Binary Decision Diagrams

Boolean functions (in finitely many variables) can be represented by so-called binary decision diagrams (BDDs), which are given by the following data type:

\[
\text{datatype bdd} = \text{Leaf bool} \mid \text{Branch bdd bdd}
\]

A constructor \text{Branch b1 b2} that is \(i\) steps away from the root of the tree corresponds to a case distinction based on the value of the variable \(v_i\). If the value of \(v_i\) is \text{False}, the left subtree \(b1\) is evaluated, otherwise the right subtree \(b2\) is evaluated. The following figure shows a Boolean function and the corresponding BDD.

<table>
<thead>
<tr>
<th>(v_0)</th>
<th>(v_1)</th>
<th>(v_2)</th>
<th>(f(v_0, v_1, v_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>*</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>*</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>*</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

**Exercise 1:** Define a function

\[
\text{consts eval :: } (\text{nat} \Rightarrow \text{bool}) \Rightarrow \text{nat} \Rightarrow \text{bdd} \Rightarrow \text{bool}
\]

that evaluates a BDD under a given variable assignment, beginning at a variable with a given index.

**Exercise 2:** Define two functions

\[
\text{consts}
\begin{align*}
\text{bdd_unop} :& "(\text{bool} \Rightarrow \text{bool}) \Rightarrow \text{bdd} \Rightarrow \text{bdd}" \\
\text{bdd_binop} :& "(\text{bool} \Rightarrow \text{bool} \Rightarrow \text{bool}) \Rightarrow \text{bdd} \Rightarrow \text{bdd} \Rightarrow \text{bdd}
\end{align*}
\]

for the application of unary and binary operators to BDDs, and prove their correctness.

Now use \text{bdd_unop} and \text{bdd_binop} to define

\[
\text{consts}
\begin{align*}
\text{bdd_and} :& "\text{bdd} \Rightarrow \text{bdd} \Rightarrow \text{bdd}"
\end{align*}
\]
and show correctness.

Finally, define a function

```plaintext
consts bdd_var :: "nat ⇒ bdd"
```

to create a BDD that evaluates to \textit{True} if and only if the variable with the given index evaluates to \textit{True}. Again prove a suitable correctness theorem.

\textbf{Hint:} If a lemma cannot be proven by induction because in the inductive step a different value is used for a (non-induction) variable than in the induction hypothesis, it may be necessary to strengthen the lemma by universal quantification over that variable (cf. Section 3.2 in the Tutorial on Isabelle/HOL).

\textbf{Example:} instead of

```
lemma "P (b::bdd) x"
apply (induct b)
```

Strengthening:

```
lemma "∀ x. P (b::bdd) x"
apply (induct b)
```

\textbf{Exercise 3:} Recall the following data type of propositional formulae (cf. the exercise on “Representation of Propositional Formulae by Polynomials”)

```plaintext
datatype form = T | Var nat | And form form | Xor form form
```

together with the evaluation function \textit{evalf}:

```plaintext
definition xor :: "bool ⇒ bool ⇒ bool" where
  "xor x y ≡ (x ∧ ¬ y) ∨ (¬ x ∧ y)"
```

```plaintext
primrec evalf :: "(nat ⇒ bool) ⇒ form ⇒ bool" where
  "evalf e T = True"
| "evalf e (Var i) = e i"
| "evalf e (And f1 f2) = (evalf e f1 ∧ evalf e f2)"
| "evalf e (Xor f1 f2) = xor (evalf e f1) (evalf e f2)"
```

Define a function

```plaintext
consts mk_bdd :: "form ⇒ bdd"
```

that transforms a propositional formula of type \textit{form} into a BDD. Prove the correctness theorem

```plaintext
theorem mk_bdd_correct: "eval e 0 (mk_bdd f) = evalf e f"
```