Overview

- Ordered Binary Decision Diagrams
- Kripke structures as OBDDs
- CTL and OBDDs
- Summary

Binary Decision Diagrams

Functions $f : \{0,1\}^n \rightarrow \{0,1\}$ can be represented by binary decision diagrams. In order to make the representation canonical two things have to be achieved;

✔ the variables must be ordered,

✔ and redundancy must be removed.

Ordered Binary Decision Diagrams

The redundancy can be removed in time linear to the size of the binary decision tree by repeatedly applying three transformations rules.

✔ Remove duplicate terminals

✔ Remove duplicate nonterminals

✔ Remove redundant tests

The size of the resulting diagram is dependent on the variable ordering. Deciding if an ordering is optimal is NP-complete.

Logical operations on OBDDs

The $Apply$ function performs logical operations on boolean formulae represented as OBDDs.

- v and v' are the roots of the OBDDs for f and f'
- $x = var(v)$ and $x' = var(v')$

We use the $Apply$ function to compute an arbitrary two argument logical operation denoted $*$

$$f|_{x_i \leftarrow b}(x_1, ..., x_n) = f(x_1, ..., x_{i-1}, b, ..., x_n)$$

The Apply function

- If v and v' are both terminal vertices, then $f * f' = value(v) * value(v')$
- If $x = x'$, then $f * f' = (\neg x \wedge (f|_{x \leftarrow 0} * f'|_{x \leftarrow 0})) \vee (x \wedge (f|_{x \leftarrow 1} * f'|_{x \leftarrow 1}))$
- If $x < x'$, then $f'|_{x \leftarrow 0} = f'|_{x \leftarrow 1} = f'$ and thus $f * f = (\neg x \wedge (f|_{x \leftarrow 0} * f')) \vee (x \wedge (f|_{x \leftarrow 1} * f'))$
- The case when $x' < x$ is analogous to the case above

Exponential complexity can be avoided using dynamic programming.

Kripke structures as OBDDs

OBDDs represent boolean formulae. An *n*-ary relation Q over $\{0,1\}$ can be represented by its characteristic function $f_Q(x_1, ..., x_n) = 1$ iff $Q(x_1, ..., x_n)$.

An *n*-ary relation Q over the finite domain D can be represented using a bijection $\phi : \{0,1\} \to D$. The *n*m*-ary relation Q' over $\{0,1\}$ can be represented as an OBDD. $Q'(\overline{x}_1, ..., \overline{x}_n) = Q(\phi(\overline{x}_1), ..., \phi(\overline{x}_n))$ where $\overline{x}_i$ is a vector of *m* boolean variables.

Now the representation of a Kripke structure $M = (S, R, L)$ as an OBDD is straight forward. The atomic proposition *a* is mapped to the set $\{s | a \in L(s)\}$.

CTL Model Checking with OBDDs

OBDDs represent sets of states and transitions, so we can not operate on individual states and transitions. Therefore we use a fixpoint calculation that operates on sets.

The set $P(S)$ represents the set of subsets of S. $P(S)$ forms a lattice under the set inclusion ordering. The least element - the empty set - is denoted by F and the greatest element - the set S - is denoted by T.

A monotonic function $\tau : P(S) \to P(S)$ has a least fixpoint, *lfp* $Z[\tau(Z)] = \bigcap\{Z | \tau(Z) = Z\}$ and a greatest fixpoint *gfp* $Z[\tau(Z)] = \bigcup\{Z | \tau(Z) = Z\}$ when τ is monotonic. When τ is $\bigcup$-continuous, *lfp* $Z[\tau(Z)] = \bigcup_i \tau^i(F)$ and when τ is $\cap$-continuous, *gfp* $Z[\tau(Z)] = \bigcap_i \tau^i(T)$.

CTL operators as fixpoints of predicate transformers

$AF\ f_1 = \text{lfp} Z[f_1 \vee AX\ Z]$

$EF\ f_1 = \text{lfp} Z[f_1 \vee EX\ Z]$

$AG\ f_1 = \text{gfp} Z[f_1 \wedge AX\ Z]$

$EG\ f_1 = \text{gfp} Z[f_1 \wedge EX\ Z]$

$A[f_1 U f_2] = \text{lfp} Z[f_2 \vee (f_1 \wedge AX\ Z)]$

$E[f_1 U f_2] = \text{lfp} Z[f_2 \vee (f_1 \wedge EX\ Z)]$

$A[f_1 R f_2] = \text{gfp} Z[f_2 \wedge (f_1 \vee AX\ Z)]$

$E[f_1 R f_2] = \text{gfp} Z[f_2 \wedge (f_1 \vee EX\ Z)]$

Lemmas

A few lemmas for proving $EG\, f_1 = \text{gfp}Z[f_1 \wedge EX\ Z]$ and $E[f_1 U f_2] = \text{lfp}Z[f_2 \vee (f_1 \wedge EX\ Z)]$: (included for reference, see textbook for details)

- $\tau(Z) = f_1 \wedge EX\ Z$ is monotonic.
- Let $\tau(Z) = f_1 \wedge EX\ Z$ and let $\tau^{i_0}(T)$ be the limit of the sequence $T \supseteq \tau(T) \supseteq \ldots$ For every $s \in S$, if $s \in \tau^{i_0}(T)$ then $s \models f_1$, and there is a state s'such that $(s, s') \in R$ and $s' \in \tau^{i_0}(T)$.
- $EG\ f_1$ is a fixpoint of the function $\tau(Z) = f_1 \wedge EX\ Z$.
- $EG\ f_1$ is the greatest fixpoint of the function $\tau(Z) = f_1 \wedge EX\ Z$.
- $E[f_1 U f_2]$ is the least fixpoint of the function $\tau(Z) = f_2 \vee (f_1 \wedge EX\ Z)$.

A Model Checking Algorithm

The procedure *Check* takes a CTL formula and a system as arguments and returns exactly the states of the system that satisfy the formula encoded as an OBDD. The parameter representing the system is implicit in the following.

- If f is an atomic proposition a, then $Check(f)$ is the OBDD representing a
- If f is the result of a logical operation, the $Apply$ function is used
- $Check(EX\ f) = CheckEX(Check(f)) = \exists \overline{v}'[Check(f(\overline{v}')) \wedge R(\overline{v}, \overline{v}')]$
- $Check(E[fUg]) = \text{lfp}Z(\overline{v})[Check(g(\overline{v})) \vee Check((f(\overline{v})) \wedge CheckEX(Z(\overline{v})))]$
- $Check(EG\ f) = \text{gfp}Z(\overline{v})[Check(f(\overline{v})) \wedge CheckEX(Z(\overline{v}))]$

Summary

- ✔ OBDDs are an efficient representation of logic formulae.
- ✔ OBDDs can be used to represent relations over arbitrary finite domains, and thus Kripke structures can be represented.
- ✔ CTL operators can be implemented as fixpoints.
- ✔ Symbolic model checking uses a representation on a higher level of abstraction, thus enabling efficient exploration of some cases where the model becomes too large when handled explicitly.