1.4 Ordered Binary Decision Diagrams

see Chapter 6.1/6.2 of Michael Huth and Mark Ryan: *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge Univ. Press, 2000.

1.5 Normal Forms

We define conjunctions of formulas as follows:

$$\bigwedge_{i=1}^{0} F_i = \top.$$

$$\bigwedge_{i=1}^{1} F_i = F_1.$$

$$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^{n} F_i \wedge F_{n+1}.$$

and analogously disjunctions:

$$\bigvee_{i=1}^{0} F_i = \bot.$$

$$\bigvee_{i=1}^{1} F_i = F_1.$$

$$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^{n} F_i \vee F_{n+1}.$$

Literals and Clauses

A literal is either a propositional variable P or a negated propositional variable $\neg P$.

A clause is a (possibly empty) disjunction of literals.

CNF and DNF

A formula is in conjunctive normal form (CNF, clause normal form), if it is a conjunction of disjunctions of literals (or in other words, a conjunction of clauses).

A formula is in disjunctive normal form (DNF), if it is a disjunction of conjunctions of literals.

Warning: definitions in the literature differ:

are complementary literals permitted?
are duplicated literals permitted?
are empty disjunctions/conjunctions permitted?

CNF and DNF

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy:

A formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals P and $\neg P$.

Conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals P and $\neg P$.

On the other hand, checking the unsatisfiability of CNF formulas or the validity of DNF formulas is known to be coNP-complete.

Conversion to CNF/DNF

Proposition 1.8:
For every formula there is an equivalent formula in CNF (and also an equivalent formula in DNF).

Proof:
We consider the case of CNF.

Apply the following rules as long as possible (modulo associativity and commutativity of $\wedge$ and $\vee$):

Step 1: Eliminate equivalences:

$$(F \leftrightarrow G) \;\Rightarrow_K\; (F \rightarrow G) \wedge (G \rightarrow F)$$

Conversion to CNF/DNF

Step 2: Eliminate implications:

$$(F \rightarrow G) \;\Rightarrow_K\; (\neg F \vee G)$$

Step 3: Push negations downward:

$$\neg(F \vee G) \;\Rightarrow_K\; (\neg F \wedge \neg G)$$

$$\neg(F \wedge G) \;\Rightarrow_K\; (\neg F \vee \neg G)$$

Step 4: Eliminate multiple negations:

$$\neg\neg F \;\Rightarrow_K\; F$$

Conversion to CNF/DNF

Step 5: Push disjunctions downward:

$$(F \land G) \lor H \;\Rightarrow_K\; (F \lor H) \land (G \lor H)$$

Step 6: Eliminate $\top$ and $\bot$:

$$\begin{aligned}
(F \land \top) &\;\Rightarrow_K\; F \\
(F \land \bot) &\;\Rightarrow_K\; \bot \\
(F \lor \top) &\;\Rightarrow_K\; \top \\
(F \lor \bot) &\;\Rightarrow_K\; F \\
\neg\bot &\;\Rightarrow_K\; \top \\
\neg\top &\;\Rightarrow_K\; \bot
\end{aligned}$$

Conversion to CNF/DNF

Proving termination is easy for most of the steps; only step 3 and step 5 are a bit more complicated.

The resulting formula is equivalent to the original one and in CNF.

The conversion of a formula to DNF works in the same way, except that disjunctions have to be pushed downward in step 5.

Complexity

Conversion to CNF (or DNF) may produce a formula whose size is exponential in the size of the original one.

Satisfiability-preserving Transformations

The goal

"find a formula G in CNF such that $\models F \leftrightarrow G$"

is unpractical.

But if we relax the requirement to

"find a formula G in CNF such that $F \models \bot$ iff $G \models \bot$"

we can get an efficient transformation.

Satisfiability-preserving Transformations

Idea: A formula $F[F']$ is satisfiable if and only if $F[P] \wedge (P \leftrightarrow F')$ is satisfiable
(where P is a new propositional variable that works as an abbreviation for F').

We can use this rule recursively for all subformulas in the original formula (this introduces a linear number of new propositional variables).

Conversion of the resulting formula to CNF increases the size only by an additional factor (each formula $P \leftrightarrow F'$ gives rise to at most one application of the distributivity law).

Optimized Transformations

A further improvement is possible by taking the polarity of the subformula F into account.

Assume that F contains neither $\rightarrow$ nor $\leftrightarrow$. A subformula F' of F has positive polarity in F, if it occurs below an even number of negation signs; it has negative polarity in F, if it occurs below an odd number of negation signs.

Optimized Transformations

Proposition 1.9:

Let $F[F']$ be a formula containing neither $\rightarrow$ nor $\leftrightarrow$; let P be a propositional variable not occurring in $F[F']$.

If F' has positive polarity in F, then $F[F']$ is satisfiable if and only if $F[P] \wedge (P \rightarrow F')$ is satisfiable.

If F' has negative polarity in F, then $F[F']$ is satisfiable if and only if $F[P] \wedge (F' \rightarrow P)$ is satisfiable.

Proof:

Exercise.