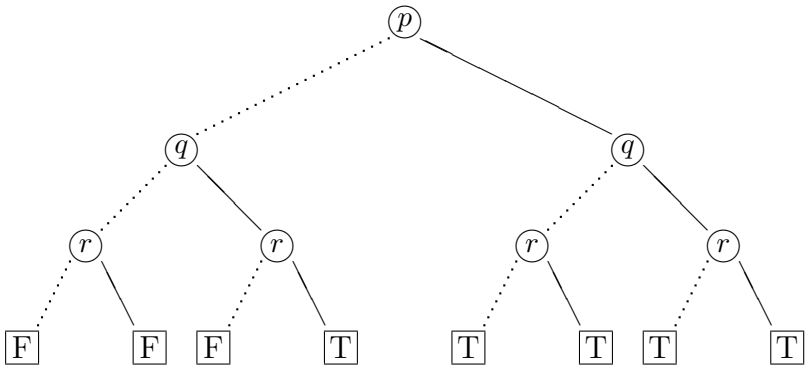
Chapter 5:
Binary Decision Diagrams

- A condensed representation of a truth table, in the form of a directed, acyclic graph.
- The representation dates at least to the 1950s, but went through important developments by Randal Bryant in the 1980s (Carnegie-Mellon University).
- This made possible the first "killer app" of formal verification: model-checking of cache coherency protocols (Ken McMillan – CMU, Cadence).
- State machines with "10^{20} states and beyond" could be verified.

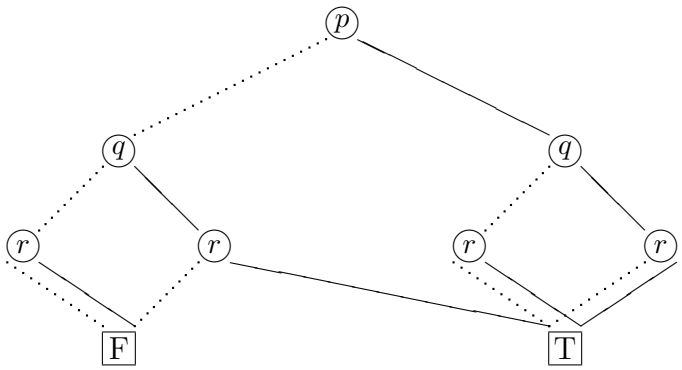
A *binary decision diagram* BDD is a data structure for the representation of a propositional formula. In the worst case, its size is exponential in the number of atoms in the formula. But often, it turns out that formulas of interest have relatively small BDDs.

p	q	r	$p \vee (q \wedge r)$
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

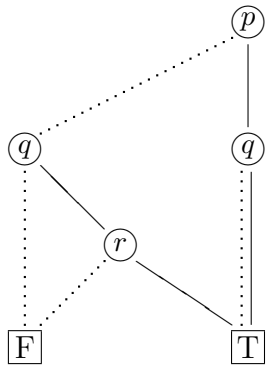
Consider first a *binary decision tree* for the formula:



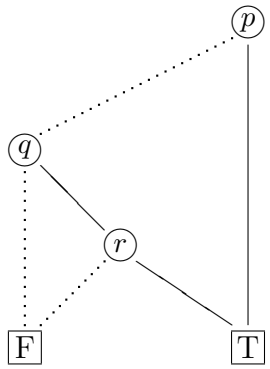
- Here, the dotted lines indicate that the atom labelling the node above has the value F, while the solid lines indicate the value T ...
- and the edges are implicitly directed from above to below.
- Once a leaf is reached, a full interpretation is determined, and the leaf is labelled with the corresponding truth value of the formula.
- Suppose that the tree is *ordered*, in the sense that the atoms appear in the same sequence along each branch.
- By merging nodes (making the tree into a directed, acyclic graph) the representation can be made more compact:
  - identical subgraphs can be merged ...



– . . . and nodes whose children are one and the same can be eliminated:



– These steps are repeated as many times as possible . . .

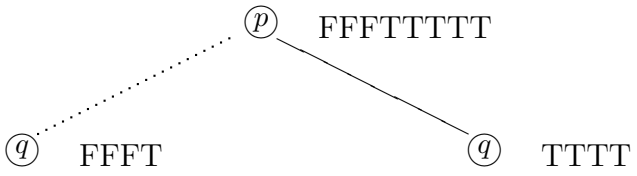


– The result is called a *reduced, ordered BDD*, or ROBDD.

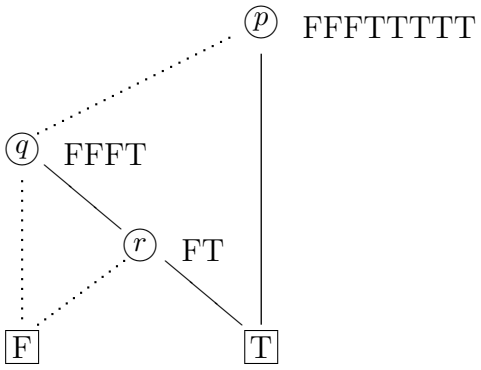
- We shall work exclusively with ROBDDs. This is quite standard; the initials BDD are generally used to stand for ROBDD.

The above example starts with a full binary decision tree, but there are some shortcuts to the BDD representation:

- Consider giving the root node an additional label that is the string of values in the final column of the truth table – or, the sequence of truth values that would label the respective leaves of the full binary decision tree:



- We know then that the leaves that lie beneath the left-hand child of the tree will be labelled by the values in the first half of the string, and those beneath the right hand child by the values in the second half of the string.
- But here we see that all leaves lying below the right-hand child are labelled by T.
- Reduction will result in the merger of those leaves, and the deletion of all internal nodes between the root and those leaves:



- ... and both of the leaves lying beneath the left-hand child of the node labelled q would be labelled F.
- Further merging of leaves labelled with the same truth value yields the ROBDD obtained earlier.

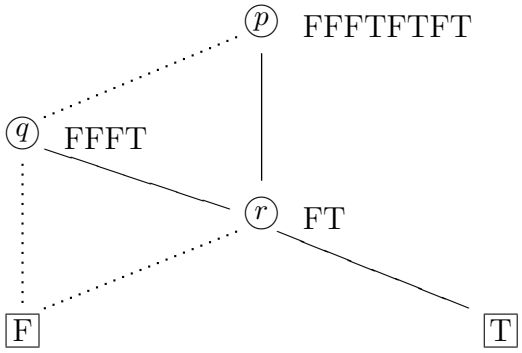
The lengths of the strings labelling internal nodes are always positive powers of two, and therefore even. In the reduced graph, the two halves of such a string can never be the same – otherwise, the subgraphs rooted at the two children would be identical, the children would therefore be merged, and their parent would be deleted:

- If the two halves of such a string are different, the string is called a *bead*.[1] The string of truth values labelling any internal node of a reduced BDD must be a bead.
- The following example illustrates this principle:

p	q	r	$(p \vee q) \wedge r$
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	F
T	T	T	T

- The right-hand child of the root node would be labelled with the string $FTFT$, which is not a bead;
- that means that the graphs rooted at *its* children – both labelled FT – would be identical.
- the children are therefore merged, and their parent deleted;
- and, in this case, the node resulting from the merger is then merged with another node labelled FT.
- We can skip all of those steps:

[1] A convenient, short name chosen by the eminent computer scientist, Donald Knuth.



- We've also skipped some steps on the left-hand side of the graph:
  - The left-hand child of the node labelled q would have been labelled by the string FF, which similarly is not a bead.
  - Its children would have been two leaves labelled F;
  - those children would be merged and their parent deleted, so we have simply linked the node labelled by q directly to a single leaf labelled with F.
  - There's also been another merger of similarly labelled leaves.
- Note that, when nodes are annotated by strings of truth values, all internal nodes of reduced BDDs must be annotated by beads, and
- all nodes must have distinct annotations.

Summary of reduction:

- Identical subgraphs are merged.
- Nodes whose children are merged are deleted (and edges from their parents linked directly to the mergers of their children).
- In particular, if the string of truth values associated with an internal node is not a *bead*, then:
  - its children are merged, and
  - it is deleted.
- These steps are repeated as long as possible.

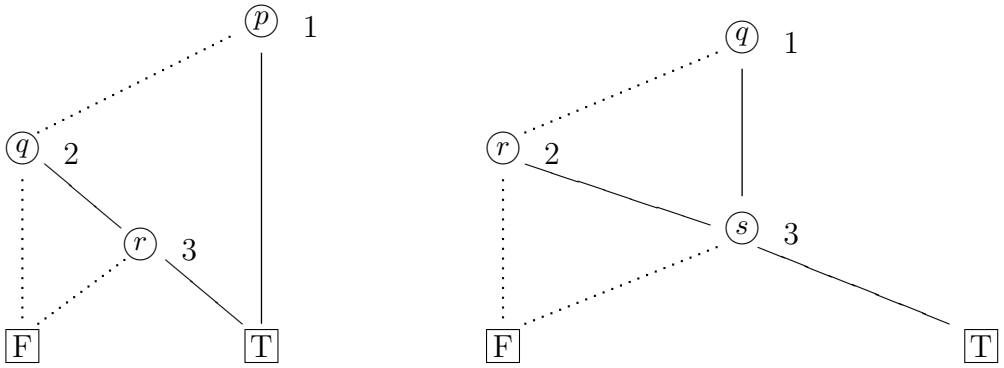
BDDs as state machines

- It is conceptually helpful to think of a BDD as a state machine (specifically, a Moore machine) such as used to design sequential digital systems.
- Think of the truth values of the atoms as being presented sequentially at the input, and of each BDD node as a distinct state.
- The reduction of BDDs can be thought of as an instance of the reduction that is used to reduce state machines.
- As with state machines, the reduction of a BDD is *canonical*: for a given ordering of the atomic propositions, two formulas have the same ROBDD if and only if they are logically equivalent.

Operations on BDDs

- Negation is easy: just swap the labels of F and T leaves.
- The state machine interpretation helps to explain how to apply binary operators directly to BDD representations of formulas.
- Assume an ordering of the set of all atoms occurring in two formulas.
- Suppose that BDD representations of two formulas are given ...
- each obeying an ordering that is consistent with the overall ordering.
- Think of the BDDs as two state machines running concurrently ...
- each changing state whenever the truth value of the atom labelling its state is presented.
- When both states are leaves, the operator is applied to the truth values of the leaves.

Example: Consider the previous two BDDs, with the nodes labelled by different atoms in the case of the second, and with the internal nodes numbered:

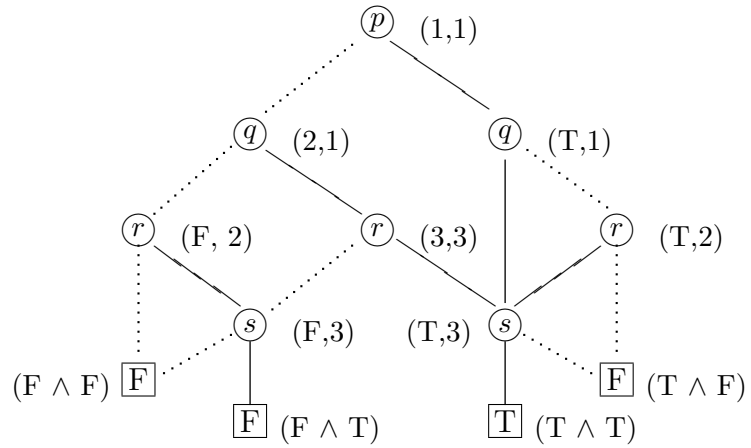


These are both consistent with this ordering of the atoms: p, q, r, s.

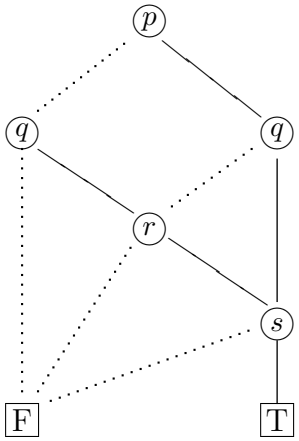
Think of the two BDDs as making concurrent 'state transitions,' as the truth values of the respective atoms are presented to them:

- They start out in the ordered pair of states $(1, 1)$, ready for the left-hand component to respond to the truth value of p.

- If p is F, the left-hand machine goes to state 2; otherwise it goes to T; meanwhile, the right-hand 'machine' stays in state 1.
- When the truth value of q is input, both machines respond, unless the left-hand one is already in state T ...
- and so forth:



Reducing,



– this gives the BDD for a 4-atom formula, whose truth table would contain 16 lines.

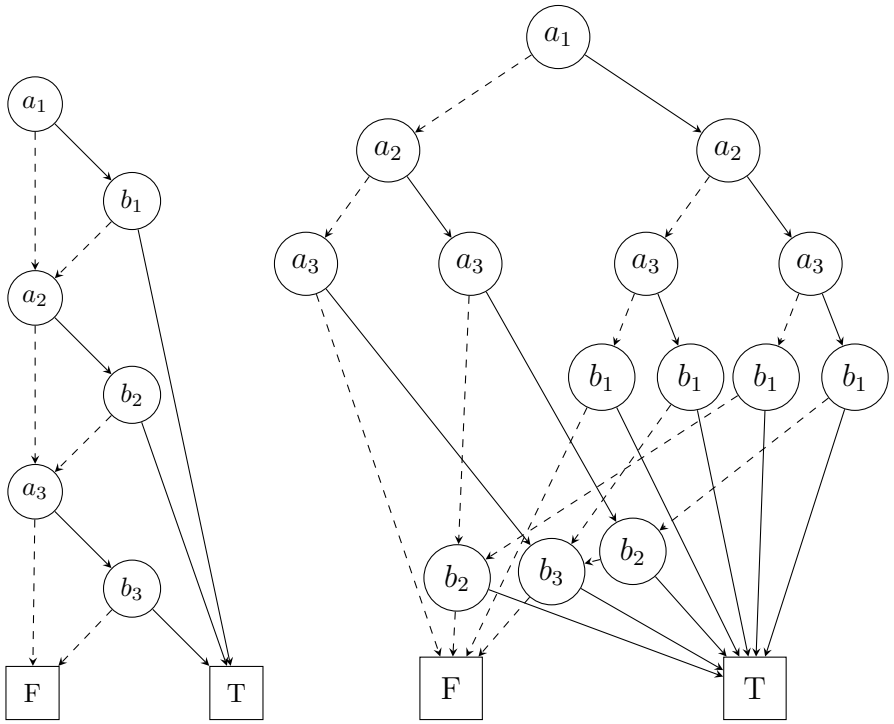
Another method is defined (more formally) in the textbook, but it produces more 'tree-like' BDDs, that require more reduction. That means it takes less advantage of the possibilities of performing operations directly on BDDs. The method described here is based on the operation called "melding" by

Importance of the order of the atoms

The choice of the order of the atoms (commonly called variable ordering in the BDD literature) can make a critical difference to the size of a BDD. For formulas of the form

$$(a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee \ldots \vee (a_n \wedge b_n) \ ,$$

the variable ordering $a_1, b_1, a_2, b_2, \ \ldots, \ a_n, b_n$ leads to BDDs of size linear in n; the variable ordering $a_1, a_2, \ \ldots \ a_n, b_1, b_2, \ldots b_n$ leads to BDDs of exponential size. Here are examples of ROBDDs based on the respective orderings, for the case $n = 3$:



Note that in the second case, the upper part of the BDD is effectively devoted to storing the truth values of all of the a_i atoms, pending the determination of the values of the b_j atoms. That upper part of the diagram contains $2^n - 1$ nodes.

Use in formal verification

The application of BDDs in formal verification is beyond the scope of this course, but a brief discussion can give a general idea of their utility. BDDs can be used to represent the state-transition relation of a digital system 'symbolically,' rather than explicitly, such as in the form of a state machine.

The outputs of a 3-bit binary counter obey the following formulas:

$$\begin{aligned} x'_1 &\iff \neg x_1 \ , \\ x'_2 &\iff x_1 \oplus x_2 \ , \\ x'_3 &\iff (x_1 \wedge x_2) \oplus x_3 \ , \end{aligned}$$

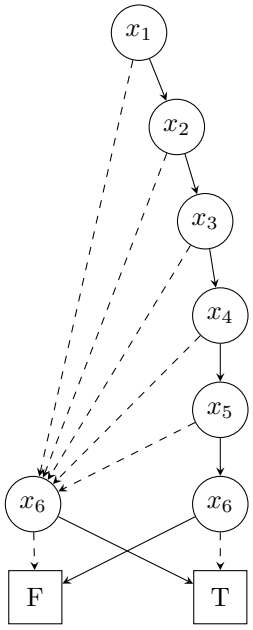
where the primed quantities represent the 'next' values of the variables. In general, we have

$$x'_n \iff (x_1 \wedge x_2 \wedge \ \ldots \ x_{n-1}) \oplus x_n \ ,$$

where, according to the usual convention, the empty conjunction is valid, so that $x'_1 \iff \mathit{true} \oplus x_1$.

Each of these formulas can be represented as a BDD, and the overall state-transition relation of the counter represented as their conjunction. Indeed, if we replace T with 1, and F with 0, the resulting BDD represents the characteristic function of the state-transition relation: this illustrates how BDDs can be used to represent sets and relations, as well as formulas.

This kind of 'symbolic' representation allows the analysis of systems whose explicit state-machine models are too large for efficient computation. The following BDD represents $(x_1 \wedge x_2 \wedge \ \ldots \ x_{n-1}) \oplus x_n$ for the case where $n = 6$, showing that the formulas can be represented as BDDs of linear size, whereas the state-transition diagram for the counter contains 2^n states:



Chapter summary

- Reduced, ordered, binary decision diagrams (ROBDDs, or BDDs) often provide a significantly compressed representation of a truth table.
- Operations on formulas can be performed directly on the compressed representation.
- The ordering of the atomic propositions can be critical in achieving a compact BDD representation.
- BDDs made it possible to perform formal verification of large, industrial systems ...
- but, in the current state of the art, methods based on 'SAT solvers' are more powerful. SAT solvers are the topic of the next chapter.