Binary Decision Diagrams

An Introduction and Some Applications

Manas Thakur

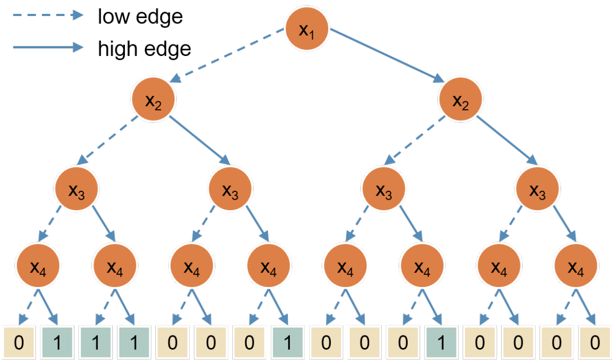
PACE Lab, IIT Madras

Programming Languages, Architecture and Compilers Education Laboratory

Binary decision tree for a truth table

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

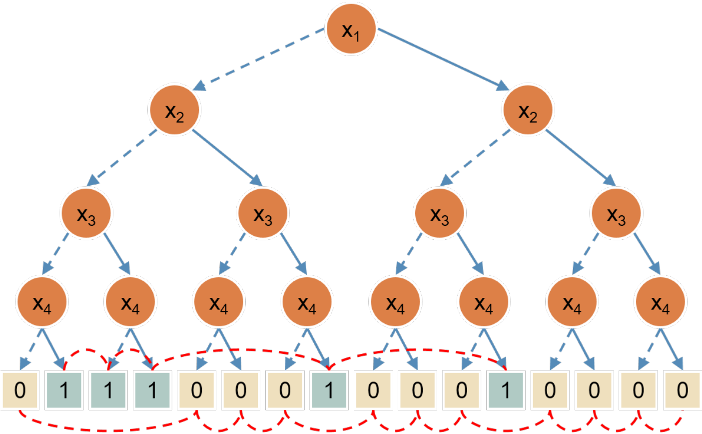
Binary decision tree for a truth table



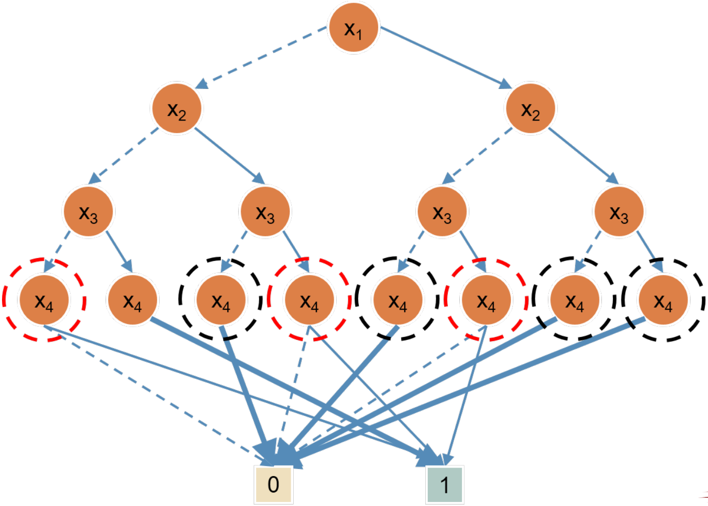
x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Collapse redundant nodes

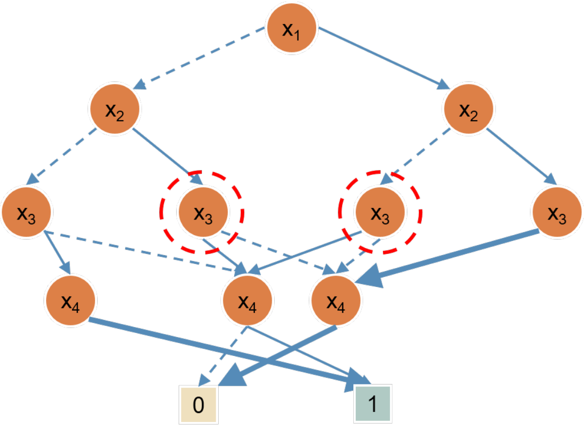


Collapse redundant nodes

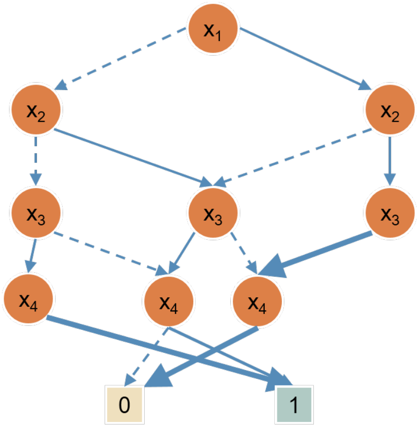


PACE
Programming Languages, Architecture and Compilers Education Laboratory

Collapse redundant nodes

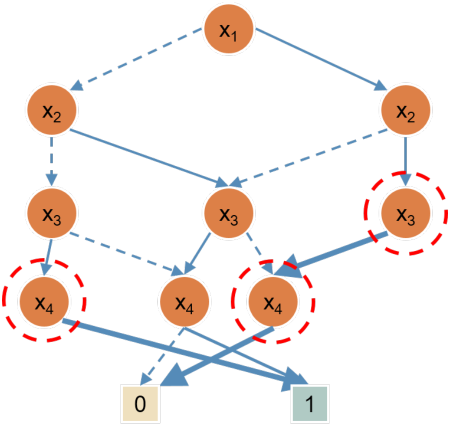


Collapse redundant nodes



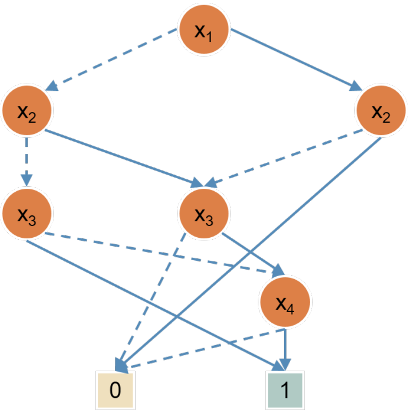
Programming Languages, Architecture and Compilers Education Laboratory

Eliminate unnecessary nodes



PACE
Programming Languages, Architecture and Compilers Education Laboratory

We got an ROBDD!!



Overview

1 Motivating Example

2 Introduction

3 Constructing ROBDDs

4 Applications

5 Conclusion

Binary Decision Diagrams

Definition

A Binary Decision Diagram is a rooted DAG with

- One or two terminal nodes of out-degree zero labeled 0 or 1
- A set of variable nodes of out-degree two

Ordered Binary Decision Diagrams (OBDDs)

- A BDD is ordered if on all paths through the graph, the variables respect a given linear order.

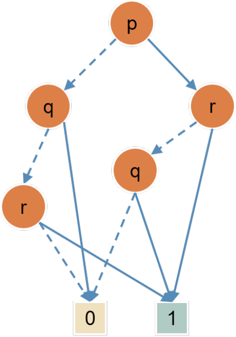
$$b_1 < b_2 < \dots < b_n$$

Ordered Binary Decision Diagrams (OBDDs)

- A BDD is ordered if on all paths through the graph, the variables respect a given linear order.

$$b_1 < b_2 < \dots < b_n$$

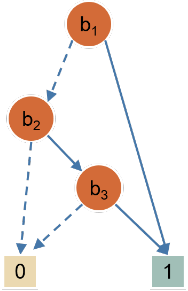
- An unordered BDD

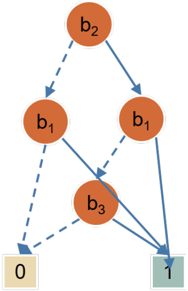


Ordered Binary Decision Diagrams (OBDDs)

- The size of a BDD depends on the variable ordering

b_1	b_2	b_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1





- The problem of finding the best variable ordering in OBDDs is NP-Complete

Reduced Ordered Binary Decision Diagrams (ROBDDs)

Definition

An OBDD is reduced if it satisfies the following properties:

- Uniqueness

$$low(u) = low(v) \text{ and } high(u) = high(v) \text{ implies } u = v$$

- Non-redundant tests

$$low(u) \neq high(u)$$

We already saw an example of ROBDDs!! See it again

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Properties of ROBDDs

- Size is correlated to amount of redundancy, NOT size of relation
  - Insight: As the relation gets larger, the number of dont-care bits increases, leading to fewer necessary nodes (usually)
- **Canonicity:** For every Boolean function, there is exactly one ROBDD representing it
  - Hence, satisfiability, tautology-check, and equivalence can be tested in deterministic time
  - For Boolean expressions, this problem is NP-Complete

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Normal forms for Boolean expressions

- Disjunctive Normal Form (DNF)
  - $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \vee \ldots \vee (a_1 \wedge a_2 \wedge \ldots \wedge a_n)$
  - Satisfiability: easy; Tautology check: hard

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Normal forms for Boolean expressions

- Disjunctive Normal Form (DNF)
  - $(a_1 \wedge a_2 \wedge ... \wedge a_n) \vee ... \vee (a_1 \wedge a_2 \wedge ... \wedge a_n)$
  - Satisfiability: easy; Tautology check: hard
- Conjunctive Normal Form (CNF)
  - $(a_1 \vee a_2 \vee ... \vee a_n) \wedge ... \wedge (a_1 \vee a_2 \vee ... \vee a_n)$
  - Tautology check: easy; Satisfiability: hard

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Normal forms for Boolean expressions

- Disjunctive Normal Form (DNF)
  - $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \vee \ldots \vee (a_1 \wedge a_2 \wedge \ldots \wedge a_n)$
  - Satisfiability: easy; Tautology check: hard
- Conjunctive Normal Form (CNF)
  - $(a_1 \vee a_2 \vee \ldots \vee a_n) \wedge \ldots \wedge (a_1 \vee a_2 \vee \ldots \vee a_n)$
  - Tautology check: easy; Satisfiability: hard
- Reduction?

Normal forms for Boolean expressions

- Disjunctive Normal Form (DNF)
  - $(a_1 \wedge a_2 \wedge \ldots \wedge a_n) \vee \ldots \vee (a_1 \wedge a_2 \wedge \ldots \wedge a_n)$
  - Satisfiability: easy; Tautology check: hard
- Conjunctive Normal Form (CNF)
  - $(a_1 \vee a_2 \vee \ldots \vee a_n) \wedge \ldots \wedge (a_1 \vee a_2 \vee \ldots \vee a_n)$
  - Tautology check: easy; Satisfiability: hard
- Reduction?
  - No hopes since conversion between CNF and DNF is exponential

PACE
Programming Languages, Architecture and Compilers Education Laboratory

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \to y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

Programming Languages, Architecture and Compilers Education Laboratory

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \to y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- More examples

$$x = x \to (1, 0)$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \rightarrow y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- More examples

$$x = x \rightarrow (1, 0)$$
$$\neg x = (x \rightarrow 0, 1)$$

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \to y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- More examples

$$x = x \to (1, 0)$$
$$\neg x = (x \to 0, 1)$$
$$x \vee y = (x \to 1, y)$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \to y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- More examples

$$x = x \to (1, 0)$$
$$\neg x = (x \to 0, 1)$$
$$x \vee y = (x \to 1, y)$$
$$x \wedge y = (x \to y, 1)$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

If-then-else Normal Form (INF)

- An If-then-else Normal Form (INF) is a Boolean expression built from the if-then-else operator and the constants 0 and 1, such that all tests are performed only on variables.

$$x \rightarrow y_0, y_1 = (x \wedge y_0) \vee (\neg x \wedge y_1)$$

- More examples

$$x = x \rightarrow (1, 0)$$
$$\neg x = (x \rightarrow 0, 1)$$
$$x \vee y = (x \rightarrow 1, y)$$
$$x \wedge y = (x \rightarrow y, 1)$$
$$x \Leftrightarrow y = x \rightarrow (y \rightarrow 1, 0), (y \rightarrow 0, 1)$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$t = x_1 \rightarrow t_1, t_0$$

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$t = x_1 \rightarrow t_1, t_0$$
$$t_0 = y_1 \rightarrow 0, t_{00}$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$t = x_1 \rightarrow t_1, t_0$$
$$t_0 = y_1 \rightarrow 0, t_{00}$$
$$t_1 = y_1 \rightarrow t_{11}, 0$$

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \to t_1, t_0 \\
t_0 &= y_1 \to 0, t_{00} \\
t_1 &= y_1 \to t_{11}, 0 \\
t_{00} &= x_2 \to t_{001}, t_{000}
\end{aligned}
$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \to t_1, t_0 \\
t_0 &= y_1 \to 0, t_{00} \\
t_1 &= y_1 \to t_{11}, 0 \\
t_{00} &= x_2 \to t_{001}, t_{000} \\
t_{11} &= x_2 \to t_{111}, t_{110}
\end{aligned}
$$

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow 0, t_{00} \\
t_1 &= y_1 \rightarrow t_{11}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, t_{000} \\
t_{11} &= x_2 \rightarrow t_{111}, t_{110} \\
t_{000} &= y_2 \rightarrow 0, 1
\end{aligned}
$$

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow 0, t_{00} \\
t_1 &= y_1 \rightarrow t_{11}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, t_{000} \\
t_{11} &= x_2 \rightarrow t_{111}, t_{110} \\
t_{000} &= y_2 \rightarrow 0, 1 \\
t_{001} &= y_2 \rightarrow 1, 0
\end{aligned}
$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow 0, t_{00} \\
t_1 &= y_1 \rightarrow t_{11}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, t_{000} \\
t_{11} &= x_2 \rightarrow t_{111}, t_{110} \\
t_{000} &= y_2 \rightarrow 0, 1 \\
t_{001} &= y_2 \rightarrow 1, 0 \\
t_{110} &= y_2 \rightarrow 0, 1
\end{aligned}
$$

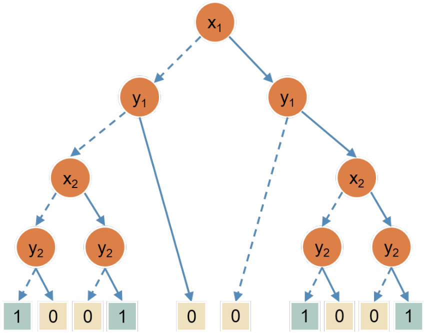
PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$
\begin{aligned}
t &= x_1 \rightarrow t_1, t_0 \\
t_0 &= y_1 \rightarrow 0, t_{00} \\
t_1 &= y_1 \rightarrow t_{11}, 0 \\
t_{00} &= x_2 \rightarrow t_{001}, t_{000} \\
t_{11} &= x_2 \rightarrow t_{111}, t_{110} \\
t_{000} &= y_2 \rightarrow 0, 1 \\
t_{001} &= y_2 \rightarrow 1, 0 \\
t_{110} &= y_2 \rightarrow 0, 1 \\
t_{111} &= y_2 \rightarrow 1, 0
\end{aligned}
$$

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$t = x_1 \rightarrow t_1, t_0$$
$$t_0 = y_1 \rightarrow 0, t_{00}$$
$$t_1 = y_1 \rightarrow t_{11}, 0$$
$$t_{00} = x_2 \rightarrow t_{001}, t_{000}$$
$$t_{11} = x_2 \rightarrow t_{111}, t_{110}$$
$$t_{000} = y_2 \rightarrow 0, 1$$
$$t_{001} = y_2 \rightarrow 1, 0$$
$$t_{110} = y_2 \rightarrow 0, 1$$
$$t_{111} = y_2 \rightarrow 1, 0$$



Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

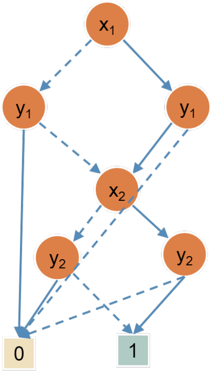
$$
\begin{aligned}
t &= x_1 \to t_1, t_0 \\
t_0 &= y_1 \to 0, t_{00} \\
t_1 &= y_1 \to t_{11}, 0 \\
t_{00} &= x_2 \to t_{001}, t_{000} \\
t_{11} &= x_2 \to t_{111}, t_{110} \\
t_{000} &= y_2 \to 0, 1 \\
t_{001} &= y_2 \to 1, 0 \\
t_{110} &= y_2 \to 0, 1 \\
t_{111} &= y_2 \to 1, 0
\end{aligned}
$$

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Example: $t = (x_1 \Leftrightarrow y_1) \wedge (x_2 \Leftrightarrow y_2)$

$$t = x_1 \rightarrow t_1, t_0$$
$$t_0 = y_1 \rightarrow 0, t_{00}$$
$$t_1 = y_1 \rightarrow t_{11}, 0$$
$$t_{00} = x_2 \rightarrow t_{001}, t_{000}$$
$$t_{11} = x_2 \rightarrow t_{111}, t_{110}$$
$$t_{000} = y_2 \rightarrow 0, 1$$
$$t_{001} = y_2 \rightarrow 1, 0$$
$$t_{110} = y_2 \rightarrow 0, 1$$
$$t_{111} = y_2 \rightarrow 1, 0$$

$$t = x_1 \rightarrow t_1, t_0$$
$$t_0 = y_1 \rightarrow 0, t_{00}$$
$$t_1 = y_1 \rightarrow t_{00}, 0$$
$$t_{00} = x_2 \rightarrow t_{001}, t_{000}$$
$$t_{000} = y_2 \rightarrow 0, 1$$
$$t_{001} = y_2 \rightarrow 1, 0$$



Let's move ahead

1 Motivating Example

2 Introduction

3 Constructing ROBDDs

4 Applications

5 Conclusion

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Applications of BDDs

- Correctness of Combinational Circuits
- Equivalence of Combinational Circuits
- Model Checking

- And yes, Program Analysis!

BDDs for representing Points-to relation

- Points-to analysis using BDDs. Berndl et al. PLDI'03.
- Let a,b,c be reference variables and A,B,C be object references.
- The points-to relation (a,A),(a,B),(b,A),(b,B),(c,A),(c,B),(c,C) is represented as:

PACE
Programming Languages, Architecture and Compilers Education Laboratory

BDDs for representing Points-to relation

- Points-to analysis using BDDs. Berndl et al. PLDI'03.
- Let a,b,c be reference variables and A,B,C be object references.
- The points-to relation (a,A),(a,B),(b,A),(b,B),(c,A),(c,B),(c,C) is represented as:

On the board.

PACE
Programming Languages, Architecture and Compilers Education Laboratory

bddbddb

- Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams. John Whaley and Monica S. Lam. PLDI'04.

bddbddb

- Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams. John Whaley and Monica S. Lam. PLDI'04.

On the Board

1. Represent program information using relations
   Express relations as BDDs

PACE
Programming Languages, Architecture and Compilers Education Laboratory

bddbddb

- Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams. John Whaley and Monica S. Lam. PLDI'04.

On the Board

1. Represent program information using relations
   Express relations as BDDs
2. Write program analyses as Datalog queries
   Express queries as BDD operations

PACE
Programming Languages, Architecture and Compilers Education Laboratory

bddbddb

- Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams. John Whaley and Monica S. Lam. PLDI'04.

On the Board

1. Represent program information using relations
   Express relations as BDDs
2. Write program analyses as Datalog queries
   Express queries as BDD operations
3. Get solutions!
   Perform operations on BDDs

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Pointers for the enthusiast

- An Introduction to Binary Decision Diagrams. Tutorial by Henrik Reif Andersen.
- Program Analysis using Binary Decision Diagrams. Ondrej Lhotak's PhD Thesis (2006).
- Context-Sensitive Pointer Analysis using Binary Decision Diagrams. John Whaley's PhD Thesis (2007).
- Fun with Binary Decision Diagrams. Video lecture by Donald Knuth.

PACE
Programming Languages, Architecture and Compilers Education Laboratory

Conclusion

Programming Languages, Architecture and Compilers Education Laboratory

Conclusion

BDDs are very interesting and useful!