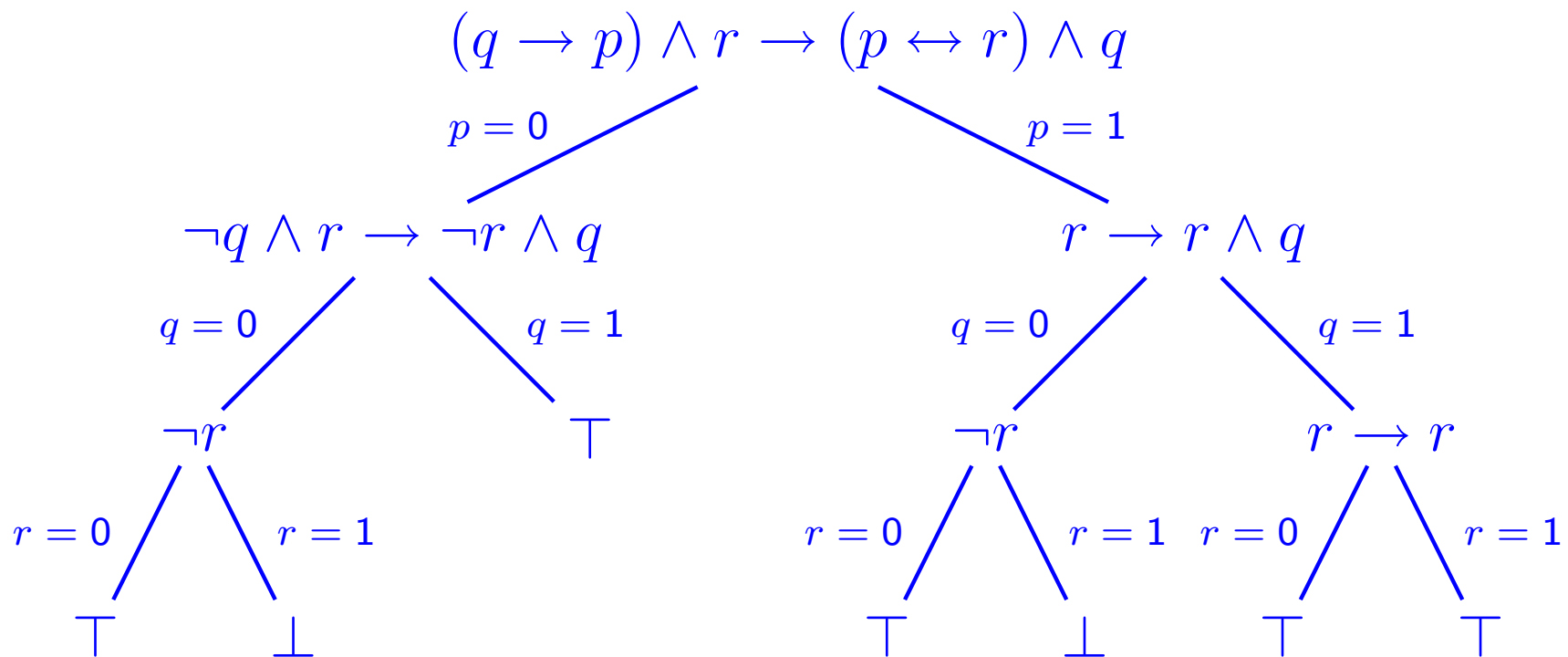
Suppose that **large propositional formulas are reused over and over again**. For example, we may

- ▶ Build a **conjunction** of several formulas;
- ▶ **Negate** a formula;
- ▶ Check if two formulas are **equivalent** . . .

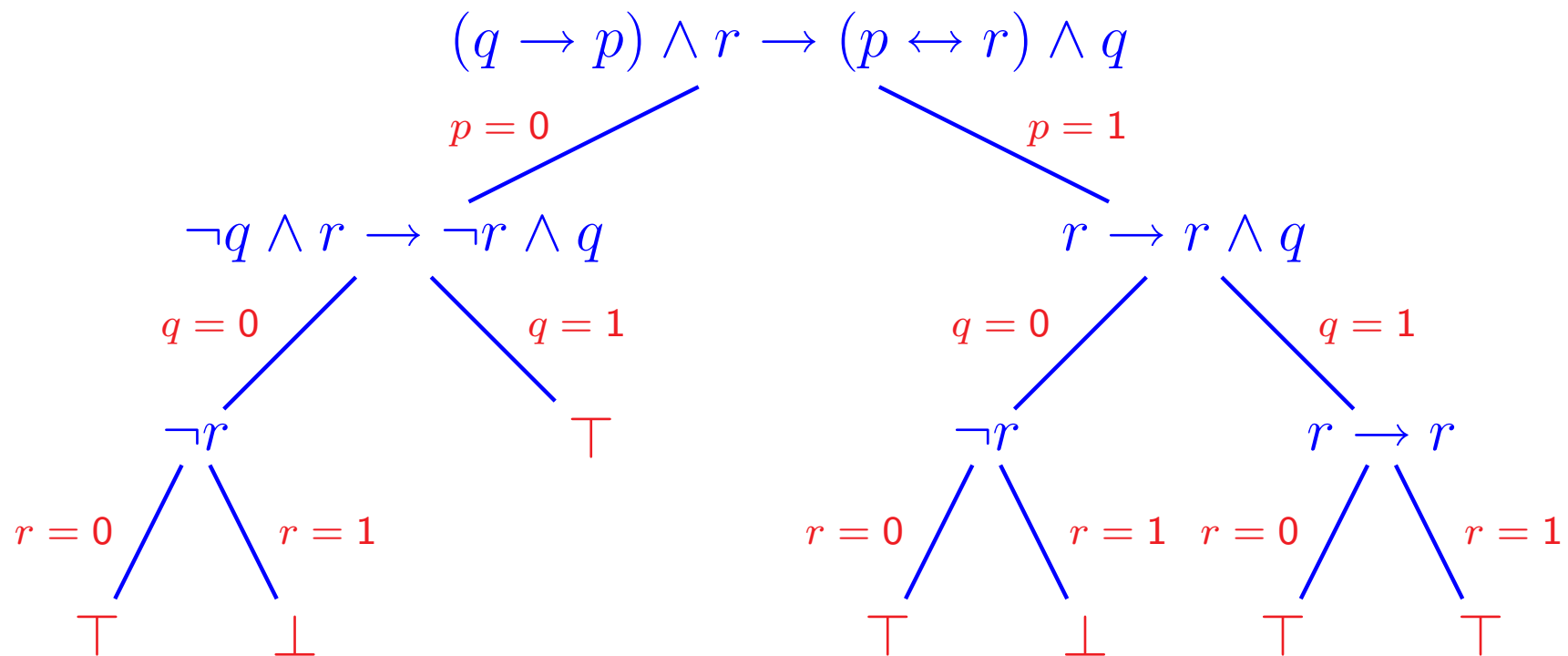
One needs **data structures** which

- ▶ give **compact representation** of formulas, or the boolean functions represented by the formulas;
- ▶ facilitate **boolean operations** on this formulas, for example, taking conjunction of several formulas;
- ▶ facilitate **checking properties of formulas**, such as satisfiability or equivalence checking.

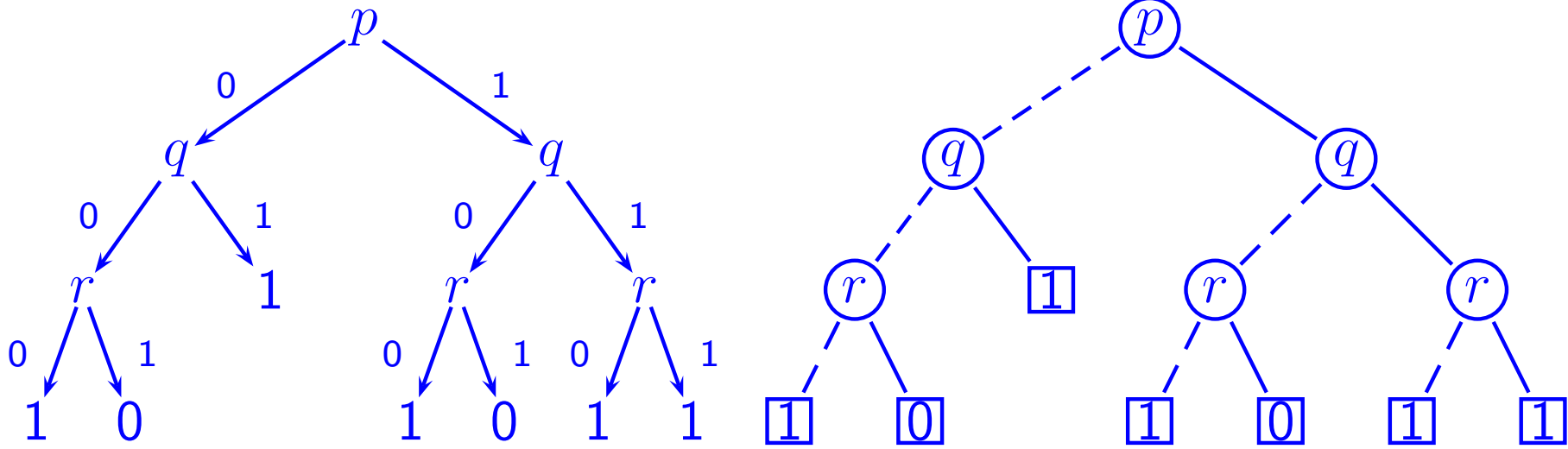
Splitting Tree



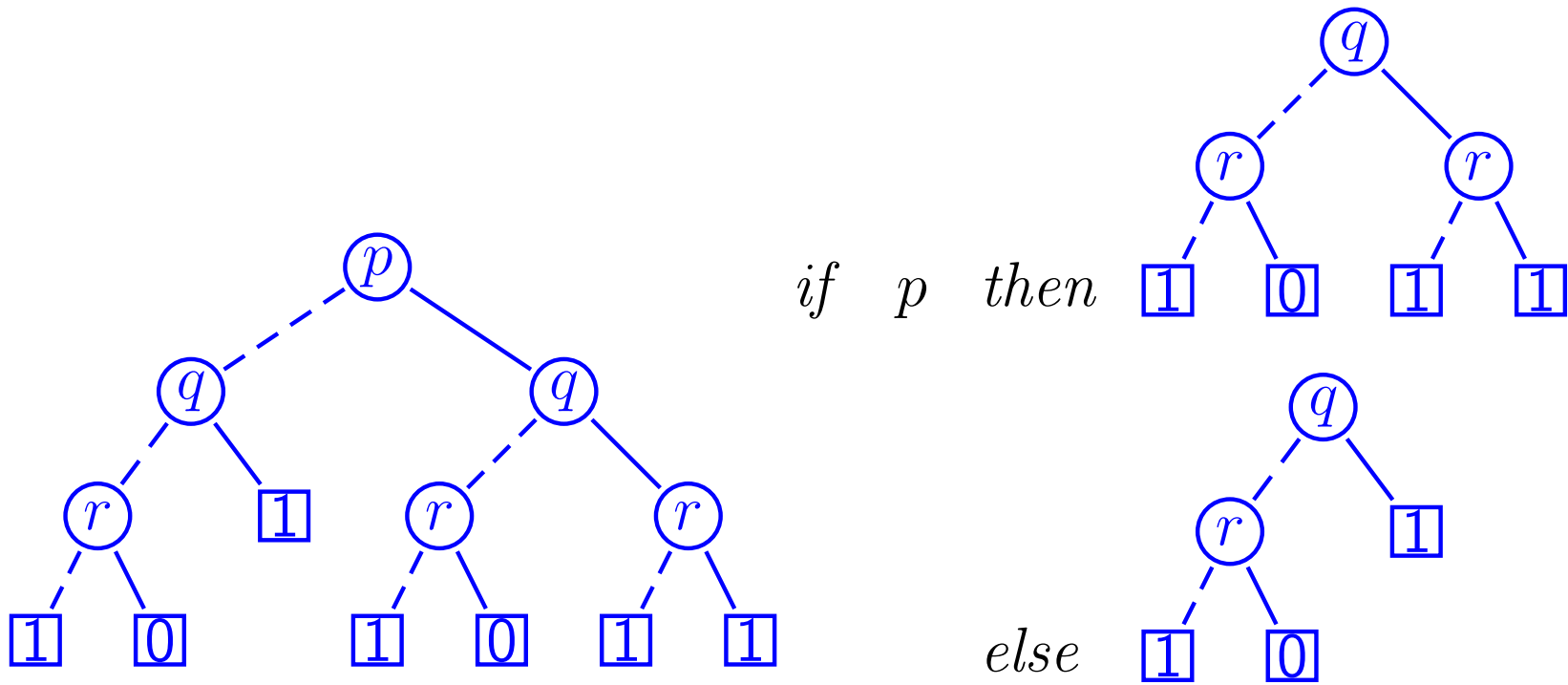
Splitting Tree



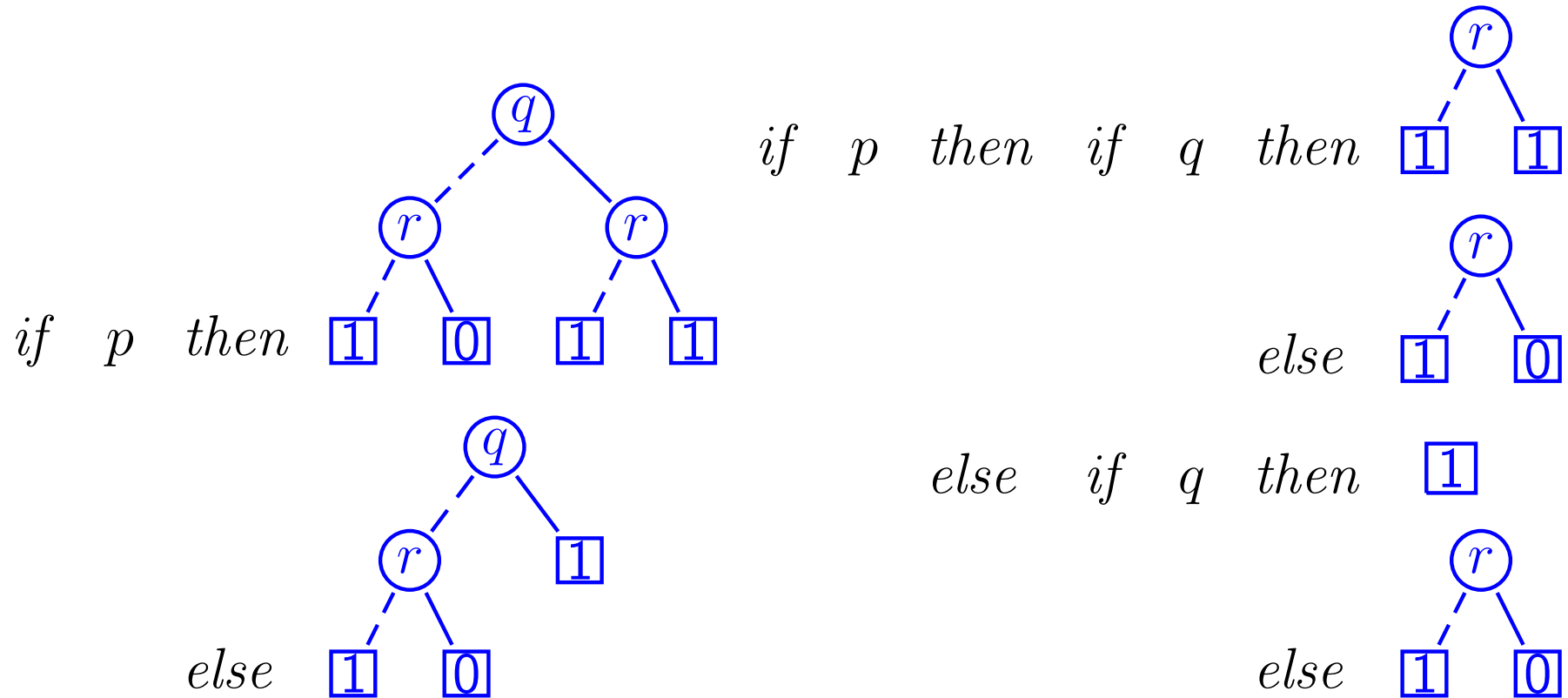
Binary Decision Tree



Tests correspond to “if-then-else”



Tests correspond to “if-then-else”



Tests correspond to “if-then-else”

if p then if q then if r then \top
else \top
else if r then \top
else \perp
else if q then \top
else if r then \top
else \perp

if A then B else C $\equiv (A \rightarrow B) \wedge (\neg A \rightarrow C)$.

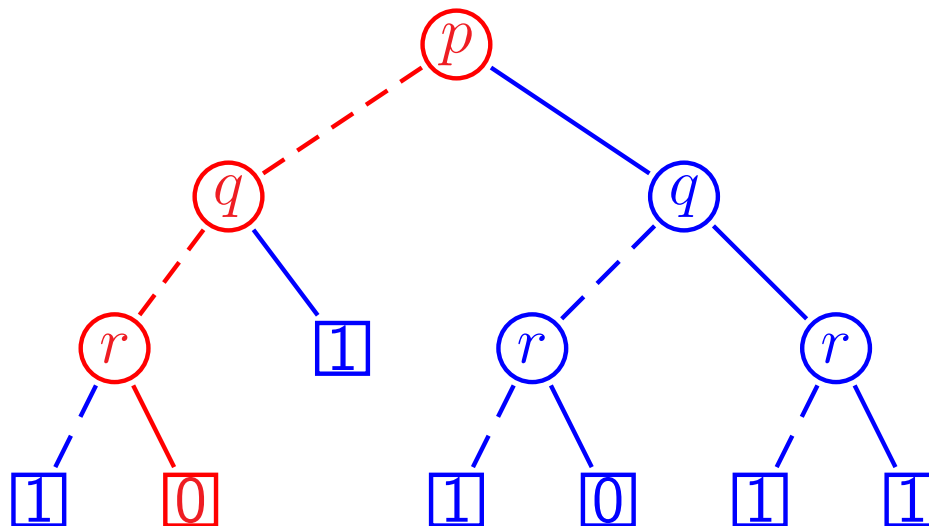
If-Then-Else Normal Form

- ▶ The only connectives are **if-then-else**, \top and \perp ;
- ▶ The if-formula A in *if A then B else C* is atomic.

Evaluating the Formula

To evaluate the formula on the interpretation

$\{p \mapsto 0, q \mapsto 0, r \mapsto 1\}$:



Algorithm for Building Binary Decision Trees

```
procedure bdt(A)  
input: propositional formula A  
output: a binary decision tree  
parameters: function select_atom  
begin  
  A := simplify(A)  
  if A =  $\perp$  then return  $\boxed{0}$   
  if A =  $\top$  then return  $\boxed{1}$   
  p := select_atom(A)  
  return tree(bdt( $A_p^\perp$ ), p, bdt( $A_p^\top$ ))  
end
```

Properties

- ▶ Satisfiability checking can be done in **linear time**;
- ▶ Validity checking can be done in **linear time**;
- ▶ Equivalence checking is **very hard**.
- ▶ Some boolean operations, e.g., conjunction, are **hard to implement**.

Are binary decision trees **compact**?