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

- Representing Sets and Relations as OBDDs
- Quantified Boolean Formulas

Representing Finite Relations

OBDDs are extremely useful for obtaining concise representations of relations over finite domains.

If R is n-ary relation over $\{0, 1\}$ then R can be represented by the OBDD for its *characteristic function*

$$f_R(x_1, \ldots, x_n) = 1 \text{ iff } R(x_1, \ldots, x_n).$$

- J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
- J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 13(4):401–424, 1994.

Representing Relations (Cont.)

If R is an n-ary relation over the domain D, where D has 2^m elements for some $m > 1$.

To represent R as an OBDD, we encode elements of D, using a bijection $\phi : \{0,1\}^m \rightarrow D$ that maps each boolean vector of length m to an element of D.

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

$$R'(\bar{x}_1, \ldots , \bar{x}_n) = R(\phi(\bar{x}_1), \ldots , \phi(\bar{x}_n))$$

where $\bar{x}_i$ is a vector of m boolean variables which encodes the variable x_i that takes values in D.

R can now be represented as the OBDD for the characteristic function $f_{R'}$ of R'.

Representing Relations (Cont.)

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.

The resulting logic is called QBF (*Quantified Boolean Formulas*).

Let $V = \{v_1, \ldots, v_n\}$ be a set of propositional variables.

$QBF(V)$ is the smallest set of formulas such that

- every variable in V is a formula,
- if f and g are formulas, then $\neg f$, $f \vee g$, and $f \wedge g$ are formulas, and
- if f is a formula and $v \in V$, then $\exists v f$ and $\forall v f$ are formulas.

Truth Assignments

A *truth assignment* for $QBF(V)$ is a function $\sigma : V \to \{0, 1\}$.

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

$$\sigma\langle v \leftarrow a\rangle(w) = \begin{cases} a & \text{if } v = w \\ \sigma(w) & \text{otherwise.} \end{cases}$$

If f is a formula in $QBF(V)$ and σ is a truth assignment, we will write $\sigma \models f$ when f is true under the assignment σ.

QBF Semantics

The relation $\models$ is defined recursively in the obvious manner:

- $\sigma \models v$ iff $\sigma(v) = 1$,
- $\sigma \models \neg f$ iff $\sigma \not\models f$,
- $\sigma \models f \vee g$ iff $\sigma \models f$ or $\sigma \models g$,
- $\sigma \models f \wedge g$ iff $\sigma \models f$ and $\sigma \models g$,
- $\sigma \models \exists v f$ iff $\sigma\langle v \leftarrow 0\rangle \models f$ or $\sigma\langle v \leftarrow 1\rangle \models f$, and
- $\sigma \models \forall v f$ iff $\sigma\langle v \leftarrow 0\rangle \models f$ and $\sigma\langle v \leftarrow 1\rangle \models f$,

QBF formulas and Relations

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

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

We will identify each formula with the boolean relation that it determines.

QBF formulas and Relations

Previously, we showed how to associate an OBDD with each formula of propositional logic.

In principle, it is easy to construct OBDDs for $\exists v\, f$ and $\forall v\, f$ when f is given as an OBDD.

- $\exists x f = f\ |_{x \leftarrow 0} + f\ |_{x \leftarrow 1}$
- $\forall x f = f\ |_{x \leftarrow 0} \cdot f\ |_{x \leftarrow 1}$

In practice, however, special algorithms are needed to handle quantifiers efficiently.

Relational Products

In model checking, quantifiers occur most frequently in *relational products*

$$\exists \bar{v}\left[f(\bar{v}) \wedge g(\bar{v})\right].$$

We give an algorithm that performs this computation in one pass over the OBDDs $f(\bar{v})$ and $g(\bar{v})$.

This is important since we avoid constructing the OBDD for $f(\bar{v}) \wedge g(\bar{v})$.

Relational Products (Cont.)

$RelProd(f, g\colon OBDD,\ E\colon setofvariables)$
if $f = \mathsf{false} \vee g = \mathsf{false}$ then return false
else if $f = \mathsf{true} \wedge g = \mathsf{true}$ then return true
else if (f, g, E, h) is cached then return h
else let x be the top variable of f
let y be the top variable of g
let z be the topmost of x and y
$h_0 := RelProd(f|_{z=0}, g|_{z=0}, E)$
$h_1 := RelProd(f|_{z=1}, g|_{z=1}, E)$
if $z \in E$ then $h := Or(h_0, h_1)$
/$*$ OBDD for $h_0 \vee h_1$ $*$/
else $h := IfThenElse(z, h_1, h_0)$
/$*$ OBDD for $(z \wedge h_1) \vee (\neg z \wedge h_0)$ $*$/
endif
insert (f, g, E, h) in cache
return h
endif

Relational Products (Cont.)

Like many OBDD algorithms, *RelProd* uses a result cache.

In this case, entries in the cache have the form (f, g, E, h), where E is a set of variables that are quantified out and f, g and h are OBDDs.

If such an entry is in the cache, then a previous call to *RelProd*(f, g, E) returned h as its result.

Although the algorithm works well in practice, it has exponential complexity in the worst case.