Constructing Binary Decision Diagrams (BDDs)

Author: Ben Andrew

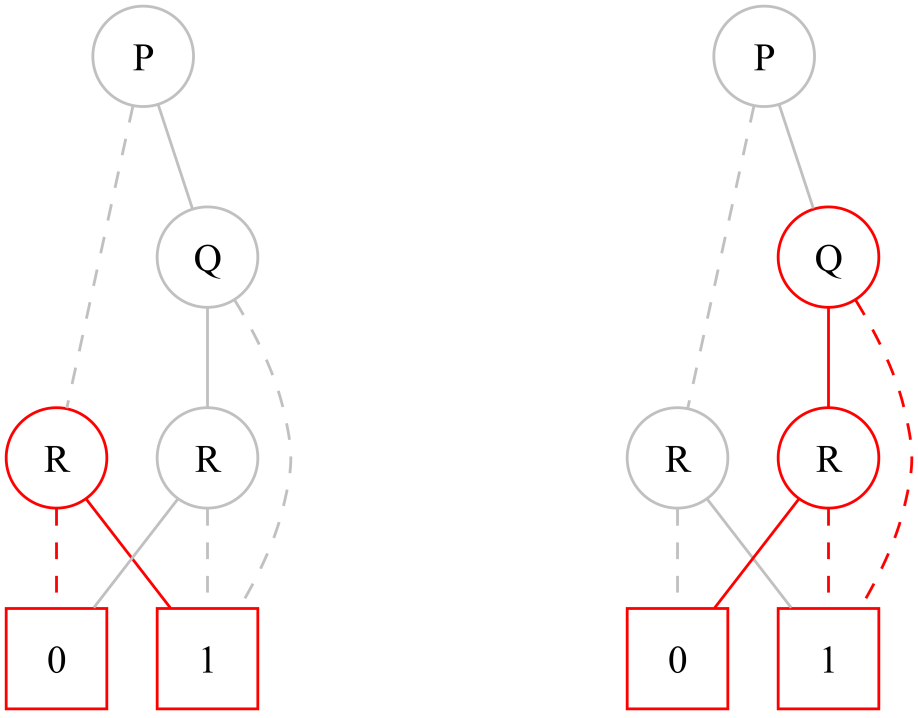
March 17, 2024

We cover a couple of examples of constructing BDDs from propositional formulae. This article assumes knowledge of the basics of BDDs.

Note: our ordering of propositional letters is alphabetical, e.g. P then Q then R.

0.1 Aside: Sub-BDDs?

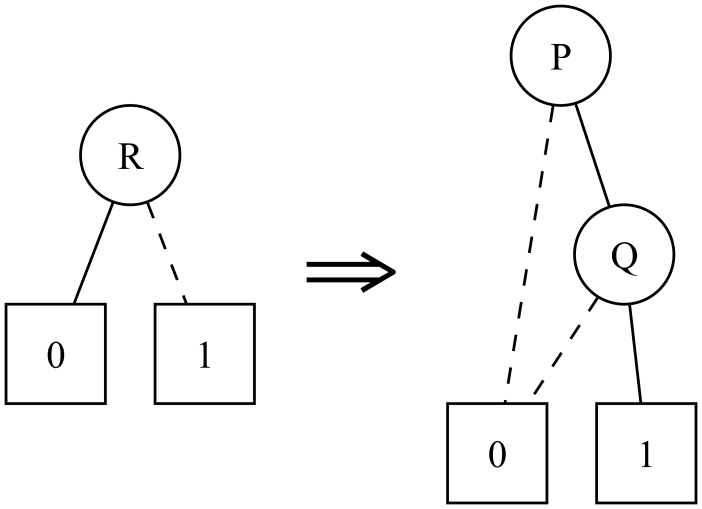
In the following text I use the term 'sub-BDD'. Below we see highlighted the two sub-BDDs that we get for an example BDD, the first if we take the $P = 0$ branch, the second the $P = 1$ branch.



We now move on to the first example.

1 BDD of $\neg R \Rightarrow (Q \wedge P)$

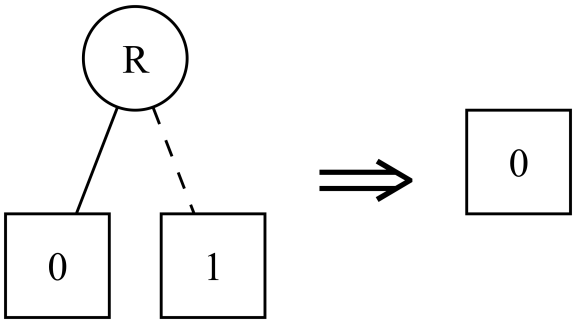
We start by computing the BDD for $\neg R \Rightarrow (Q \wedge P)$. On the left of the image below we see the BDD for $\neg R$, and on the right the BDD for $Q \wedge P$. The implication symbol inbetween them is drawn as a reminder of the connective that we're applying.



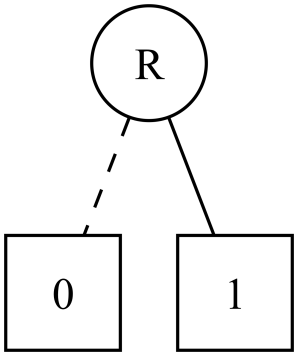
We case split on the P node, for each BDD getting the sub-BDD following the $P = 0$ path, and the $P = 1$ path.

1.1 Case: $P = 0$

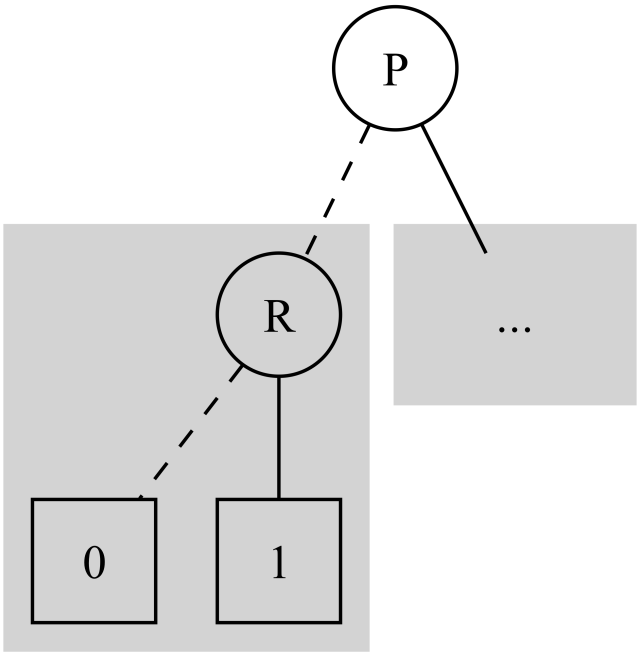
The left BDD has no P node, and so is left unchanged. In the right BDD we simply get the 0 (zero) leaf node.



We could now case split on R and continue the recursion, but now that we've encountered a leaf node we can just compute the result of the connective applied to each of the leaves of the left sub-BDD with the boolean value on the right. This would give us $0 \Rightarrow 0 = 1$ if $R = 1$ and $1 \Rightarrow 0 = 0$ if $R = 0$. Effectively this has swapped the outgoing edges of the R node, producing the below BDD.



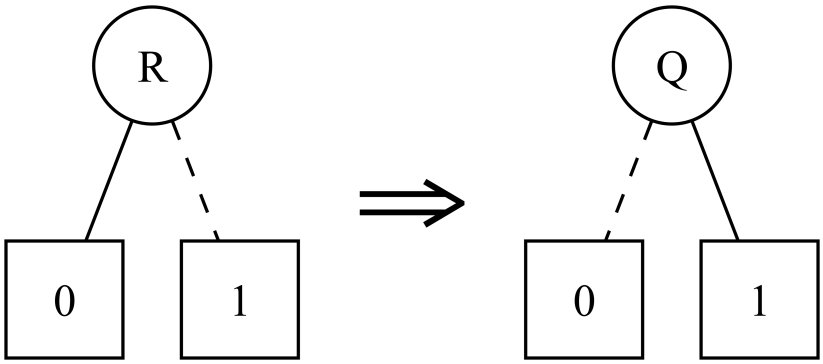
We can now recurse upwards and attach the $P = 0$ edge to this new sub-BDD.



We now turn to the $P = 1$ case.

1.2 Case: $P = 1$

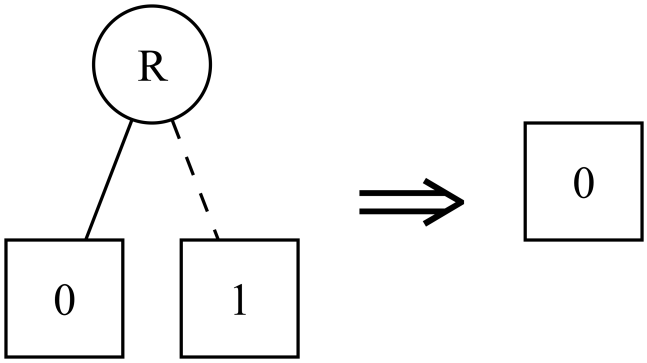
We need the sub-BDDs produced by following the $P = 1$ edge for each of the initial BDDs, giving us



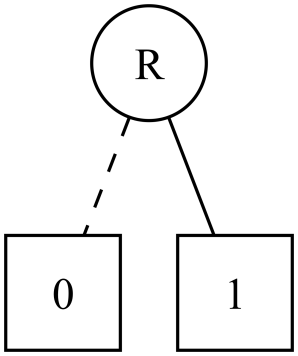
As we don't have any leaf nodes, we can recurse again, now on Q.

1.3 Case: $Q = 0$

Following the $Q = 0$ path gives us

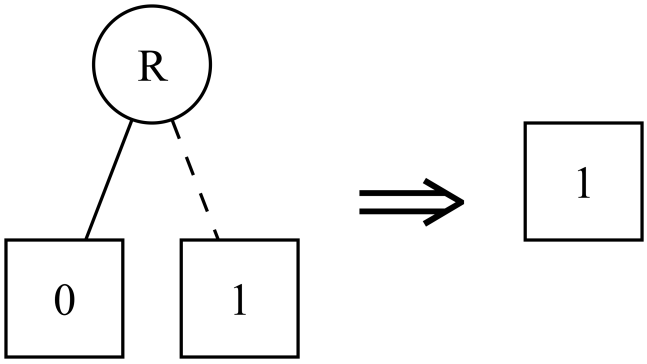


which produces

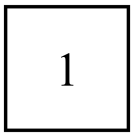


1.4 Case: $Q = 1$

Recursing upwards and following the $Q = 1$ path gives us

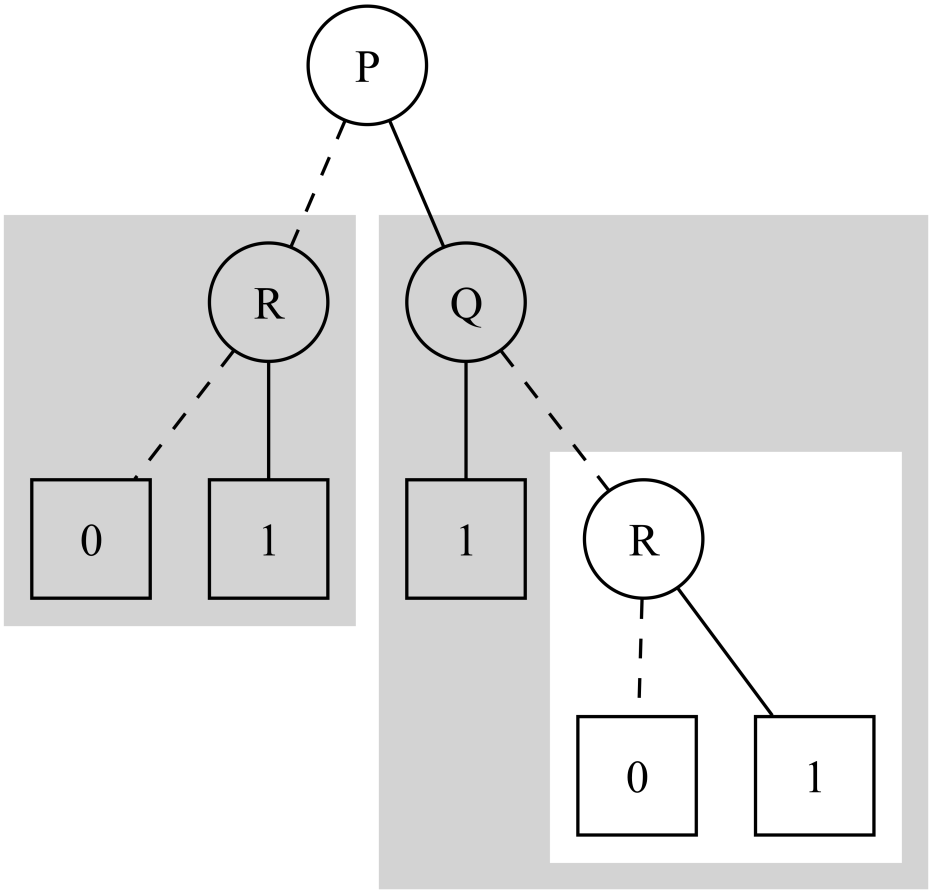


Doing the computation with the $\Rightarrow$ gives us a BDD where both the positive and negative edge of R lead to a 1 (one) leaf. This can be simplified by removing the redundant R node and simply returning



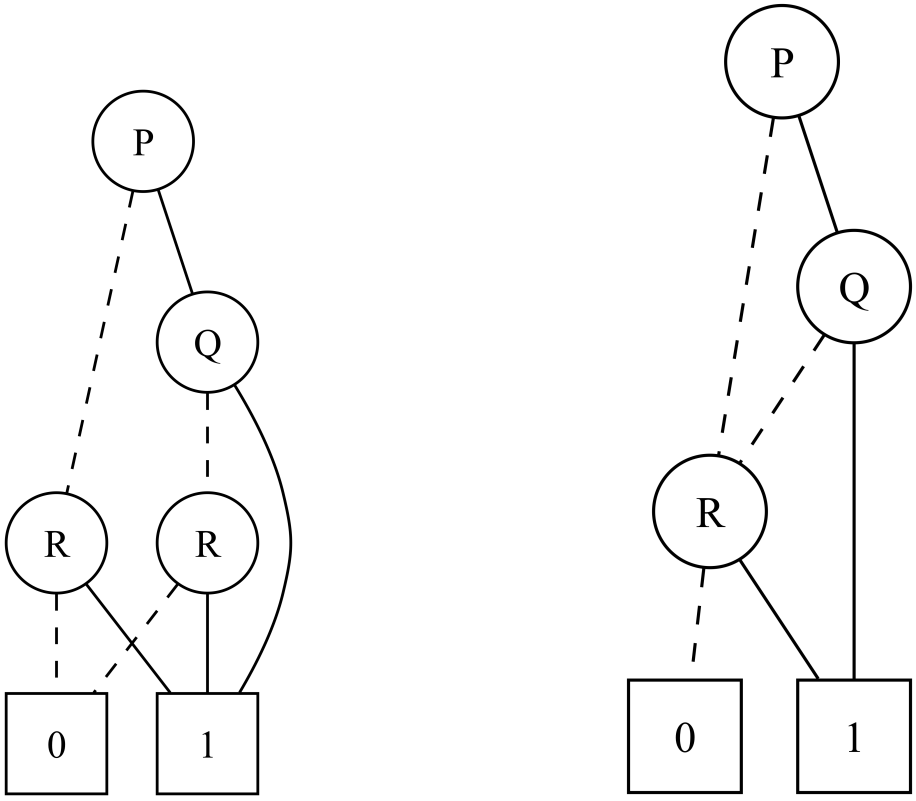
1.5 Putting It Together

Recursing all the way up, we draw on our page the following tree,



We first combine each of the 1 and 0 leaf nodes together, producing the BDD below on the left.

We can then remove duplicated nodes. Notice that both of the R nodes have the exact same output edges. This means that we can combine these into one new node, which takes the input edges of both — in this case from P and R. This gives us the BDD on the right, which is fully reduced.

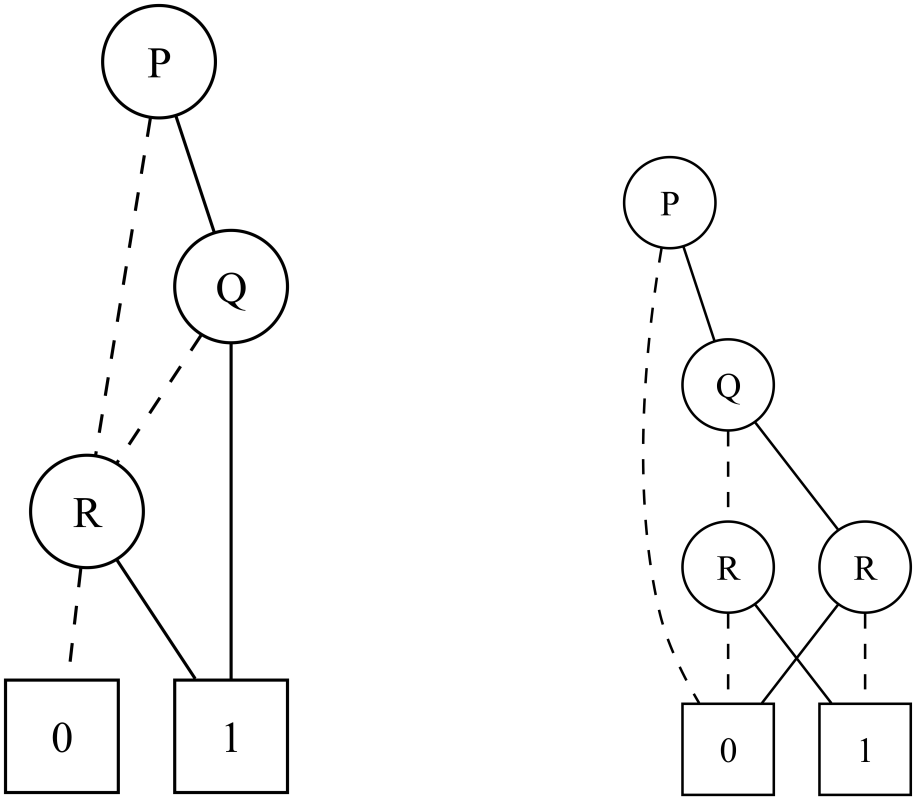


We have computed the Reduced, Ordered, Binary Decision Diagram (ROBDD) of $\neg R \Rightarrow (Q \wedge P)$.

Note that the choice of connective had no impact for most of the execution of the algorithm. It is only at the leaf nodes that we directly computed the implication on boolean values, which is trivial for any operator we choose. Combining two BDDs with the exclusive or ($\oplus$) is exactly as simple as finding their conjunction ($\wedge$).

2 BDD of $(\neg R \Rightarrow (Q \wedge P)) \iff (P \wedge (Q \oplus R))$

We will now combine the BDD we produced above for $\neg R \Rightarrow (Q \wedge P)$ with another BDD for $P \wedge (Q \oplus R)$ by the $\iff$ connective. We repeat the new BDD for the former on the left, and show the BDD for the latter on the right.

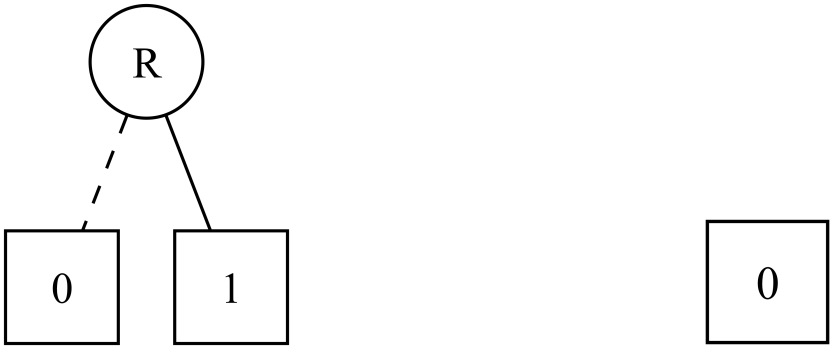


It's easy to feel intimidated by the $\iff$ symbol and try to expand it out into its implication form, but this will make the task far more complex than it needs to be. As we saw above, the choice of connective makes no difference for 90% of the algorithm's execution.

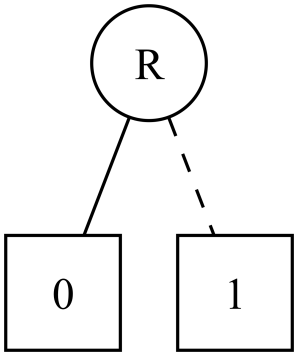
We start by case splitting on the P node. For each of the cases, we have two sub-BDDs gotten by following the choice for each BDD.

2.1 Case: $P = 0$

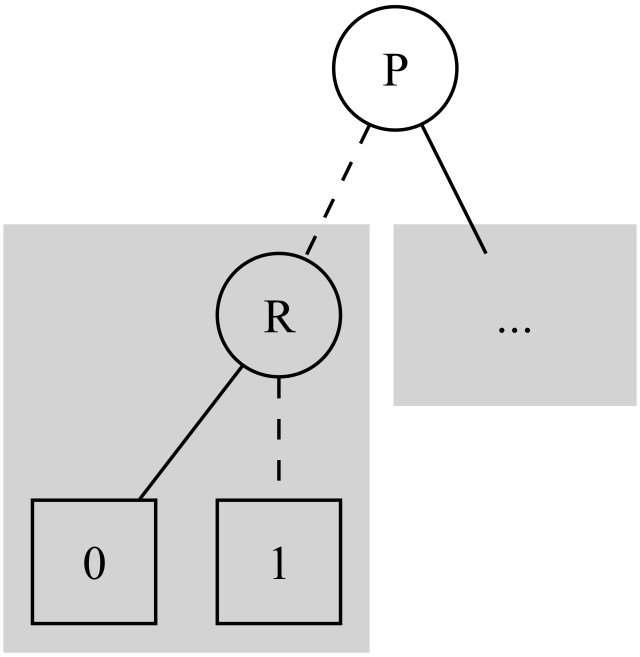
The two sub-BDDs we get by following the $P = 0$ choice for each BDD above are



As we have reached a leaf node, we can apply the $\iff$ to it and each of the leaves of the other sub-BDD, giving us

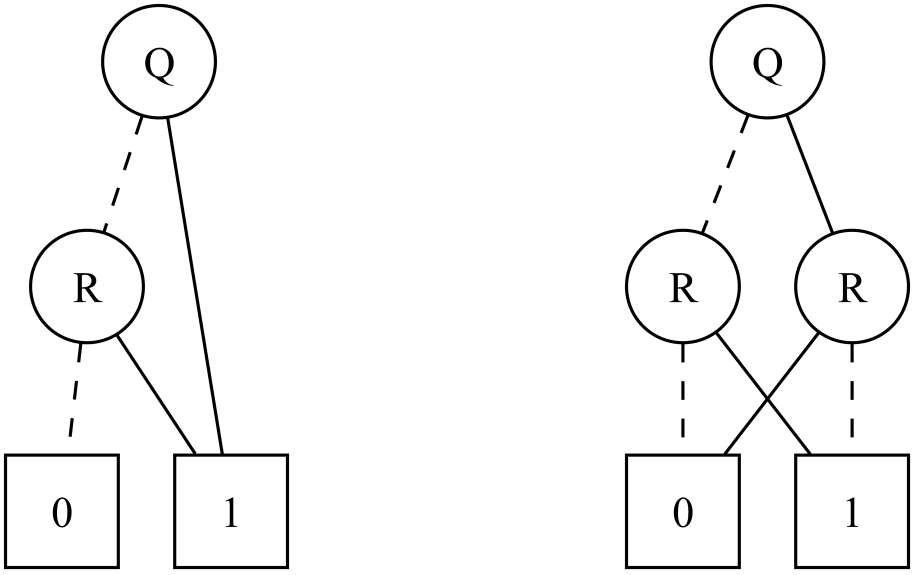


As this case is finished we can attach it to the 0 branch of the P node, and recurse into the $P = 1$ case.



2.2 Case: $P = 1$

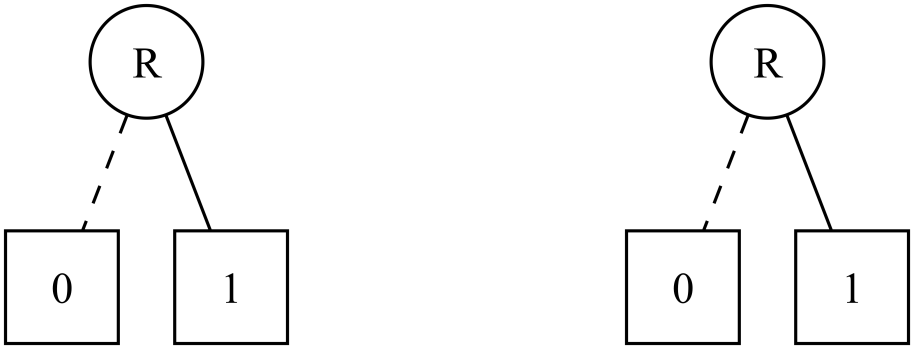
The two sub-BDDs we get by following the $P = 1$ choice for each BDD are



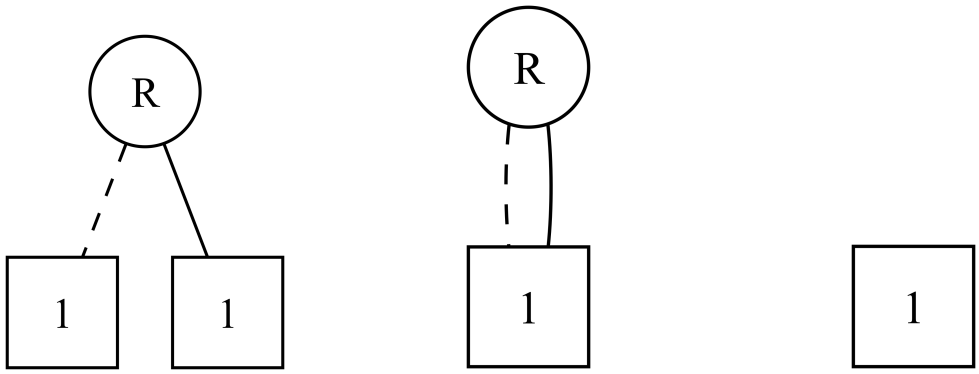
We need to case split on the Q node.

2.3 Case: $Q = 0$

The two sub-BDDs we get by following the $Q = 0$ choice are

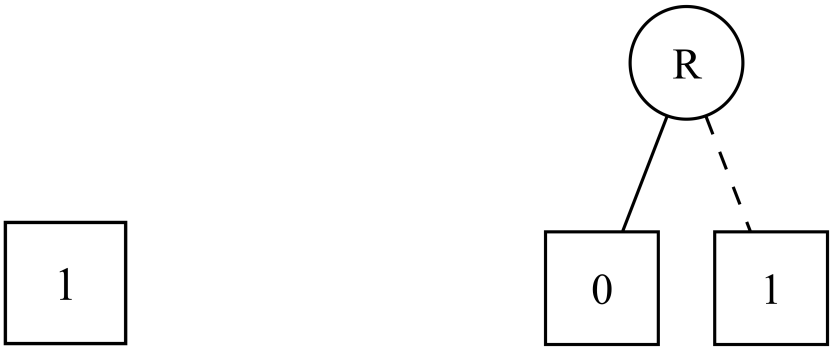


Case splitting on the R node and applying the $\iff$ connective to each leaf, we get a BDD where both branches of R lead to the same outcome. This node is redundant, and the BDD can be simplified into a 1 node as follows.

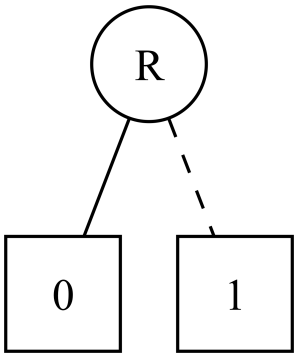


2.4 Case: $Q = 1$

The two sub-BDDs are

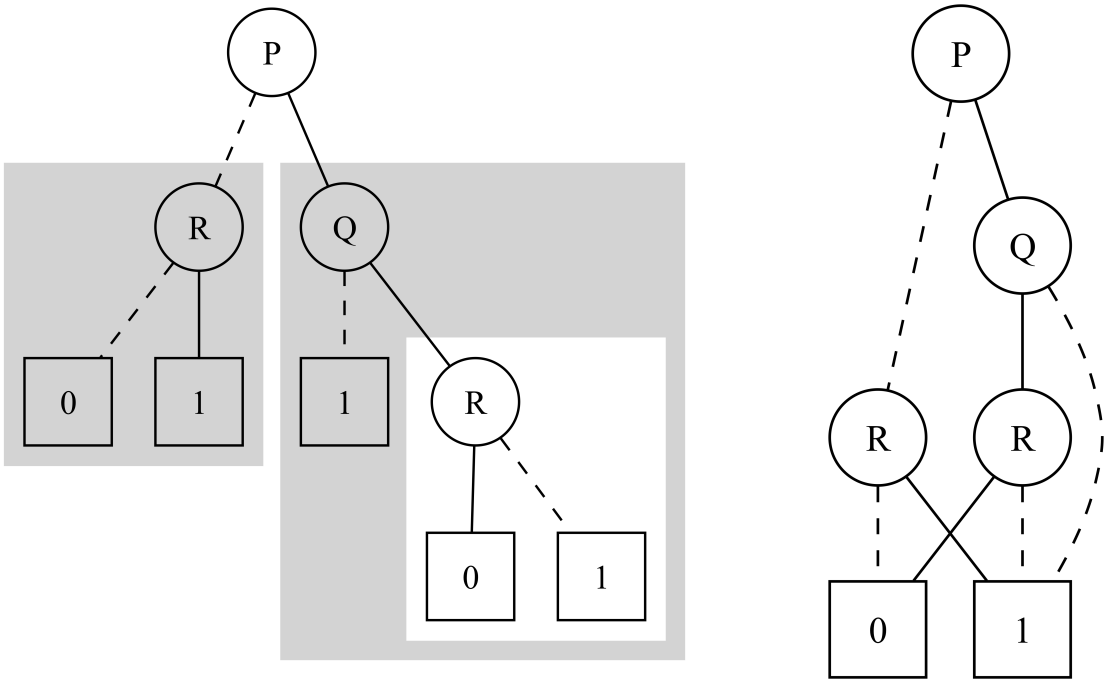


which, combined by $\Longleftrightarrow$, gives us



2.5 Putting It Together

As before, we reattach the new sub-BDDs and then combine duplicate nodes.



This is the BDD of $(\neg R \Rightarrow (Q \wedge P)) \iff (P \wedge (Q \oplus R))$.