Solving Systems of Boolean Polynomials Using Binary Decision Diagrams

Håvard Raddum
Simula Research Laboratories

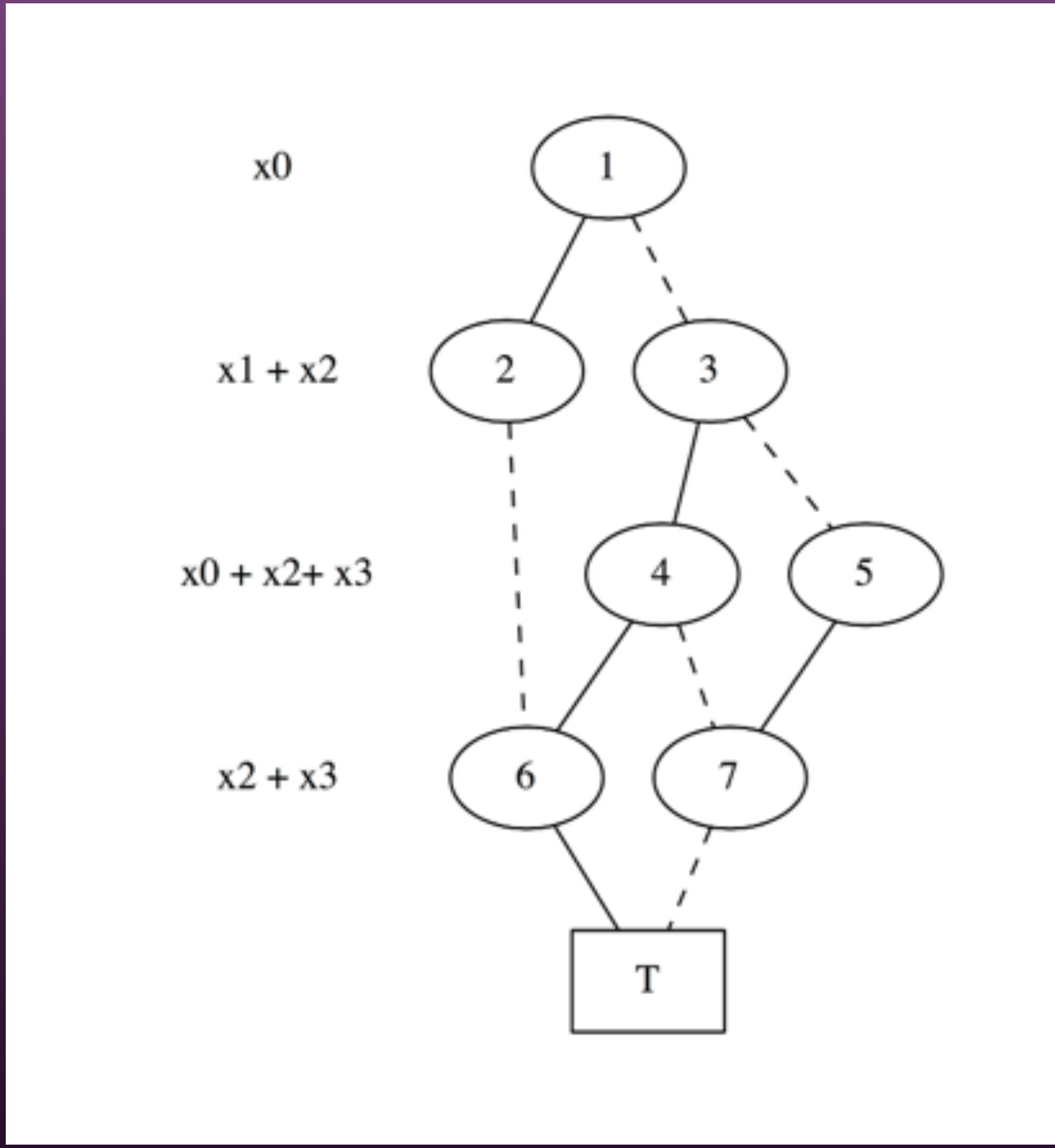
Solving equation systems

- Solving (non-linear) system of equations is NP-hard in general
- Several solving algorithms exist, which is the best?
- Equations may be represented as
  - Boolean polynomials
  - SAT formulas
  - MRHS
  - Binary Decision Diagrams (BDDs)

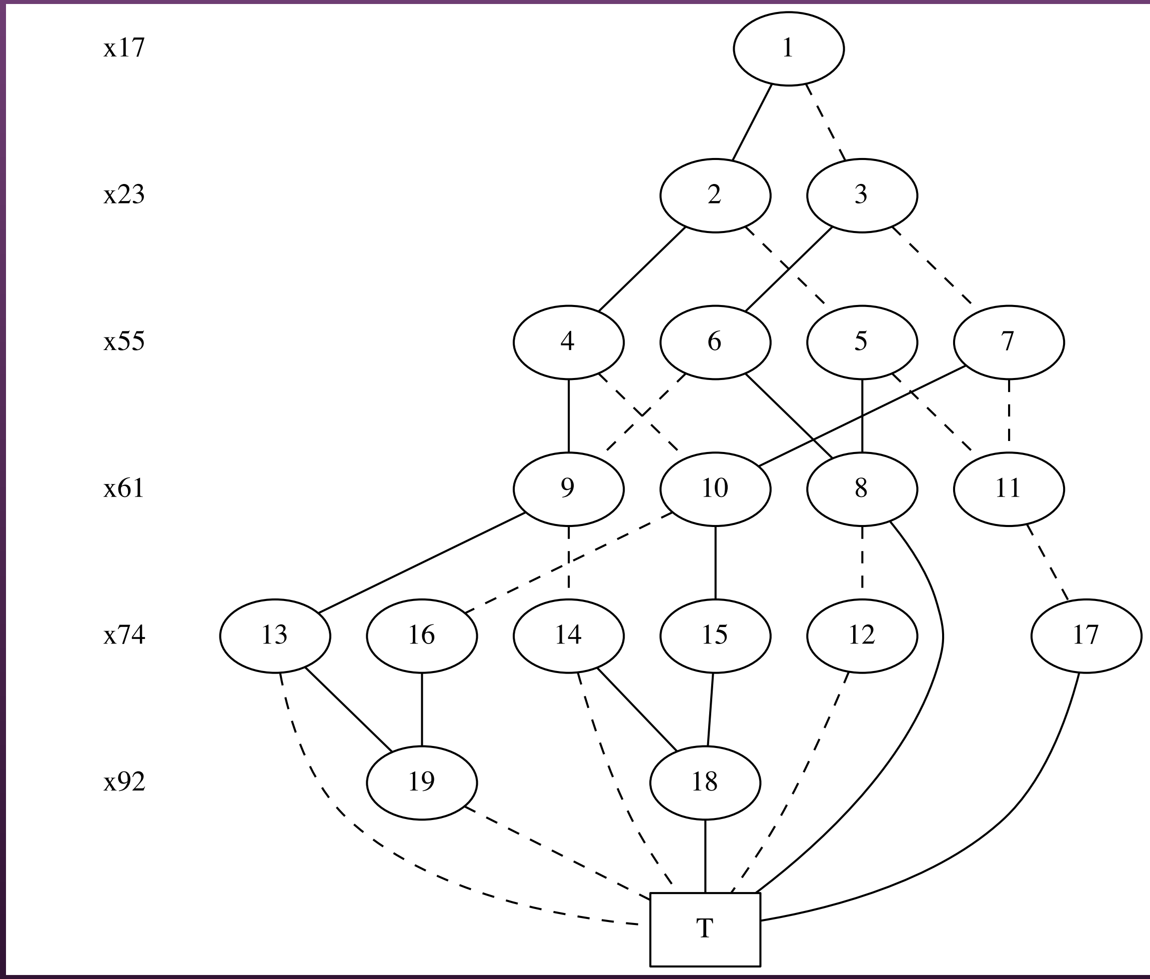
Binary Decision Diagrams
(in this talk)

- Directed acyclic graph starting in one source node and ending in one sink node
- Drawn top to bottom, nodes in horizontal levels
- No edges between nodes on same level
- At most two out-going edges from each node, called 0-edge and 1-edge
- Nodes on same level associated to some linear combination of variables

Examples



x0
x1 + x2
x0 + x2+ x3
x2 + x3
1
2
3
4
5
6
7
T



x17
x23
x55
x61
x74
x92
1
2
3
4
6
5
7
9
10
8
11
13
16
14
15
12
17
19
18
T

Constructing BDD systems

Constructing BDDs

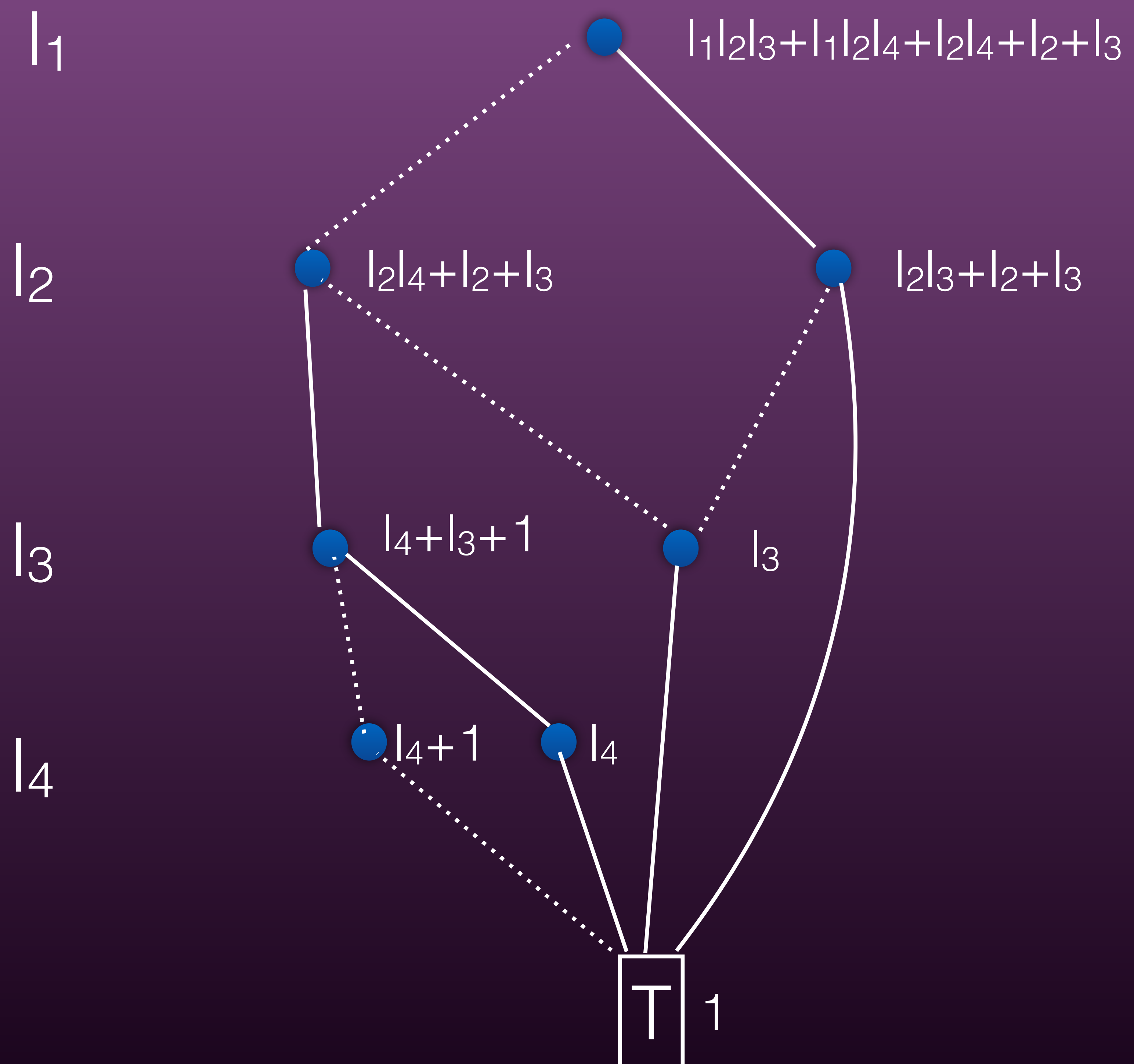
- Easy construction of BDD from any Boolean polynomial
- May also construct BDD directly from non-linear components (S-boxes, + mod 2^n, bitwise AND,..)

Boolean Equation to BDD

- $f(l_1(x),\ldots,l_n(x)) = 1$
- Assign f to source node, 1 to sink node and associate $l_1(x)$ to level 1 (top level)
- For $i=2\ldots n$
  - For each node A on level i-1 (ass. to func. $g\neq 0$)
    - make two nodes on level i, connected to A by 0-edge and 1-edge
    - assign $g|_{l_{i-1}(x)=0}$ and $g|_{l_{i-1}(x)=1}$ ($\neq 0$) to new nodes on level i
  - Associate $l_i(x)$ to level i

Example

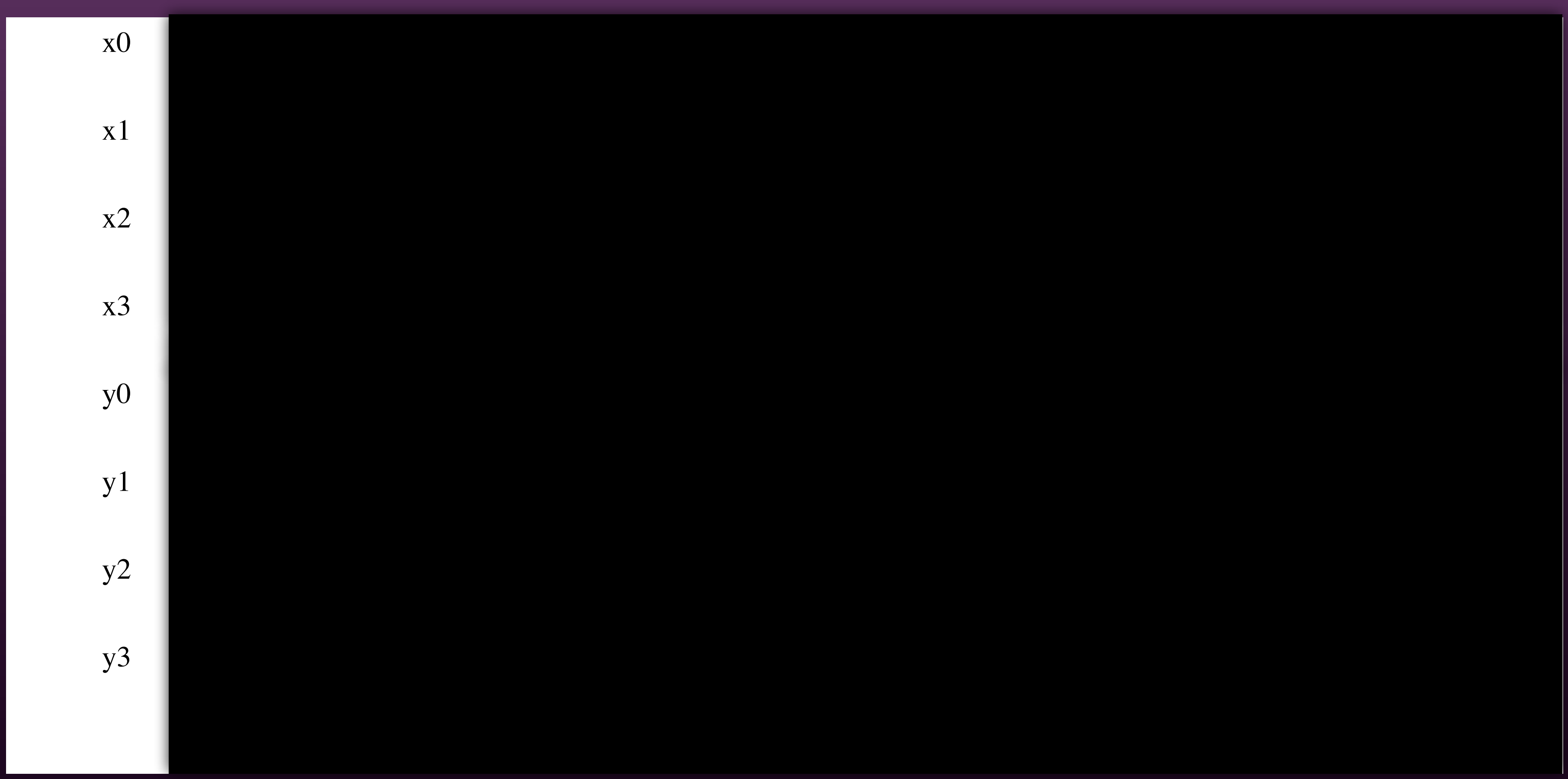
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



BDD representing S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
y	5	C	8	F	9	7	2	B	6	A	0	D	E	4	3	1

y=S(**x**)



Constructing system

$$f_1(l_{11},\ldots,l_{1k}) = 1$$
$$\ldots$$
$$f_n(l_{n1},\ldots,l_{nk}) = 1$$

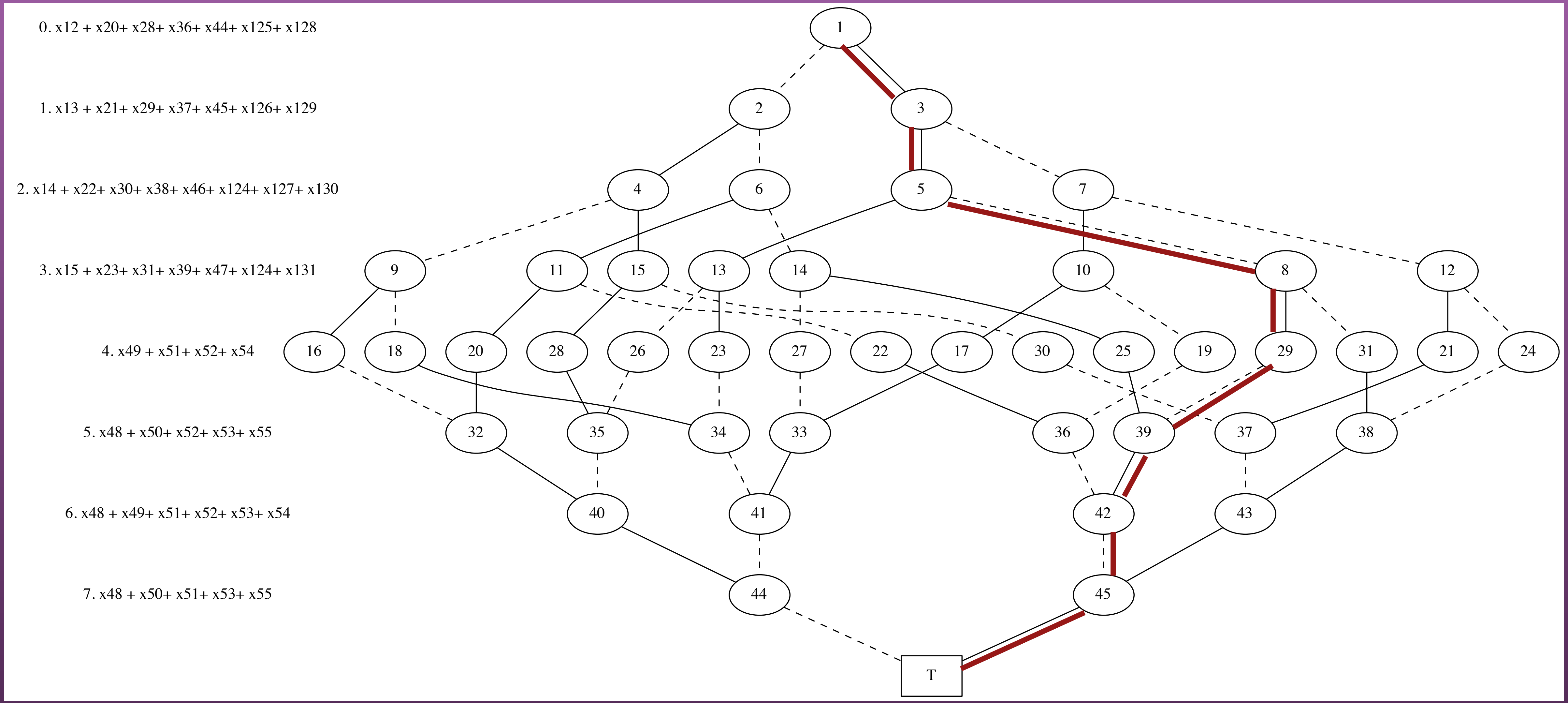
k relatively small,
$l_{ij} = l_{ij}(x_1,..,x_n)$

- Build one BDD for each f_i (or non-lin. component)
- Set of BDDs = representation of equation system (cryptographic primitive)

Solving BDD systems

Paths = valid assignments

- Set of paths from source to sink nodes in BDD describe constraint of equation
- Selecting a path assigns values to linear combinations
- The edge out from a node on a level gives value to lin. comb. associated with level
- One path gives right-hand side to linear system



$$x12 + x20 + x28 + x36 + x44 + x125 + x128 = 1$$

$$x13 + x21 + x29 + x37 + x45 + x126 + x129 = 1$$

$$x14 + x22 + x30 + x38 + x46 + x124 + x127 + x130 = 0$$

$$x15 + x23 + x31 + x39 + x47 + x124 + x131 = 1$$

$$x49 + x51 + x52 + x54 = 0$$

$$x48 + x50 + x52 + x53 + x55 = 1$$

$$x48 + x49 + x51 + x52 + x53 + x54 = 0$$

$$x48 + x50 + x51 + x53 + x55 = 1$$

Naive solving attempt

- Select a path from each BDD
- Collect linear systems from each BDD into one big linear system
- Solve big linear system
- Solution found :-)

Naive failure

- Big linear system is overdefined, with lots of dependencies among lin. combs.
- Selected paths will, in all likelihood, lead to an inconsistent system
- No solution :-(

Operations on BDDs

- We may manipulate a BDD to:
  - ✦ Reduce the BDD (remove redundant nodes)
  - ✦ Swap the lin. combs. of two adjacent levels
  - ✦ Add (xor) the lin. combs. of two adjacent levels

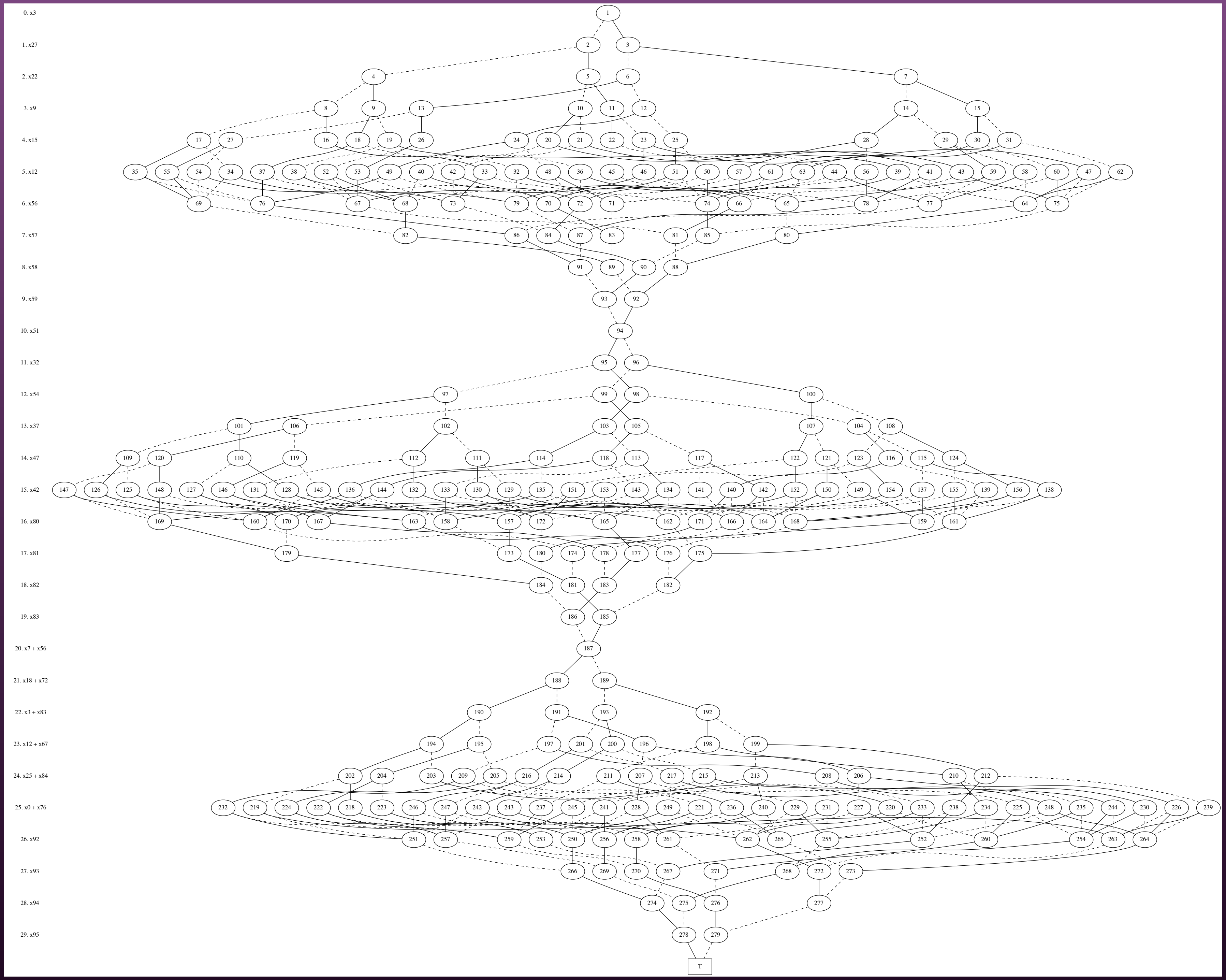
BDD Operations

- BDD reduction runs in polynomial time
- Swapping/adding levels are local operations, only affecting the two involved levels
- May swap/add repeatedly to perform Gaussian elimination on lin. combs. of BDD

Joining BDDs

- Two or more BDDs may be joined into one BDD very easily
  - ✦ To join two BDDs, replace the sink node of one with the source node of the other

Three joined BDDs



Linear absorption

- Assume a BDD where some lin. combs. are linearly dependent
- Use add/swap repeatedly to add dependent lin. combs. together
- Creates the 0-vector for a level

Linear absorption

x1 + x4

x2

x3 + x7

x3 + x4 + x6

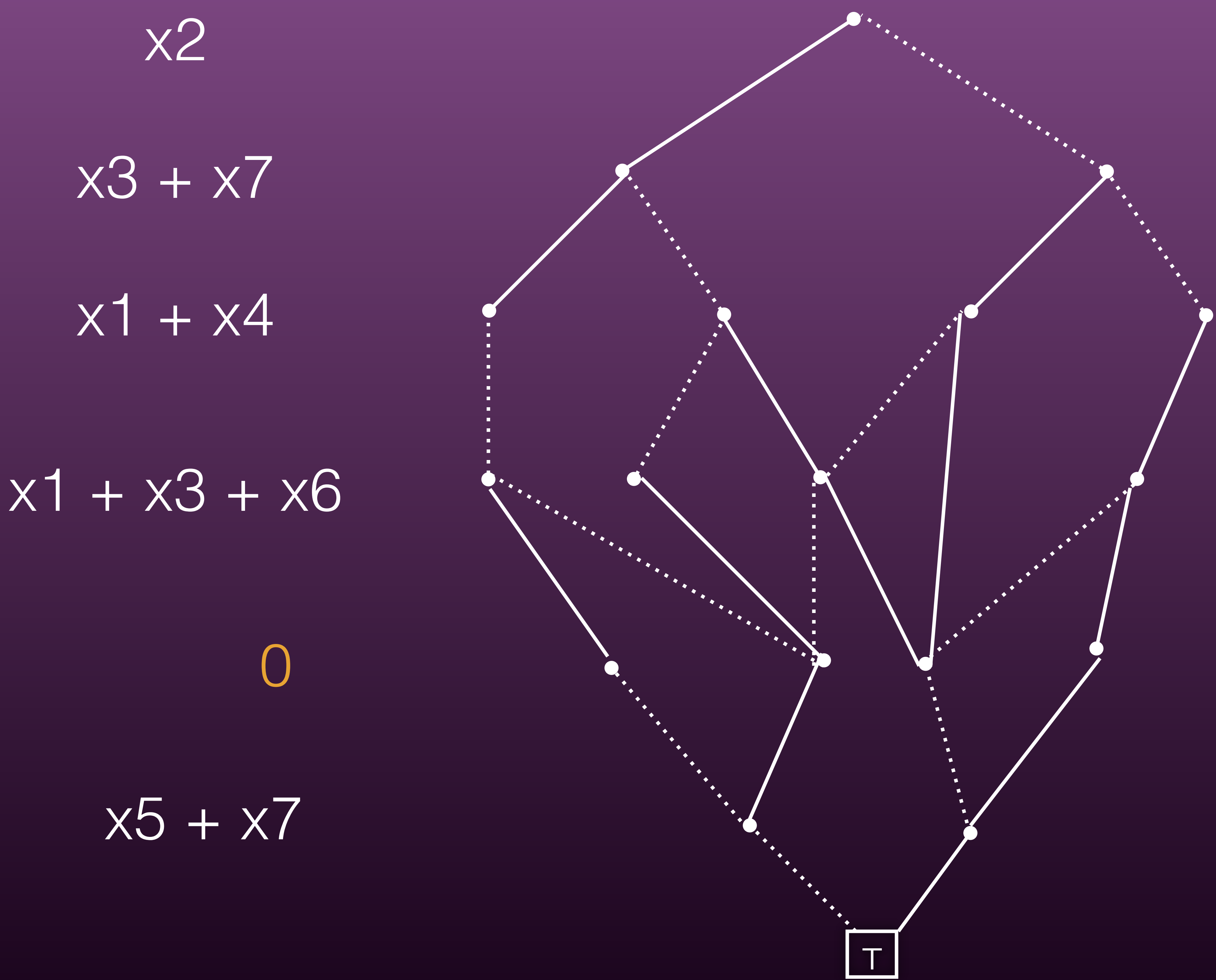
x1 + x3 + x6

x5 + x7

Level with 0-vector

- Level associated with 0-vector = 0-level
- Selecting 1-edge out from 0-level gives «0=1» assignment
- Remove all 1-edges out from nodes on 0-level

Linear absorption



Removing 0-level

- A node on 0-level and its child along 0-edge represent the same Boolean function
- All nodes on 0-level are merged with their 0-child, effectively removing whole level
- The linear dependency we started with has been absorbed in the BDD

Linear absorption

$x2$

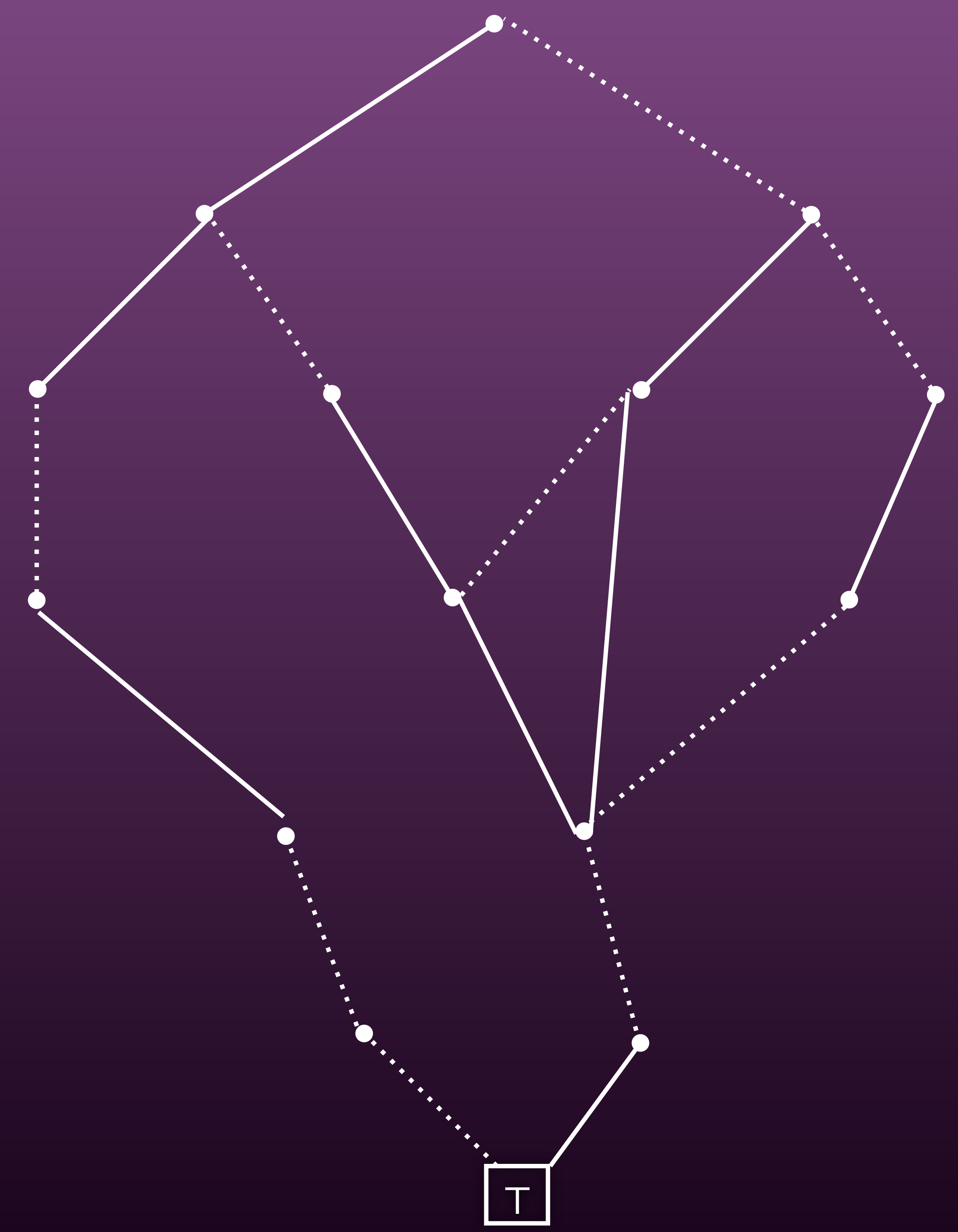
$x3 + x7$

$x1 + x4$

$x1 + x3 + x6$

0

$x5 + x7$



General solving algorithm

- While more than 1 BDD in system
  - ✦ Join some BDDs (in some order) creating a BDD with lin. dependencies
  - ✦ Absorb lin. dependencies
- Any remaining path in final BDD gives right-hand side leading to consistent linear system
- Solve linear system

Complexity

- Number of nodes on one level may (worst case) double when swapping or adding levels
- Absorbing one linear dependency may double the size of BDD
- In practice: very far from worst-case behavior

Some practical results and examples

DES

- 2007: Eq. system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)
- But…necessary to first fix 20 bits of the key to correct values
- BDD system for 6-round DES solved in the same time without guessing (8 chosen plaintexts)

Determining EA-equivalence

- To vectorial functions F, G are EA-equivalent if
- $F(x) = M_1 \cdot G(M_2 x + V_2) + M_3 x + V_1$ for all x
- M_i are $n \times n$ matrices and V_j are n-bit vectors, M_1 and M_2 invertible
- May create equation system describing EA-equivalence, entries to M_i and V_j are variables (number of vars. is $3n^2 + 2n$)

Finding EA-equivalence

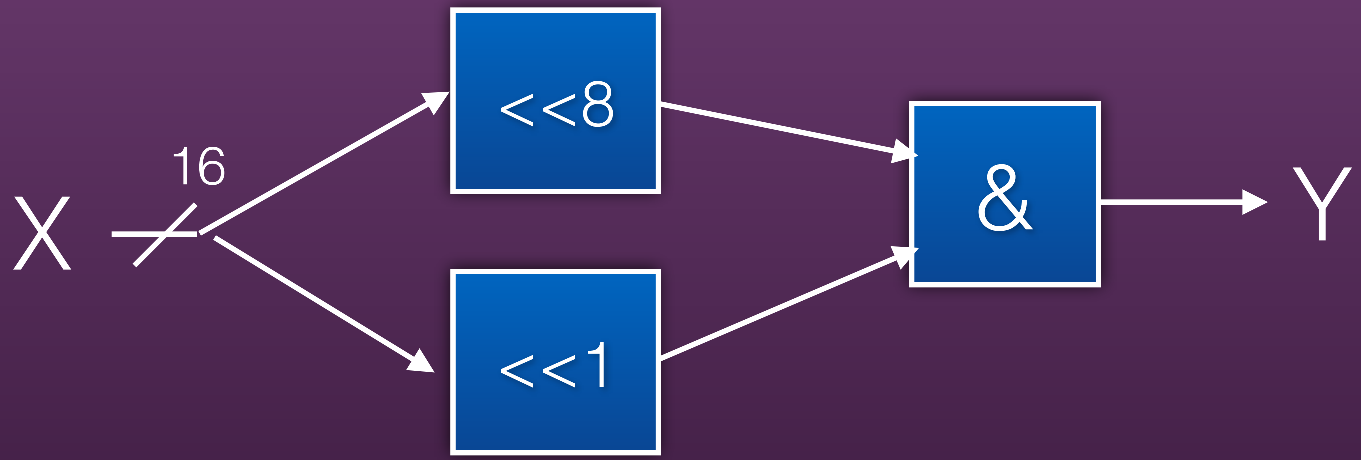
- A few experiments for n=4 and n=5

Instance	n	Number of solutions	Time (sec) BDD	Time (sec) CryptoMiniSat
1	4	2	2	2
2	4	60	2	2
3	4	2	2	2
4	5	1	2	>2
5	5	155	2	>2

* Not finished after 78 hours

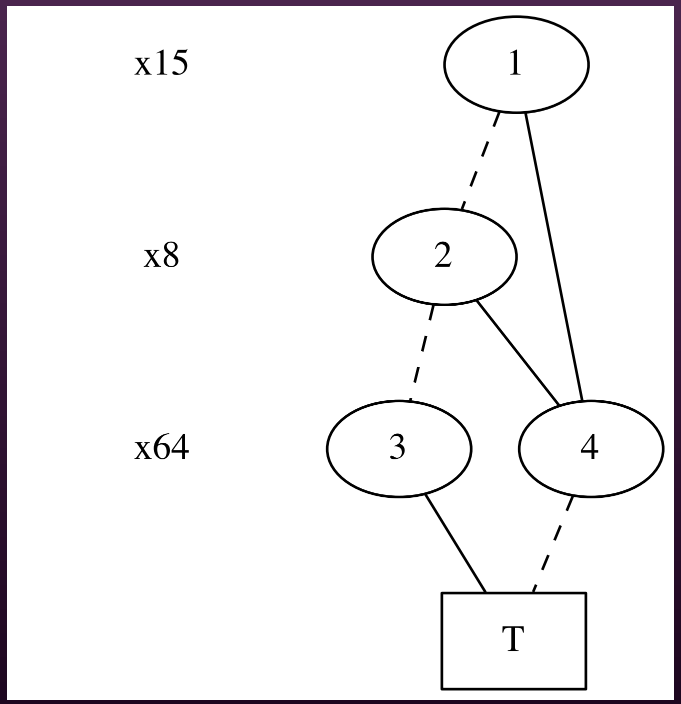
Simon-32

- Feistel cipher with very simple non-linear component



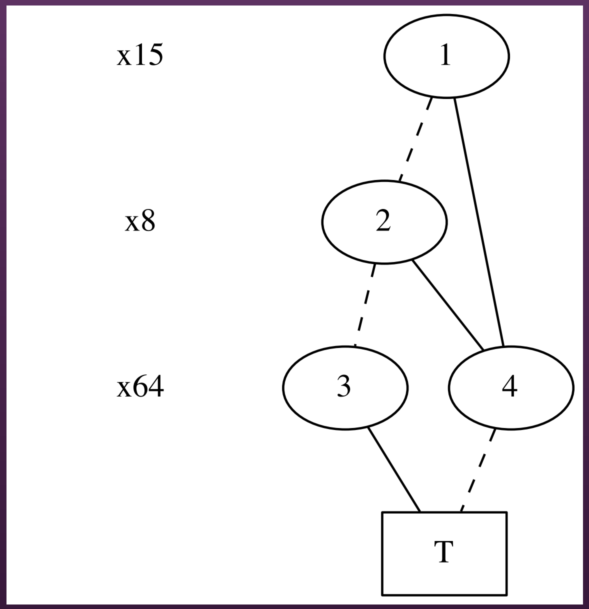
BDD for

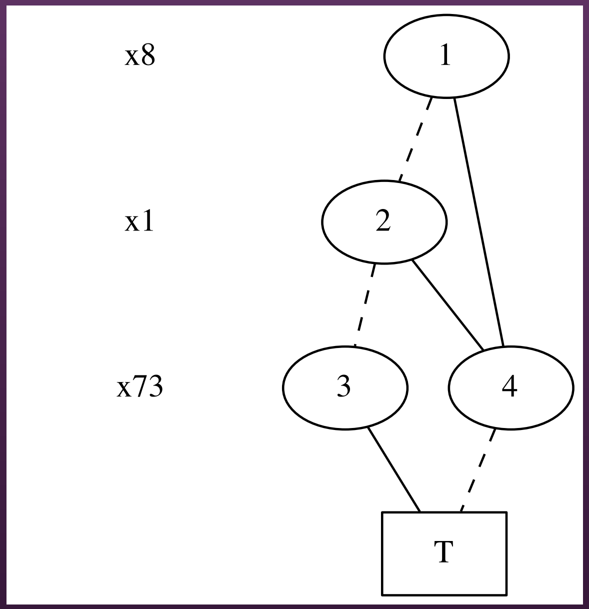
$(x_{15} + 1) \cdot (x_8 + 1) = x_{64}$



Simon-32

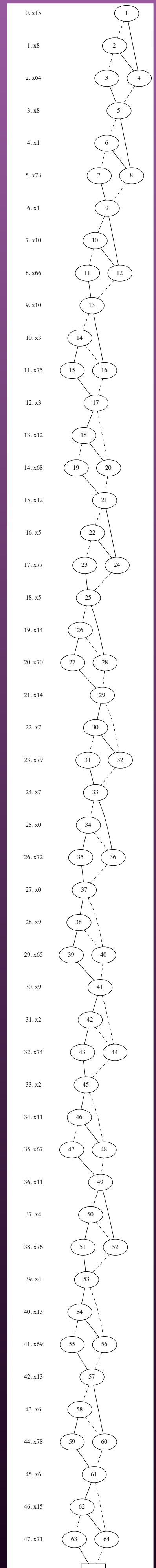
- 16 BDDs for one round
- Each input variable appears in two BDDs



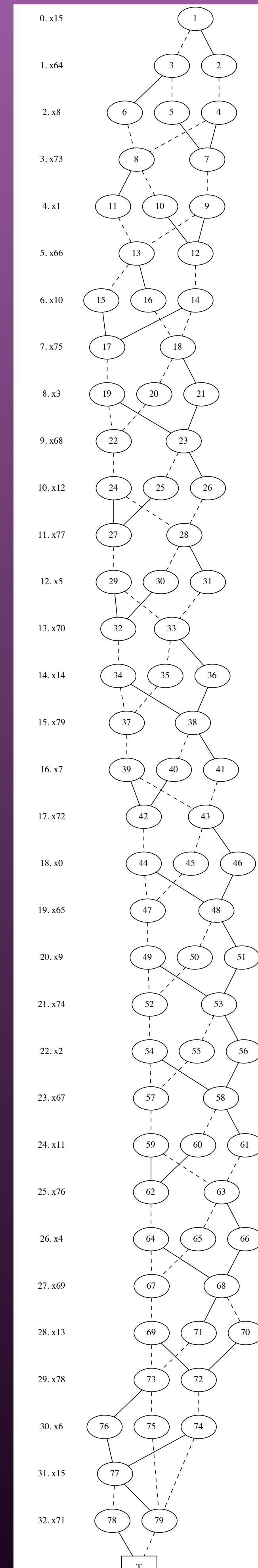


- Join: let consecutive BDDs share one variable

Joined 16 BDDs



Absorbed 15 dependencies



Absorbed 16 dependencies