

Planning and Optimization

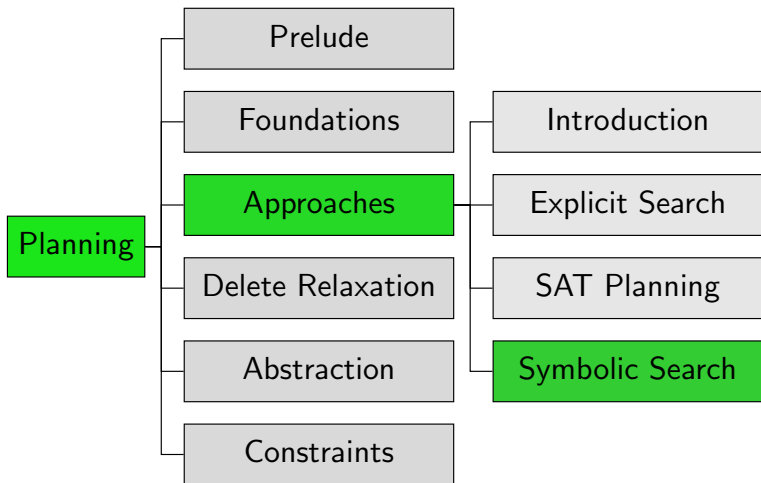
C7. Symbolic Search: Binary Decision Diagrams

Malte Helmert and Gabriele Röger

Universität Basel

October 15, 2025

Content of the Course



Motivation

Symbolic Search Planning: Basic Ideas

- come up with a good **data structure** for **sets of states**
- **hope**: (at least some) exponentially large state sets can be represented as polynomial-size data structures
- simulate a standard search algorithm like **breadth-first search** using these set representations

Symbolic Breadth-First Progression Search

Symbolic Breadth-First Progression Search

```
def bfs-progression( $V, I, O, \gamma$ ):  
     $goal\_states := models(\gamma)$   
     $reached_0 := \{I\}$   
     $i := 0$   
    loop:  
        if  $reached_i \cap goal\_states \neq \emptyset$ :  
            return solution found  
         $reached_{i+1} := reached_i \cup apply(reached_i, O)$   
        if  $reached_{i+1} = reached_i$ :  
            return no solution exists  
         $i := i + 1$ 
```

↪ If we can implement operations *models*, $\{I\}$, $\cap$, $\neq \emptyset$, $\cup$, *apply* and $=$ efficiently, this is a reasonable algorithm.

Data Structures for State Sets

Representing State Sets

We need to represent and manipulate state sets (again)!

- How about an explicit representation, like a **hash table**?
- And how about our good old friend, the **formula**?

Time Complexity: Explicit Representations vs. Formulas

Let k be the **number of state variables**,
 $|S|$ the **number of states** in S and
 $\|S\|$ the **size of the representation** of S .

	Hash table	Formula
$s \in S?$	$O(k)$	$O(\ S\)$
$S := S \cup \{s\}$	$O(k)$	$O(k)$
$S := S \setminus \{s\}$	$O(k)$	$O(k)$
$S \cup S'$	$O(k S + k S')$	$O(1)$
$S \cap S'$	$O(k S + k S')$	$O(1)$
$S \setminus S'$	$O(k S + k S')$	$O(1)$
$\bar{S}$	$O(k2^k)$	$O(1)$
$\{s \mid s(v) = \mathbf{T}\}$	$O(k2^k)$	$O(1)$
$S = \emptyset?$	$O(1)$	co-NP-complete
$S = S'?$	$O(k S)$	co-NP-complete
$ S $	$O(1)$	#P-complete

Which Operations are Important?

- **Explicit representations** such as hash tables are unsuitable because their size grows linearly with the number of represented states.
- **Formulas** are very efficient for some operations, but not for other important operations needed by the breadth-first search algorithm.
 - Examples: $S \neq \emptyset?$, $S = S'?$

Canonical Representations

- One of the problems with formulas is that they allow **many different representations** for the same set.
 - For example, all unsatisfiable formulas represent $\emptyset$.
- This makes equality tests expensive.
- We would like data structures with a **canonical representation**, i.e., with only **one possible representation** for every state set.
 - Reduced ordered **binary decision diagrams** (BDDs) are an example of such a canonical representation.

Time Complexity: Formulas vs. BDDs

Let k be the **number of state variables**,

$|S|$ the **number of states** in S and

$\|S\|$ the **size of the representation** of S .

	Formula	BDD
$s \in S?$	$O(\ S\)$	$O(k)$
$S := S \cup \{s\}$	$O(k)$	$O(k)$
$S := S \setminus \{s\}$	$O(k)$	$O(k)$
$S \cup S'$	$O(1)$	$O(\ S\ \ S'\)$
$S \cap S'$	$O(1)$	$O(\ S\ \ S'\)$
$S \setminus S'$	$O(1)$	$O(\ S\ \ S'\)$
$\overline{S}$	$O(1)$	$O(\ S\)$
$\{s \mid s(v) = \mathbf{T}\}$	$O(1)$	$O(1)$
$S = \emptyset?$	co-NP-complete	$O(1)$
$S = S'?$	co-NP-complete	$O(1)$
$ S $	#P-complete	$O(\ S\)$

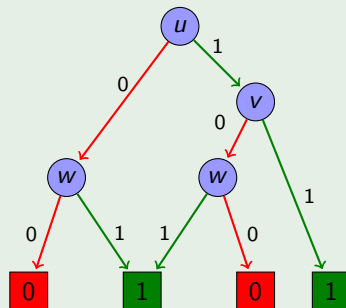
Remark: Optimizations allow BDDs with complementation ($\overline{S}$) in constant time, but we will not discuss this here.

Binary Decision Diagrams

BDD Example

Example

Possible BDD for $(u \wedge v) \vee w$



Binary Decision Diagrams: Definition

Definition (BDD)

Let V be a set of propositional variables.

A **binary decision diagram** (BDD) over V is a directed acyclic graph with labeled arcs and labeled vertices such that:

- There is exactly one node without incoming arcs.
- All sinks (nodes without outgoing arcs) are labeled **0** or **1**.
- All other nodes are labeled with a variable $v \in V$ and have exactly two outgoing arcs, labeled **0** and **1**.

A note on notation:

- In BDDs, 1 stands for **T** and 0 for **F**.
- We follow this customary notation in BDDs, but stick to **T** and **F** when speaking of logic.

Binary Decision Diagrams: Terminology

BDD Terminology

- The node without incoming arcs is called the **root**.
- The labeling variable of an internal node is called the **decision variable** of the node.
- The nodes reached from node n via the arc labeled $i \in \{0, 1\}$ is called the **i -successor** of n .
- The BDDs which only consist of a single sink are called the **zero BDD** and **one BDD**.

Observation: If B is a BDD and n is a node of B , then the subgraph induced by all nodes reachable from n is also a BDD.

- This BDD is called the **BDD rooted at n** .

BDD Semantics

Testing whether a BDD Includes a Variable Assignment

```
def bdd-includes( $B$ : BDD,  $I$ : variable assignment):  
    Set  $n$  to the root of  $B$ .  
    while  $n$  is not a sink:  
        Set  $v$  to the decision variable of  $n$ .  
        Set  $n$  to the 1-successor of  $n$  if  $I(v) = \mathbf{T}$  and  
            to the 0-successor of  $n$  if  $I(v) = \mathbf{F}$ .  
    return true if  $n$  is labeled 1, false if it is labeled 0.
```

Definition (Set Represented by a BDD)

Let B be a BDD over variables V .

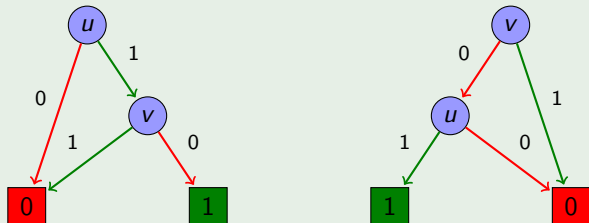
The **set represented by B** , in symbols $r(B)$, consists of all variable assignments $I : V \rightarrow \{\mathbf{T}, \mathbf{F}\}$ for which $bdd\text{-}includes(B, I)$ returns true.

BDDs as Canonical Representations

Ordered BDDs: Motivation

In general, BDDs are not a canonical representation for sets of interpretations. Here is a simple counter-example ($V = \{u, v\}$):

Example (BDDs for $u \wedge \neg v$ with Different Variable Order)



Both BDDs represent the same state set, namely the singleton set $\{\{u \mapsto \mathbf{T}, v \mapsto \mathbf{F}\}\}$.

Ordered BDDs: Definition

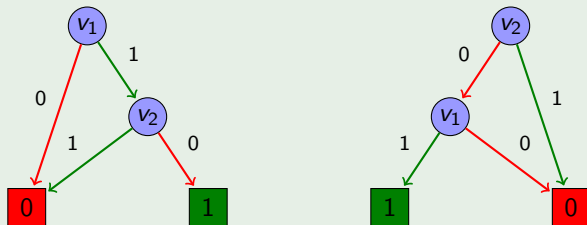
- As a first step towards a canonical representation, we now require that the set of variables is **totally ordered** by some ordering $\prec$.
- In particular, we will only use variables $v_1, v_2, v_3, \dots$ and assume the ordering $v_i \prec v_j$ iff $i < j$.

Definition (Ordered BDD)

A BDD is **ordered** (w.r.t. $\prec$) iff for each arc from a node with decision variable u to a node with decision variable v , we have $u \prec v$.

Ordered BDDs: Example

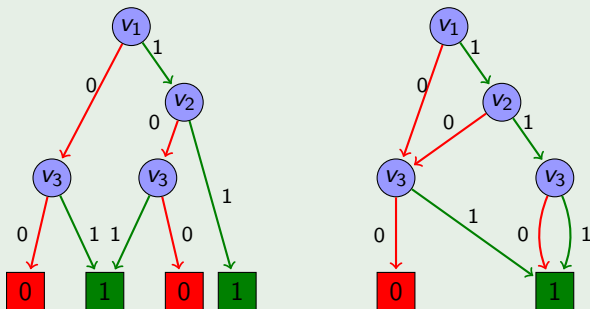
Example (Ordered and Unordered BDD)



The left BDD is ordered w.r.t. the ordering we use in this chapter, the right one is not.

Reduced Ordered BDDs: Are Ordered BDDs Canonical?

Example (Two equivalent BDDs that can be reduced)



- Ordered BDDs are still not canonical:
both ordered BDDs represent the same set.
- However, ordered BDDs can easily be **made** canonical.

Reduced Ordered BDDs: Reductions (1)

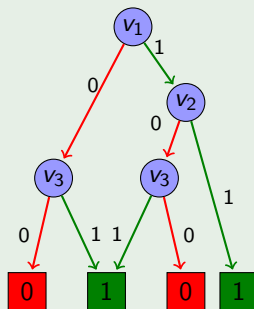
There are two important operations on BDDs that do not change the set represented by it:

Definition (Isomorphism Reduction)

If the BDDs rooted at two different nodes n and n' are **isomorphic**, then all incoming arcs of n' can be redirected to n , and all BDD nodes unreachable from the root can be removed.

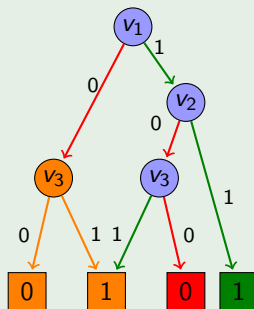
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



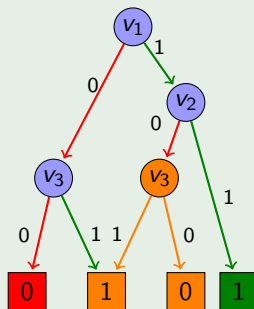
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



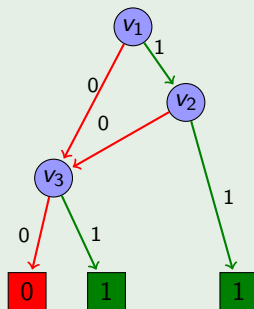
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



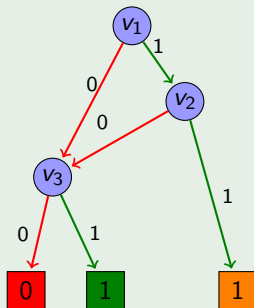
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



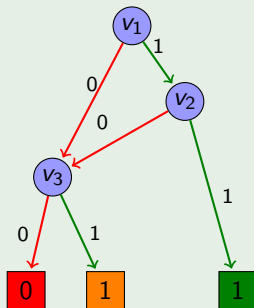
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



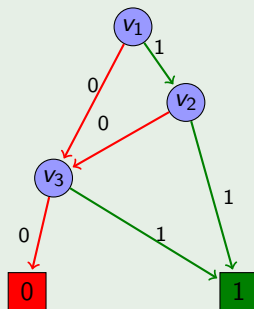
Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



Reduced Ordered BDDs: Reductions (2)

Example (Isomorphism Reduction)



Reduced Ordered BDDs: Reductions (3)

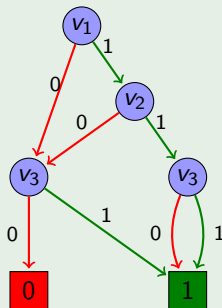
There are two important operations on BDDs that do not change the set represented by it:

Definition (Shannon Reduction)

If both outgoing arcs of an internal node n of a BDD lead to the same node m , then n can be removed from the BDD, with all incoming arcs of n going to m instead.

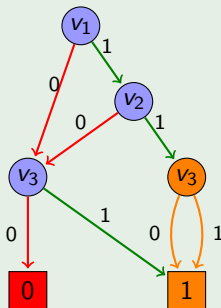
Reduced Ordered BDDs: Reductions (4)

Example (Shannon Reduction)



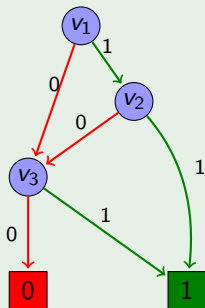
Reduced Ordered BDDs: Reductions (4)

Example (Shannon Reduction)



Reduced Ordered BDDs: Reductions (4)

Example (Shannon Reduction)



Reduced Ordered BDDs: Definition

Definition (Reduced Ordered BDD)

An ordered BDD is **reduced** iff it does not admit any isomorphism reduction or Shannon reduction.

Theorem (Bryant 1986)

For every state set S and a fixed variable ordering, there exists exactly one reduced ordered BDD representing S .

Moreover, given any ordered BDD B , the equivalent reduced ordered BDD can be computed in linear time in the size of B .

↪ Reduced ordered BDDs are the canonical representation we are looking for.

From now on, we simply say **BDD** for **reduced ordered BDD**.

Summary

Summary

- **Symbolic search** is based on the idea of performing a state-space search where many states are considered “at once” by operating on **sets of states** rather than individual states.
- **Binary decision diagrams** are a data structure to compactly represent and manipulate sets of variable assignments.
- **Reduced ordered** BDDs are a **canonical representation** of such sets.