CS 243
Lecture 11
Binary Decision Diagrams (BDDs)
in Pointer Analysis

1. Datalog → BDD
2. BDDs
3. Context-Sensitive Pointer Analysis
4. Performance of BDD Algorithms

Readings: Chapter 12

Automatic Analysis Generation

Programmer:
Security analysis in 10 lines

Compiler writer:
Ptr analysis in 10 lines

BDD operations
BDD: 10,000s-lines library

PQL
Datalog

bddbddb
(BDD-based deductive database) with Active Machine Learning

1000s of lines 1 year tuning

Advanced Compilers M. Lam & J. Whaley
1. Datalog → BDDs

- Example

```
calls(A,B)
calls(A,C)
calls(A,D)
calls(B,D)
calls(C,D)
```

```
A

B  C

D
```

Call Graph Relation

- Relation expressed as a binary function.
  - A=00, B=01, C=10, D=11

```
\begin{array}{cccc|c}
  x_1 & x_2 & x_3 & x_4 & f \\
  0   & 0   & 0   & 0   & 0   \\
  0   & 0   & 1   & 1   & 1   \\
  0   & 0   & 1   & 0   & 0   \\
  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   \\
\end{array}
```
Binary Decision Diagrams
(Bryant, 1986)

- Graphical encoding of a truth table.

```
  x1
 / \  \
 x2  x2
 /   /
 x3  x3
/    /
 x4  x4
```

```
0 1 1 1 0 0 0 1 0 0 1 0 0 0 0
```

Binary Decision Diagrams

- Collapse redundant nodes.

```
  x1
 / \  \
 x2  x2
 /   /
 x3  x3
/    /
 x4  x4
```

```
0 1 1 1 0 0 0 1 0 0 1 0 0 0 0
```
Binary Decision Diagrams

- Collapse redundant nodes.

Advanced Compilers
L11. BDDs
Binary Decision Diagrams

- Collapse redundant nodes.

Binary Decision Diagrams

- Eliminate unnecessary nodes.
Binary Decision Diagrams

- Eliminate unnecessary nodes.

Datalog $\rightarrow$ BDDs

<table>
<thead>
<tr>
<th>Datalog</th>
<th>BDDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relations</td>
<td>Boolean functions</td>
</tr>
<tr>
<td>Relation ops: $\bowtie$, $\cup$, select, project</td>
<td>Boolean function ops: $\land$, $\lor$, $\neg$, $\sim$</td>
</tr>
<tr>
<td>Relation at a time</td>
<td>Function at a time</td>
</tr>
<tr>
<td>Semi-naïve evaluation</td>
<td>Incrementalization</td>
</tr>
<tr>
<td>Fixed-point</td>
<td>Iterate until stable</td>
</tr>
</tbody>
</table>
2. Binary Decision Diagrams

- Represent tiny and huge relations compactly
- Size depends on redundancy
  - Similar contexts have similar numberings
  - Variable ordering in BDDs

BDD Variable Order is Important!

\[ x_1x_2 + x_3x_4 \]

\[ x_1 < x_2 < x_3 < x_4 \]
\[ x_1 < x_3 < x_2 < x_4 \]
Expanded Call Graph

Numbering Clones
3. Context-Sensitive Pointer Analysis Algorithm

1. First, do context-insensitive pointer analysis to get call graph.
2. Number clones.
3. Do context-insensitive algorithm on the cloned graph.
   - Results explicitly generated for every clone.
   - Individual results retrievable with Datalog query.

4. Performance of BDD Algorithm

- Direct implementation
  - Does not finish even for small programs
  - > 3000 lines of code
- Requires tuning for about 1 year
- Easy to make mistakes
  - Mistakes found months later
An Adventure in BDDs

- Context-sensitive numbering scheme
  - Modify BDD library to add special operations.
  - Can’t even analyze small programs.  \( \text{Time: } \infty \)

- Improved variable ordering
  - Group similar BDD variables together.
  - Interleave equivalence relations.
  - Move common subsets to edges of variable order.  \( \text{Time: } 40h \)

- Incrementalize outermost loop
  - Very tricky, many bugs.  \( \text{Time: } 36h \)

- Factor away control flow, assignments
  - Reduces number of variables  \( \text{Time: } 32h \)


An Adventure in BDDs

- Exhaustive search for best BDD order
  - Limit search space by not considering intradomain orderings.  \( \text{Time: } 10h \)

- Eliminate expensive rename operations
  - When rename changes relative order, result is not isomorphic.  \( \text{Time: } 7h \)

- Improved BDD memory layout
  - Preallocate to guarantee contiguous.  \( \text{Time: } 6h \)

- BDD operation cache tuning
  - Too small: redo work, too big: bad locality
  - Parameter sweep to find best values.  \( \text{Time: } 2h \)
An Adventure in BDDs

- Simplified treatment of exceptions
  - Reduce number of vars, iterations necessary for convergence.  \(\text{Time: 1h}\)

- Change iteration order
  - Required redoing much of the code.  \(\text{Time: 48m}\)

- Eliminate redundant operations
  - Introduced subtle bugs.  \(\text{Time: 45m}\)

- Specialized caches for different operations
  - Different caches for and, or, etc.  \(\text{Time: 41m}\)

Compacted BDD nodes
- 20 bytes \(\rightarrow\) 16 bytes  \(\text{Time: 38m}\)

Improved BDD hashing function
- Simpler hash function.  \(\text{Time: 37m}\)

Total development time: 1 year
- 1 year per analysis?!?

Optimizations obscured the algorithm.
- Many bugs discovered, maybe still more.
Variable Numbering:
Active Machine Learning

- Must be determined dynamically
- Limit trials with properties of relations
- Each trial may take a long time
- Active learning:
  select trials based on uncertainty
- Several hours
- Comparable to exhaustive for small apps

Optimizations in bddbdbdb

- Algorithmic
  - Clever context numbering to exploit similarities
- Query optimizations
  - Magic-set transformation
  - semi-naive evaluation
- Compiler optimizations
  - Redundancy elimination, liveness analysis
- BDD optimizations
  - Active machine learning
- BDD library extensions and turning
Software

- System is publicly available at: http://bddbddb.sourceforge.net
- A ready-to-use version is available as a LivePC at http://www.moka5.com