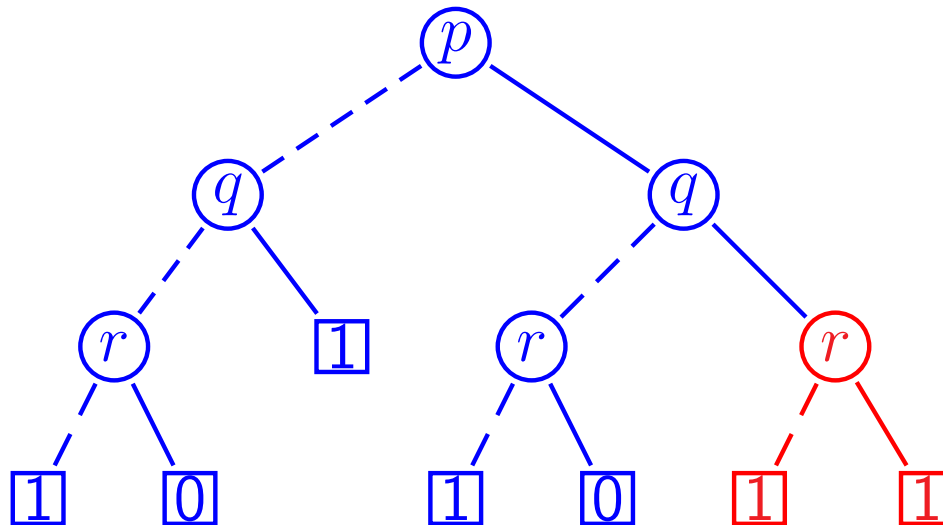


Redundant Tests

Are binary decision trees compact?

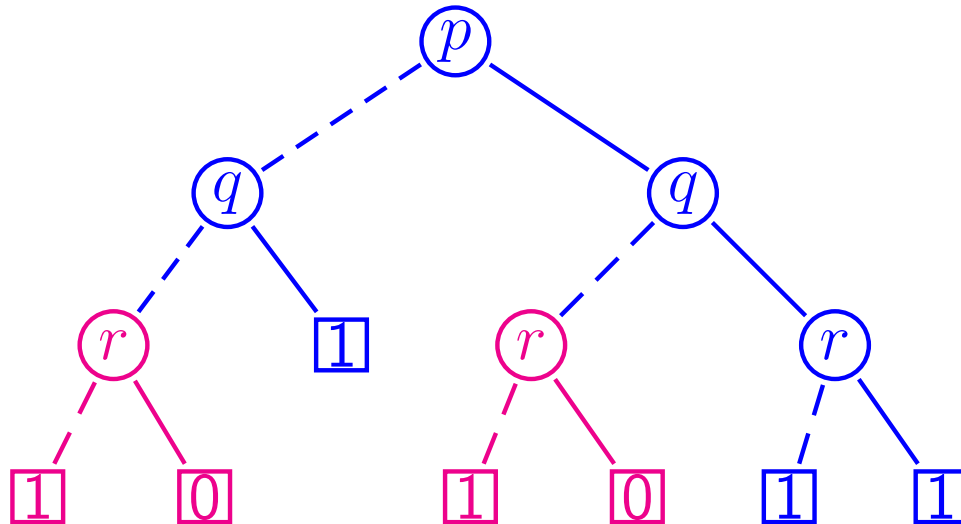
No: they may contain **redundant tests:**



Isomorphic Subtrees

Are binary decision trees compact?

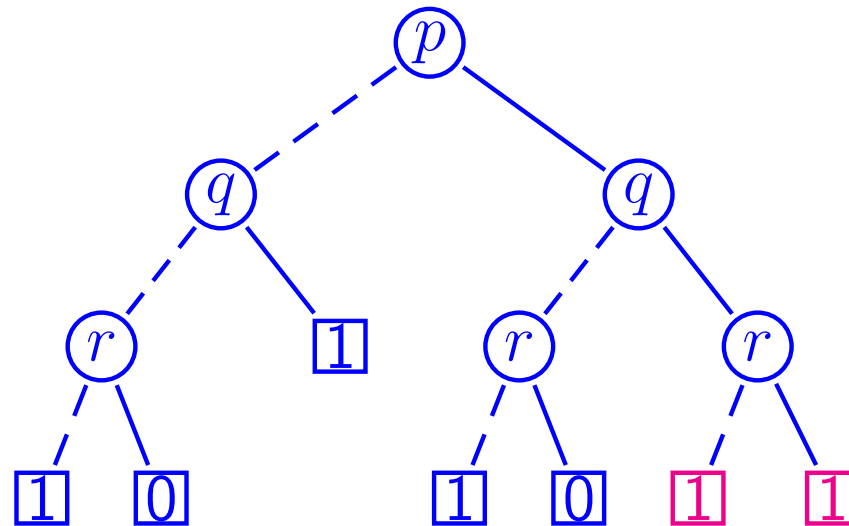
No: they may contain isomorphic subtrees:



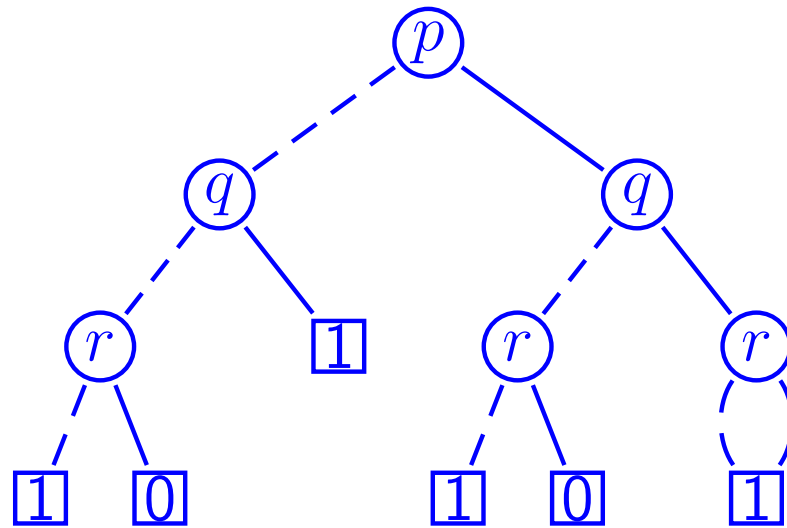
Binary Decision Diagrams

A **binary decision diagrams** or **BDD** is a dag built like a binary decision tree but containing **no redundant tests** and **no isomorphic subtrees**.

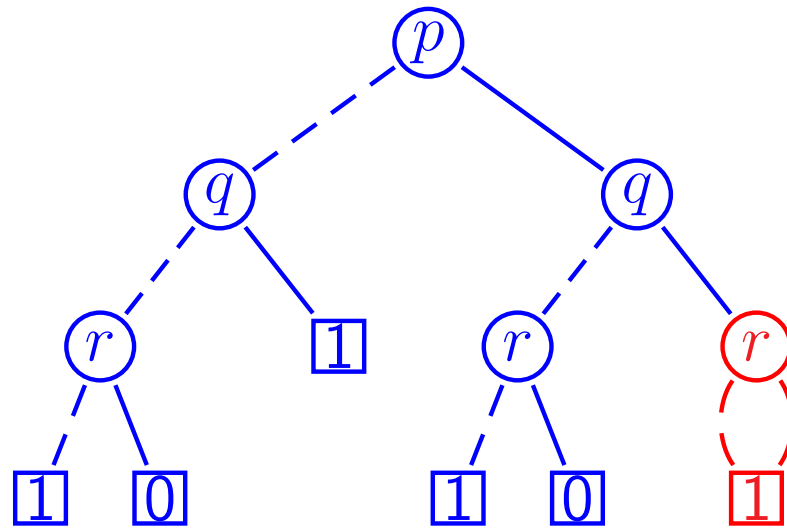
Transforming Binary Decision Tree into a BDD



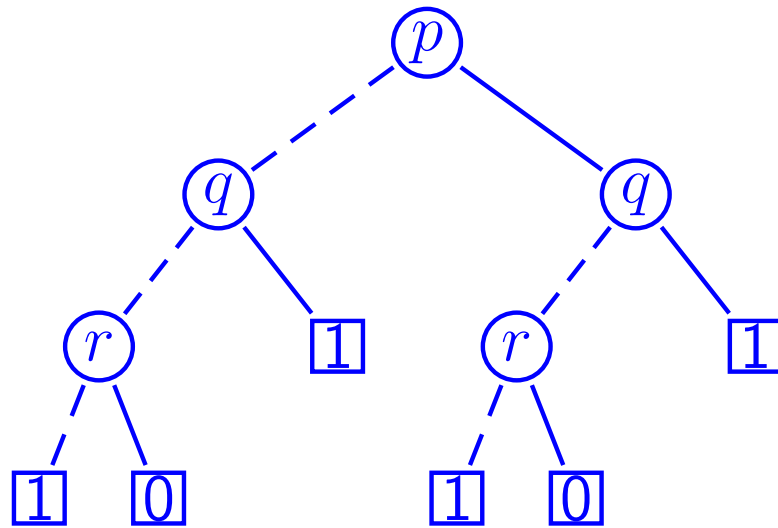
Transforming Binary Decision Tree into a BDD



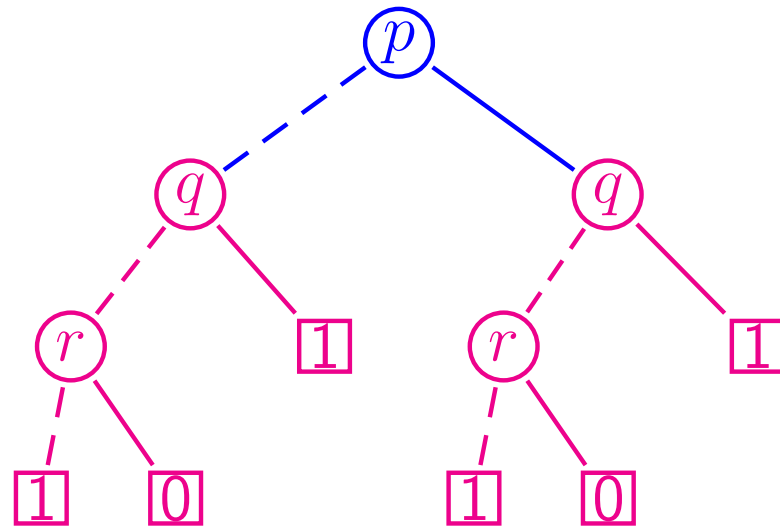
Transforming Binary Decision Tree into a BDD



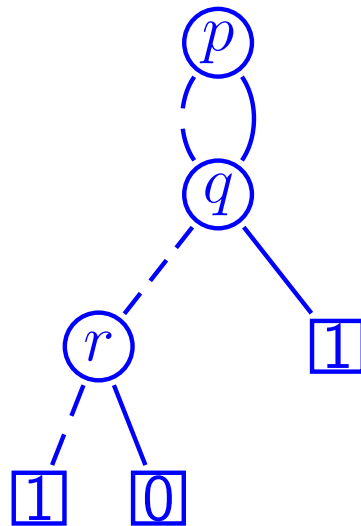
Transforming Binary Decision Tree into a BDD



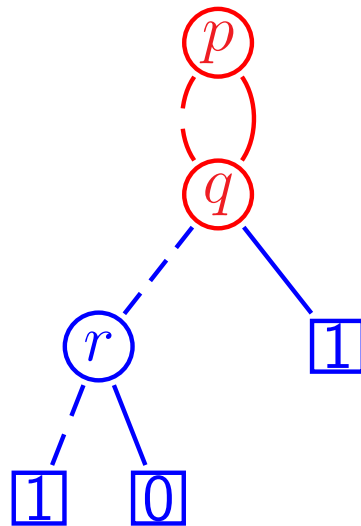
Transforming Binary Decision Tree into a BDD



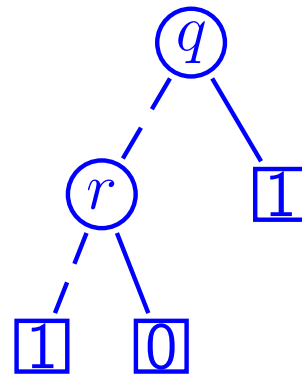
Transforming Binary Decision Tree into a BDD



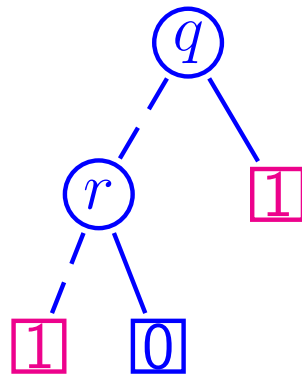
Transforming Binary Decision Tree into a BDD



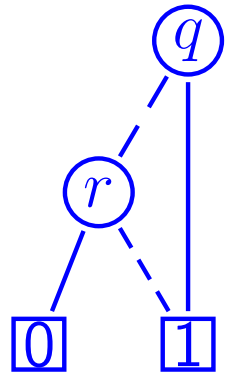
Transforming Binary Decision Tree into a BDD



Transforming Binary Decision Tree into a BDD



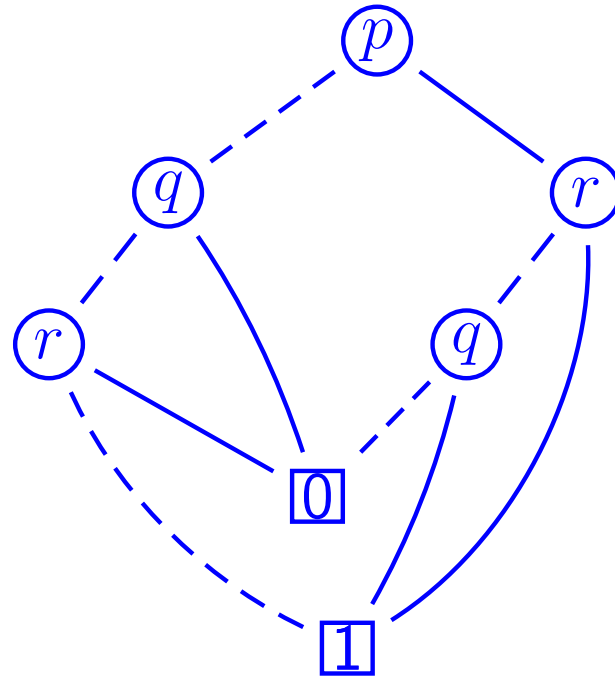
Transforming Binary Decision Tree into a BDD



Properties

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

OBDDs



Idea:

- ▷ introduce an order $>$ on atoms;
- ▷ make tests in this order.

Properties

- ▷ Satisfiability checking can be done in constant time;
- ▷ Validity checking can be done in constant time;
- ▷ Equivalence checking can be done in constant time;
- ▷ Boolean operations e.g., conjunction, are easy to implement.

Algorithms on OBDDS

Suppose we have to compute an OBDD that represents a boolean function $f(b_1, \dots, b_n)$, for example $b_1 \vee \dots \vee b_n$ and we have already built OBDDs for b_1, \dots, b_n .

- ▶ We assume a **global dag** that contains all OBDDs so that all isomorphic subdags are **shared**.
- ▶ Use the property

$$\begin{aligned} & f(\textit{if } p \textit{ then } l_1 \textit{ else } r_1, \\ & \quad \dots, \\ & \quad \textit{if } p \textit{ then } l_n \textit{ else } r_n) = \\ & \textit{if } p \textit{ then } f(l_1, \dots, l_n) \textit{ else } f(r_1, \dots, r_n). \end{aligned}$$

Integrating a node in a dag

procedure *integrate*(n_1, p, n_2, D)

parameters: global dag D

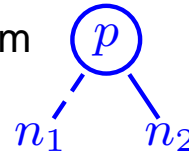
input: nodes n_1, n_2 in D representing formulas F_1, F_2 , variable p

output: node n in (modified) D representing *if p then F_1 else F_2*

begin

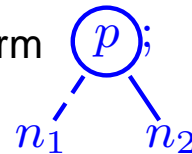
if $n_1 = n_2$ then return n_1 ;

if D contains a node n having the form



then return n ;

add to D a new node n of the form



return n

end