Lecture 13: Binary Decision Diagrams in Detail (cont.)

• Quantified Boolean Formulas

• Representing Sets and Relations as OBDDs
OBDDs are extremely useful for obtaining concise representations of relations over finite domains.

Representing Finite Relations
$R$ can now be represented as the OBDD for the characteristic function of $R'$.

where $x^i$ is a vector of $m$ boolean variables which encodes the variable $x_i^j$ that takes values in $D_i^j$.

We construct a new boolean relation $R'$ of arity $u \times m$ according to the following rule:

$((u \phi)x^1 \cdots (1 \phi)x^m)R = (u \phi \cdots 1 \phi)x^0 \cdots x^m$.

To represent $R'$ as an OBDD, we encode elements of $D_i^j$ using a bijection of $\phi$, that maps each boolean vector of length $m$ to an element of $D_i^j$.

$D_i^j \leftarrow \{1, \ldots, 0\}^m : \phi \mapsto \phi$.

If $R'$ is an $n$-ary relation over the domain $D$, where $D$ has $\leq m$ elements for some $m > 1$.
This technique can be easily extended to relations over different domains, $D_1, \ldots, D_n$.

Since sets can be viewed as unary relations, the same technique can be used to represent sets as OBDDs.
Quantified Boolean Formulas (QBF)

In order to construct complex relations it is convenient to permit quantification over boolean variables.

\[
\begin{align*}
\text{Let } \{a_1, \ldots, a_n\} &= \Lambda, \\
\text{is the smallest set of formulas such that } &
\end{align*}
\]

\[
\begin{align*}
\text{if } f \text{ is a formula and } \Lambda \in \nu \text{ then } &\wedge f \text{ and } f \wedge \nu \text{ are formulas, and } \\
\text{if } f \text{ and } \nu \text{ are formulas, then } &f \land \nu \text{ and } f \land \nu \text{ are formulas, and } \\
\text{every variable in } \Lambda \text{ is a formula,}
\end{align*}
\]

The resulting logic is called QBF (Quantified Boolean Formulas).
If $f$ is a formula in $\mathcal{L}$ and $\varphi$ is a truth assignment, we will use the notation $\langle \varphi \rangle f = \varphi$ for the truth assignment defined by

$$m = \varphi \quad \text{otherwise},$$

$$\{ \langle m \rangle \varphi \} = \{ \langle m \rangle (p \to q) \varphi \}$$

If $a \in \{0, 1\}$, then we will use the notation $\langle a \rangle \varphi$ for the truth assignment defined by

$$\{ \langle 0 \rangle \varphi \} \leftarrow \mathcal{L} : \langle \varphi \rangle$$

**Truth Assignments**
The relation $\models$ is defined recursively in the obvious manner.
We will identify each formula with the boolean relation that it determines.

Every QBF formula determines an n-ary boolean relation consisting of those truth assignments for the variables in \( \Lambda \) that make the formula true.

QBF formulas have the same expressive power as ordinary propositional formulas, however, they are sometimes much more concise.

QBF formulas and Relations
Previously, we showed how to associate an OBDD with each formula of propositional logic. In practice, however, special algorithms are needed to handle quantifiers efficiently.

\[
\begin{align*}
1 \rightarrow x & \quad f \cdot 0 \rightarrow x \quad f = f x A \\
1 \rightarrow x & \quad f + 0 \rightarrow x \quad f = f x E
\end{align*}
\]

In principle, it is easy to construct OBDDs for \( f \) when \( f \) is given as an OBDD.

QBF formulas and Relations
In model checking, quantifiers occur most frequently in relational products. We give an algorithm that performs this computation in one pass over the OBDDs, and we avoid constructing the OBDD for (a)φ ∨ (a)ψ. This is important since we avoid constructing the OBDD for (a)ψ.

\[ [(a)φ ∨ (a)ψ] aE \]
end

return

insert \( \cdot \), \( b, \), \( f \) (y, \( \cdot \), \( A, \), \( z \)) in cache

end

*/ (0y \lor z \mbox{\text{\textendash}}} \land (1y \lor z) */

(0y, 1y, z) \{ If the \{B \} is \{E \}

else

*/ 1y \land 0y */

(1y, 0y) \{ O =: y \in \{F \} \{G \} \{H \} \{I \}

(\{G \} ; \{H \} ; \{I \} ; \{f \}) pos \{P \} =: \{1y \}

(\{G \} , \{H \} , \{I \} , \{f \}) pos \{P \} =: \{0y \}


let \{z \} be the bottom of \{x \} and \{h \}

let \{h \} be the top variable of \{b \}

else let \{z \} be the top variable of \{b \}

else if \{y \} is cached then return \{y \}

else if \{y \}, \{b \}, \{f \} then return true

else if \{y \}, \{false \} then return false

else \{y \}, \{false \} then return \{false \}

end

RelProd(\{G\}, \{H\}, \{I\}, \{B\}, \{E\}, \{O\}BDD, \{E\} : \{select\}variables)
Although the algorithm works well in practice, it has exponential complexity in the worst case.

Like many OBDD algorithms, \texttt{RelProd} uses a result cache.

Relational Products (Cont.)