Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Alexander Krauss, Sascha Böhme

WS 2008/09
24. 10. 2008

Logic

Exercise Sheet 2

Exercise 2.1

a) Find a (linear or quadratic) decision procedure for propositional formulas involving only `Atom` and `Iff`. *Hint: Recall that* `Iff` *is associative and commutative.*

b) Prove that your decision procedure returns true iff the input formula is a tautology.

c) Implement the procedure in OCaml.

Solution

The central idea is that a formua is a tautology iff every atom occurs an even number of times. This is because we can reorder atoms as we like (due to associativity and commutativity) and two occurences of the same atom "cancel out" (due to the law $(A \Leftrightarrow A) \equiv \text{True}$). After cancelling we end up with a formula where every atom occurs at most once, and the order does not matter. The most natural representation for this is a set of atoms.

We call this set $R(p)$, the set of *relevant* atoms, directly by recursion over the structure of the formula p:

$$\begin{aligned} R(x) &= \{x\} \\ R(p_1 \Leftrightarrow p_2) &= R(p_1) \mathbin{\Delta} R(p_2) \qquad \text{where } A \mathbin{\Delta} B = (A \setminus B) \cup (B \setminus A) \end{aligned}$$

Wee need to prove: p is a tautology iff $R(p) = \emptyset$.

However, this statement is not directly provable by induction. We may hit a subformula p' for which $R(p')$ is nonempty. Then, our induction hypothesis is too weak and we are stuck.

We must strengthen the claim to be meaningful for the other cases, too:

Let v be any valuation, and x an atom in p. We write v^T for the valuation $((x \mapsto \text{true})\ v)$ that maps x to true and coincides with v on all other atoms. Similarly, v^F abbreviates $((x \mapsto \text{false})\ v)$.

Lemma 1. *If $x \notin R(p)$, then* `eval` $p\ v^t = $ `eval` $p\ v^f$.
If $x \in R(p)$, then `eval` $p\ v^t = \text{not}$ (`eval` $p\ v^f$).

Proof. By induction on the structure of the formula p.

Case Atom: $p = y$. Then $R(p) = \{y\}$.

If $x \in R(p)$ then $x = y$ and `eval` $p\ v^t = v^t(x) = \text{true} = \text{not}\ (v^f(x)) = \text{not}$ (`eval` $p\ v^f$).
If $x \notin R(y)$ then $x \neq y$ and hence `eval` $p\ v^t = v(y) =$ `eval` $p\ v^f$

Case $\Leftrightarrow$: $p = p_1 \Leftrightarrow p_2$:

If $x \in R(p)$, there are two cases:

a) $x \in R(p_1), x \notin R(p_2)$. Then

$$\begin{aligned}
&\texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^t \\
&= (\texttt{eval}\ p_1\ v^t = \texttt{eval}\ p_2\ v^t) && \text{def. of } \texttt{eval} \\
&= (\text{not}\ (\texttt{eval}\ p_1\ v^f) = \texttt{eval}\ p_2\ v^f) && \text{ind. hyp.} \\
&= \text{not}\ (\texttt{eval}\ p_1\ v^f = \texttt{eval}\ p_2\ v^f) && \text{pull negation out} \\
&= \text{not}\ (\texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^f) && \text{def. of } \texttt{eval}
\end{aligned}$$

b) $x \notin R(p_1), x \in R(p_2)$: Symmetric.

If $x \notin R(p)$, there are again two cases:

c) $x \notin R(p_1), x \notin R(p_2)$. Then

$$\begin{aligned}
&\texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^t \\
&= (\texttt{eval}\ p_1\ v^t = \texttt{eval}\ p_2\ v^t) \\
&= (\texttt{eval}\ p_1\ v^f = \texttt{eval}\ p_2\ v^f) \\
&= \texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^f
\end{aligned}$$

d) $x \in R(p_1), x \in R(p_2)$. Then

$$\begin{aligned}
&\texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^t \\
&= (\texttt{eval}\ p_1\ v^t = \texttt{eval}\ p_2\ v^t) \\
&= (\text{not}\ (\texttt{eval}\ p_1\ v^f) = \text{not}\ (\texttt{eval}\ p_2\ v^f)) \\
&= (\texttt{eval}\ p_1\ v^f = \texttt{eval}\ p_2\ v^f) \\
&= \texttt{eval}\ (p_1 \Leftrightarrow p_2)\ v^f
\end{aligned}$$

□

From the above property we now conclude that if $R(p) = \emptyset$ then $\texttt{eval}\ p\ v$ is independent of the valuation v. This means that p is either a tautology or unsatisfiable. But it is easy to show that if p consists only of atoms and $\Leftrightarrow$, the valuation $\hat{v}$ that maps all atoms to true always satisfies the formula:

Proof. By induction on the structure of p:

Case Atom: $p = y$. Then $\texttt{eval}\ p\ \hat{v} = \hat{v}(y) = \text{true}$.

Case $\Leftrightarrow$: $p = p_1 \Leftrightarrow p_2$. Then

$$\begin{aligned}
&\texttt{eval}\ (p_1 \Leftrightarrow p_2)\ \hat{v} \\
&= (\texttt{eval}\ p_1\ \hat{v} = \texttt{eval}\ p_2\ \hat{v}) \\
&= (True = True) \\
&= True
\end{aligned}$$

□

Thus, we have proved the correctness of the following tautology checker, which has a very simple implementation:

```
let symdiff xs ys = union (subtract xs ys) (subtract ys xs);;

let rec relevant = (function
    Atom p -> [p]
  | Iff (p, q) -> symdiff (relevant p) (relevant q));;

let taut frm = (relevant frm = []);;
```

Exercise 2.2 Functional completeness

A set A of logical connectives is called *functionally complete* iff every boolean-valued function can be expressed as a formula with just connectives from A.

a) Show that $\{\wedge, \vee, \neg\}$ is functionally complete.

b) Show that $\{\wedge, \vee\}$ is not functionally complete.

c) Show that $\{\overline{\wedge}\}$ is functionally complete, where $p \,\overline{\wedge}\, q$ is interpreted as $\neg(p \wedge q)$

Solution

a) Let f be any boolean-valued function in n arguments, given by a truth table. Then we can express f by the a DNF formula with just $\wedge$, $\vee$, and $\neg$. For each valuation that makes the function true, we add a conjunction of literals expressing this valuation. For example, the binary function that is true iff and only exactly one of its inputs is true can be expressed by the formula $(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$. Thus, $\wedge$, $\vee$, and $\neg$ are sufficient for describing all functions.

b) By induction, we easily show that if a formula p consists only of $\wedge$ and $\vee$, then `eval` $p\ \hat{v} =$ true for the valuation that maps every variable to true. Hence, the function that is always false cannot be expressed.

c) Since we have already shown that $\{\wedge, \vee, \neg\}$ are functionally complete, it suffices to express them in terms of $\overline{\wedge}$:

$$\begin{aligned} \neg p &\equiv (p \overline{\wedge} p) \\ p \wedge q &\equiv \neg(p \overline{\wedge} q) \\ p \vee q &\equiv \neg p \overline{\wedge} \neg q \end{aligned}$$