Solving Systems of Boolean Polynomials Using Binary Decision Diagrams

Oleksandr Kazymyrov[1] Håvard Raddum[2]

[1] University of Bergen
[2] Simula Research Laboratories
Norway

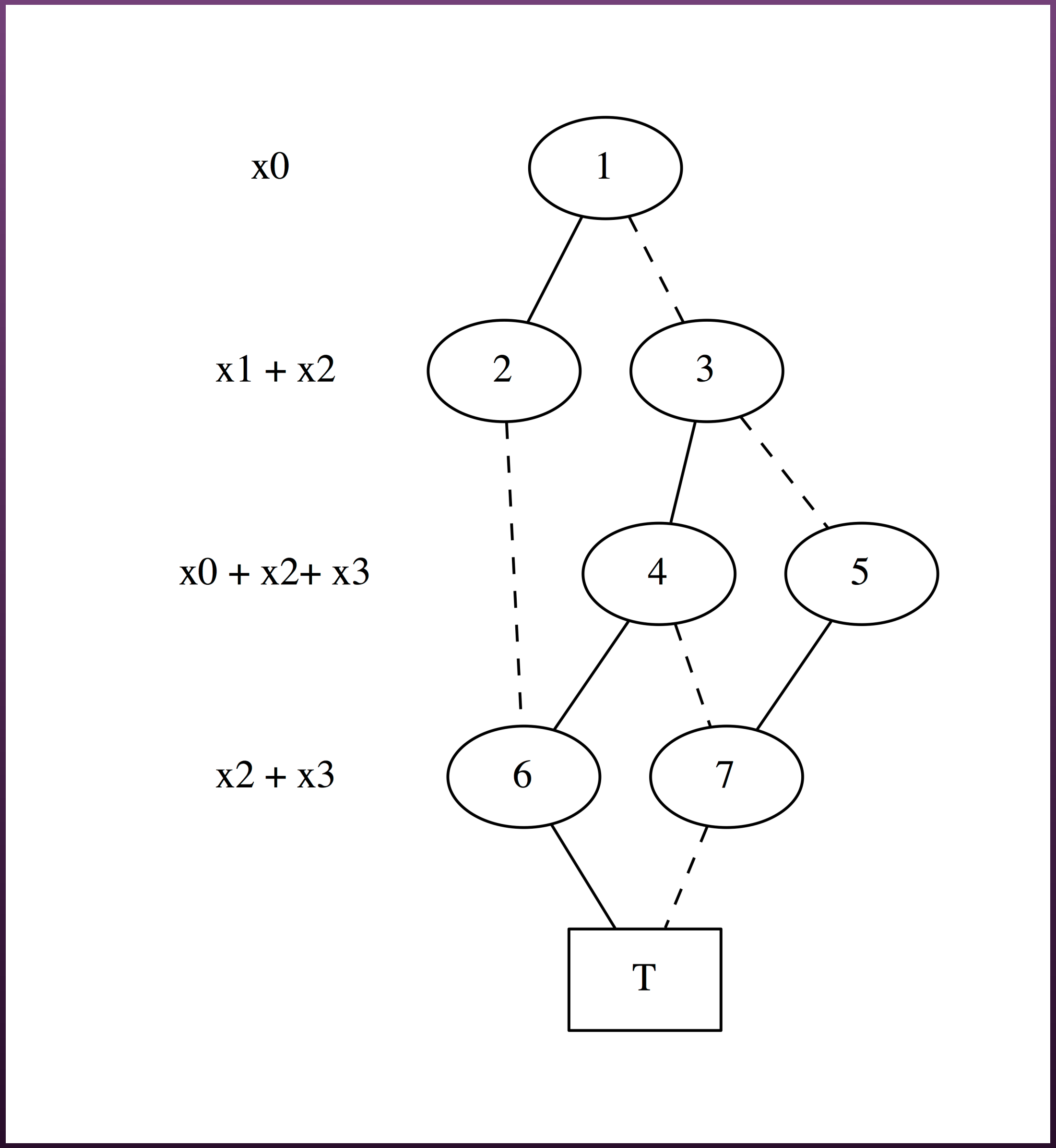
Solving equation systems

- 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)

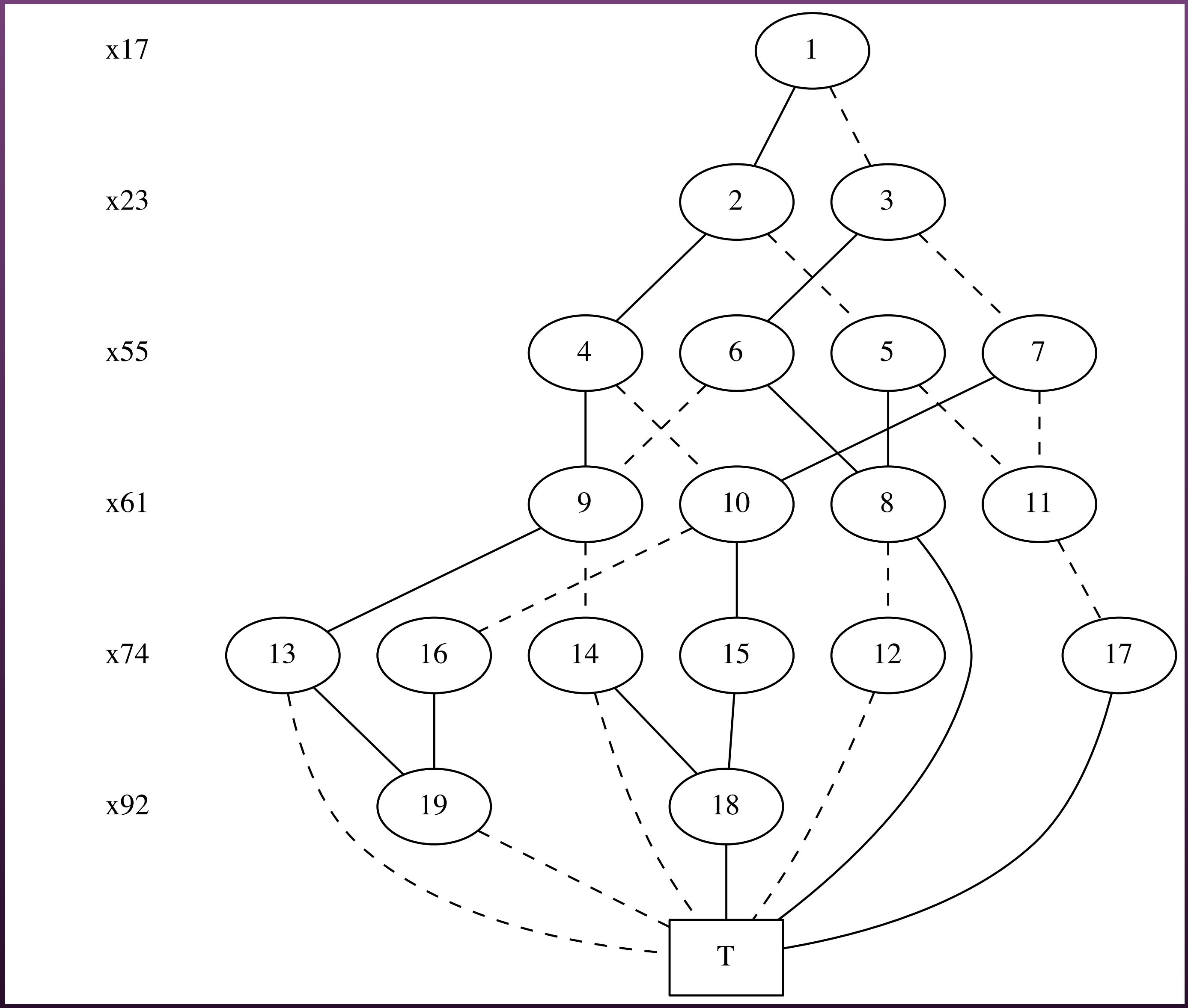
Binary Decision Diagrams
(in this talk)

- 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 the 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…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

$f(I_1,I_2,I_3,I_4) = I_1I_2I_3+I_1I_2I_4+I_2I_4+I_2+I_3 = 1$

Example

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

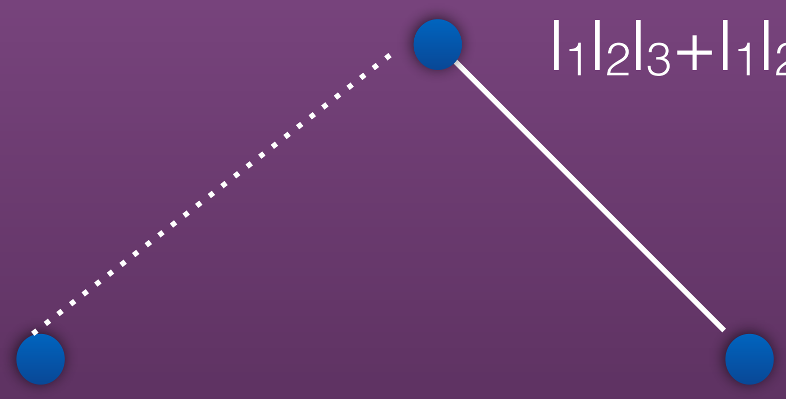
l_1

- $l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3$

T 1

Example

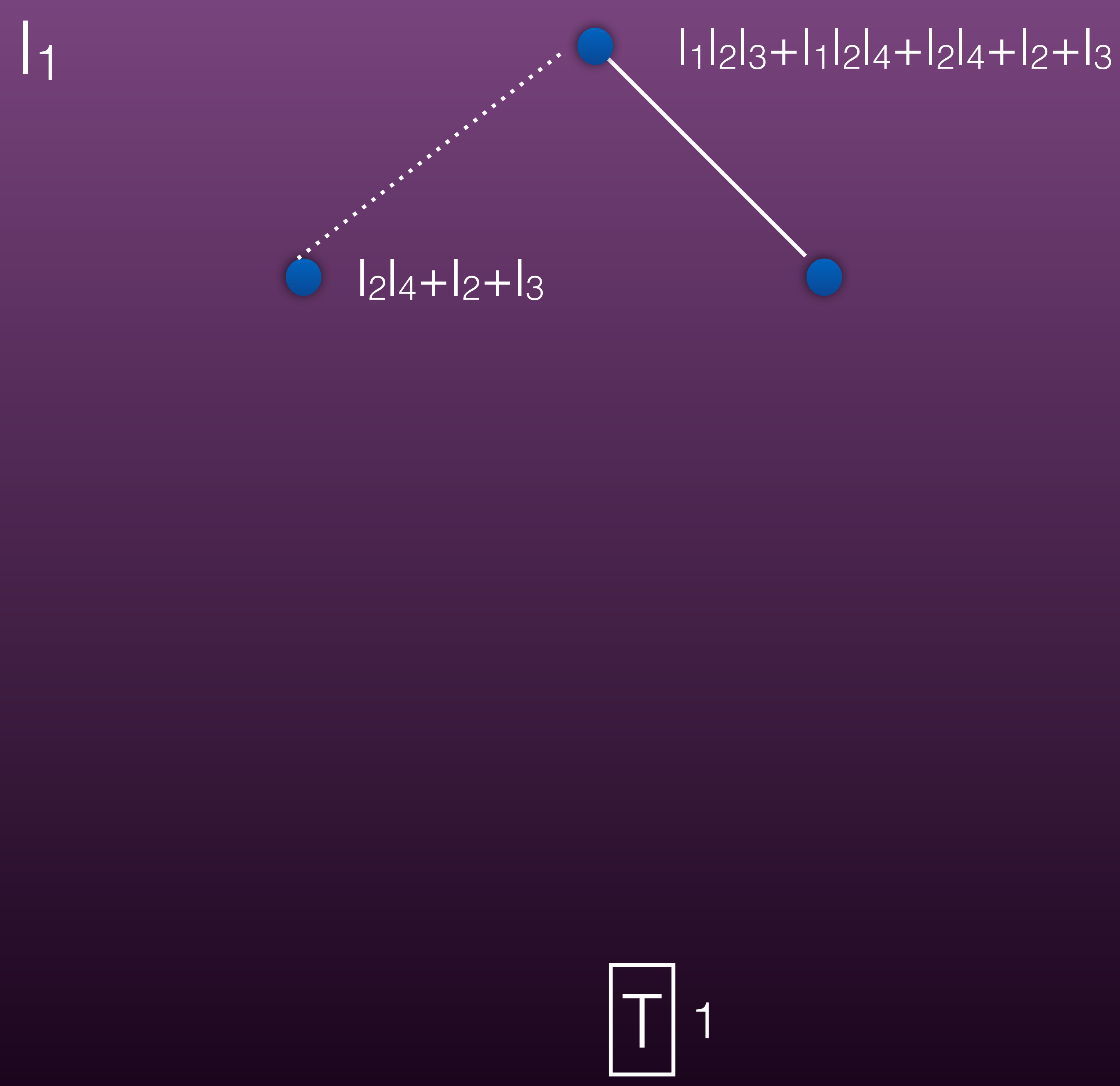
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



T 1

Example

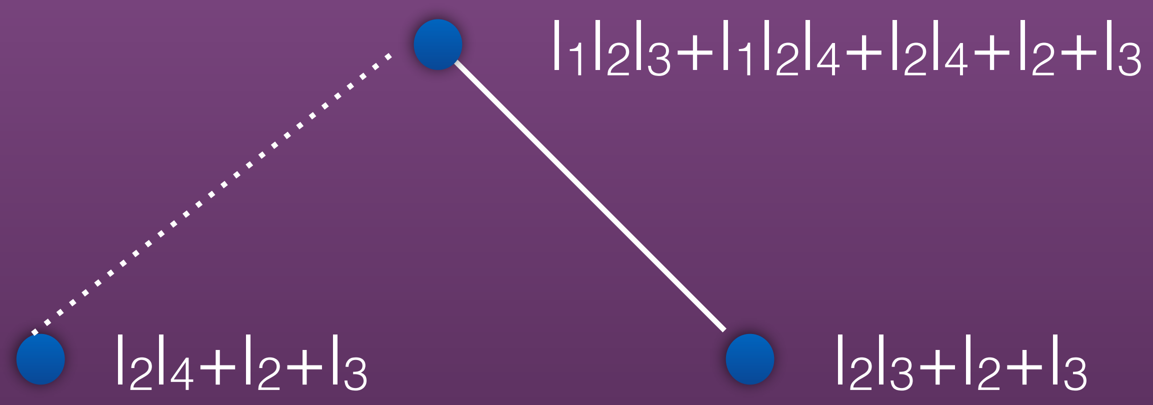
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



Example

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

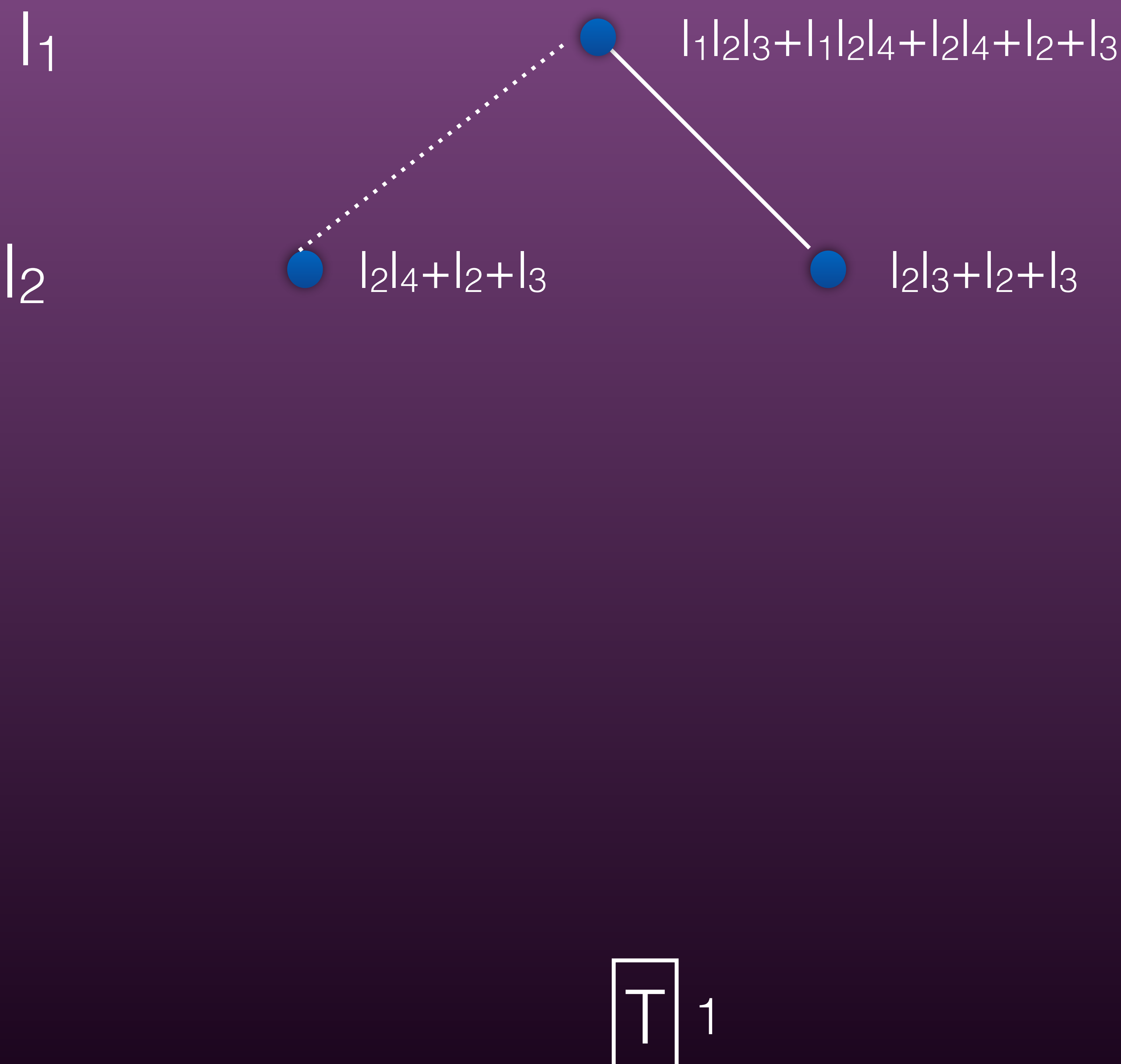
l_1



T 1

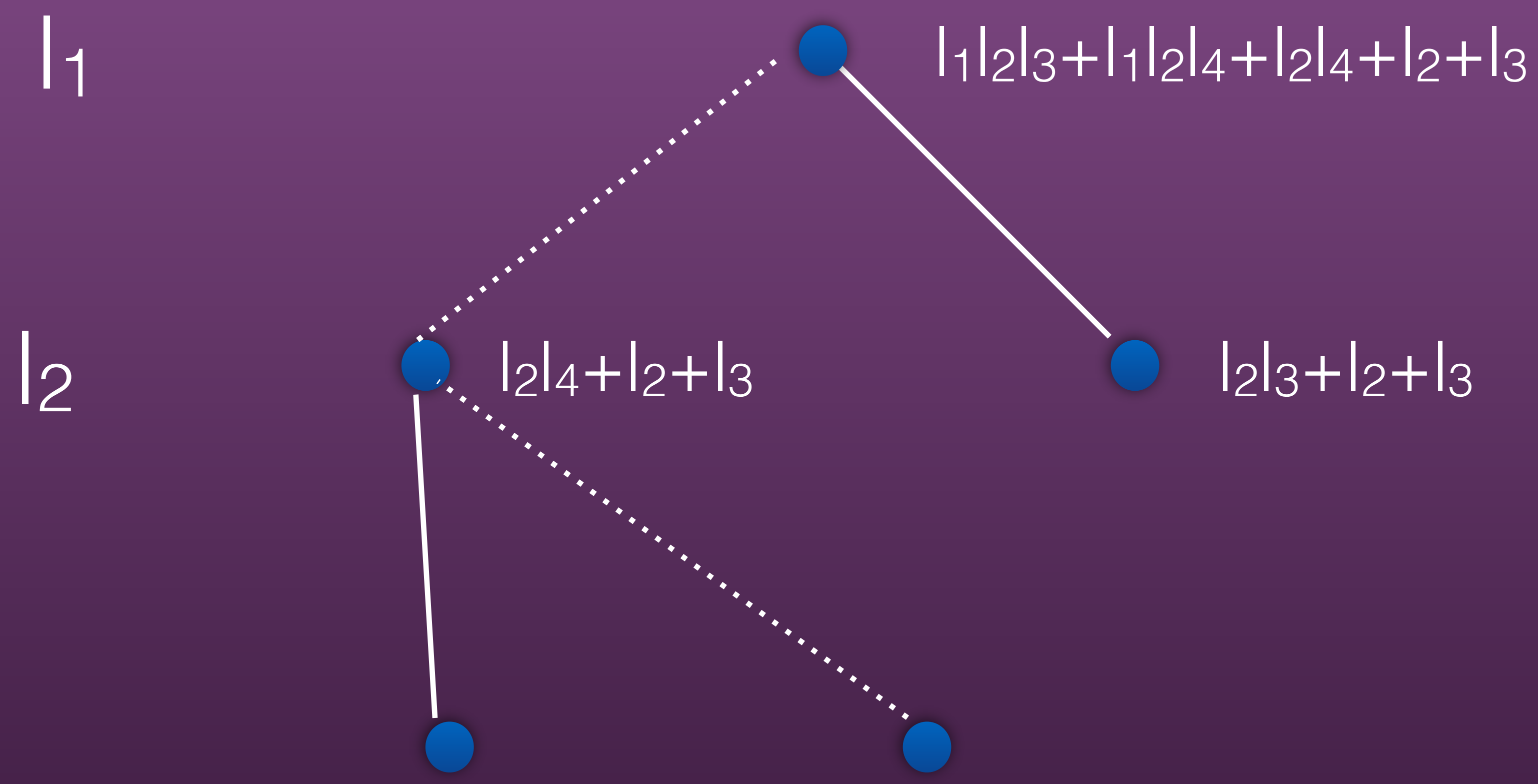
Example

$\underline{f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1}$



Example

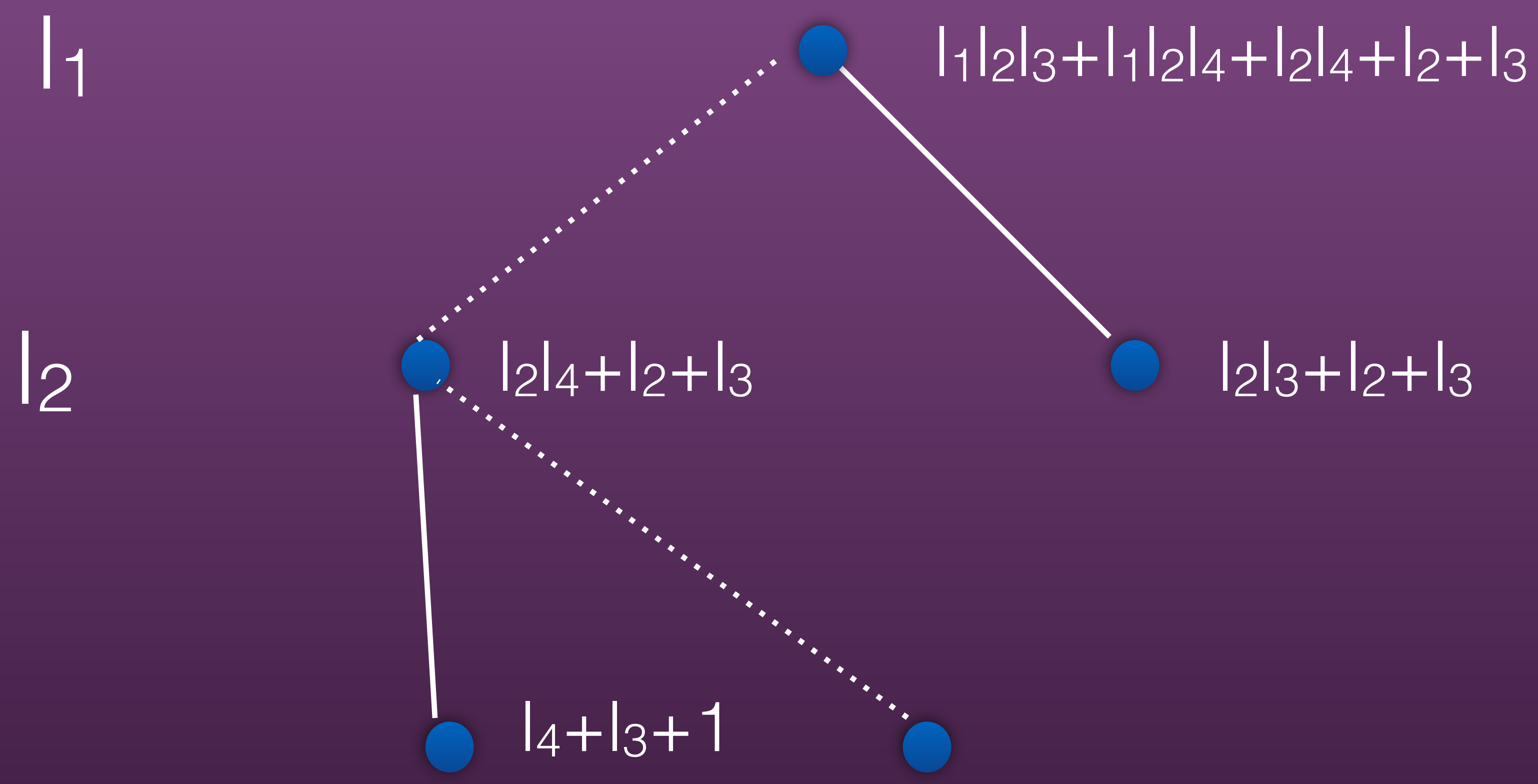
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



T 1

Example

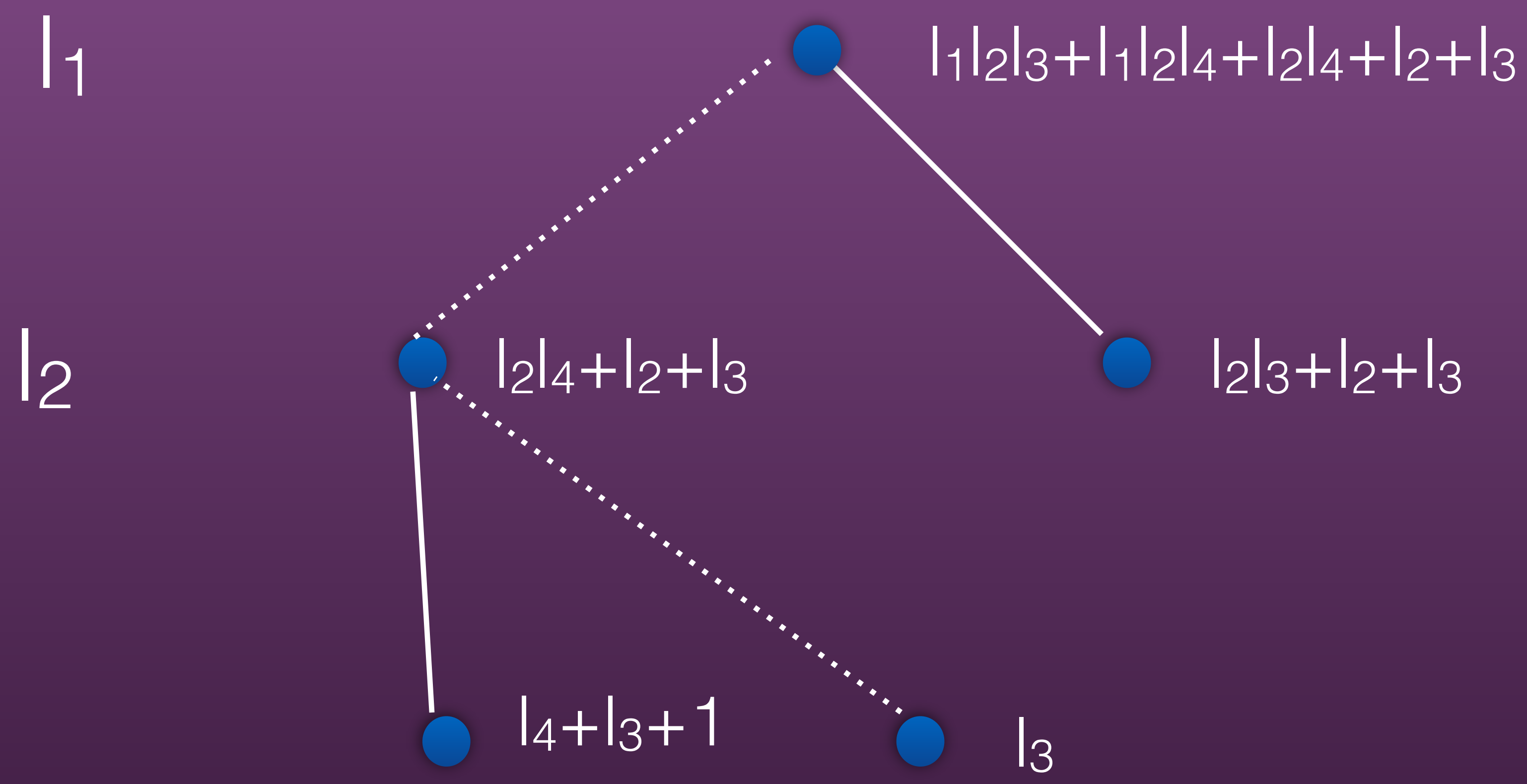
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



T 1

Example

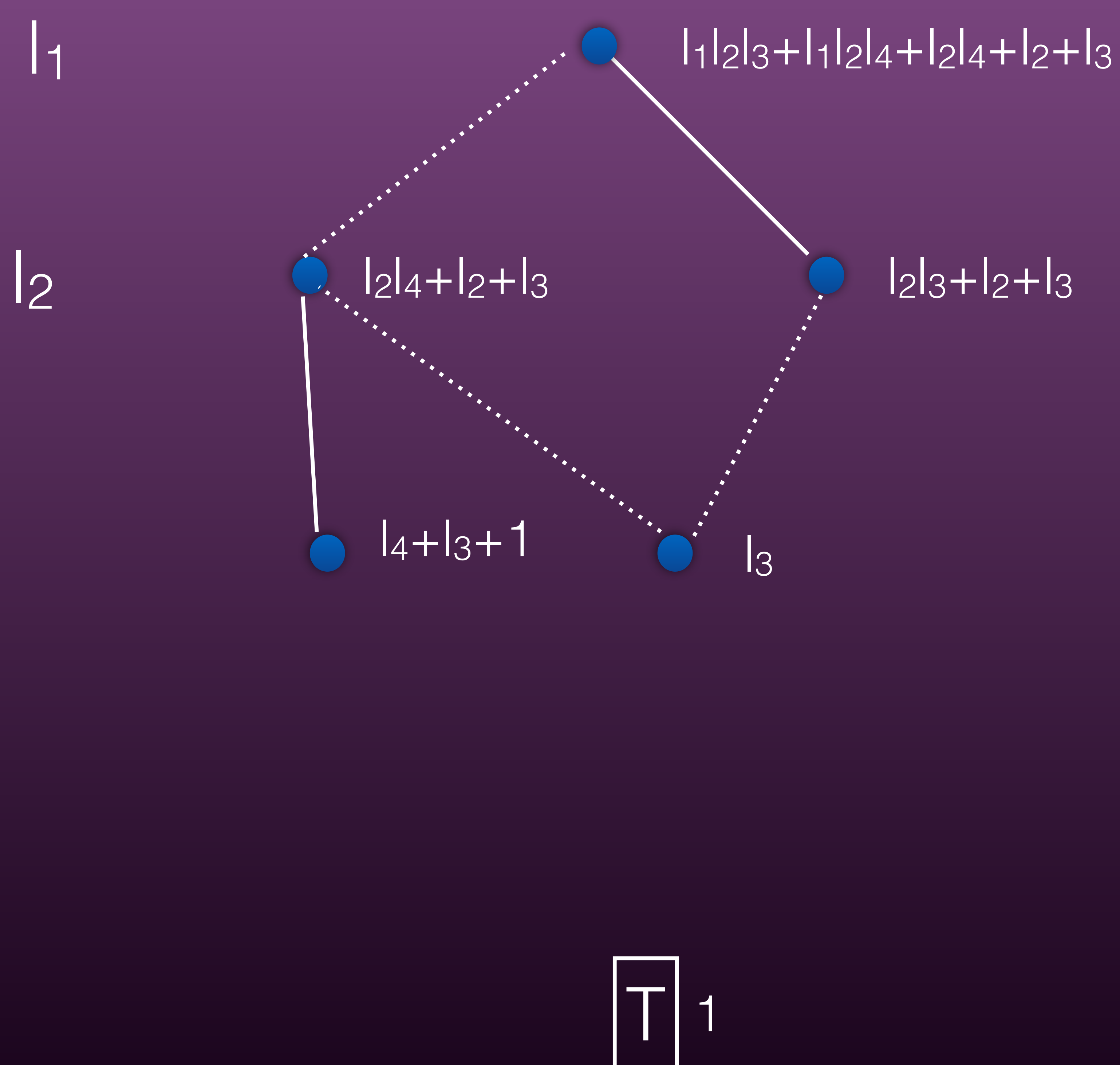
$f(l_1,l_2,l_3,l_4) = l_1l_2l_3+l_1l_2l_4+l_2l_4+l_2+l_3 = 1$



T 1

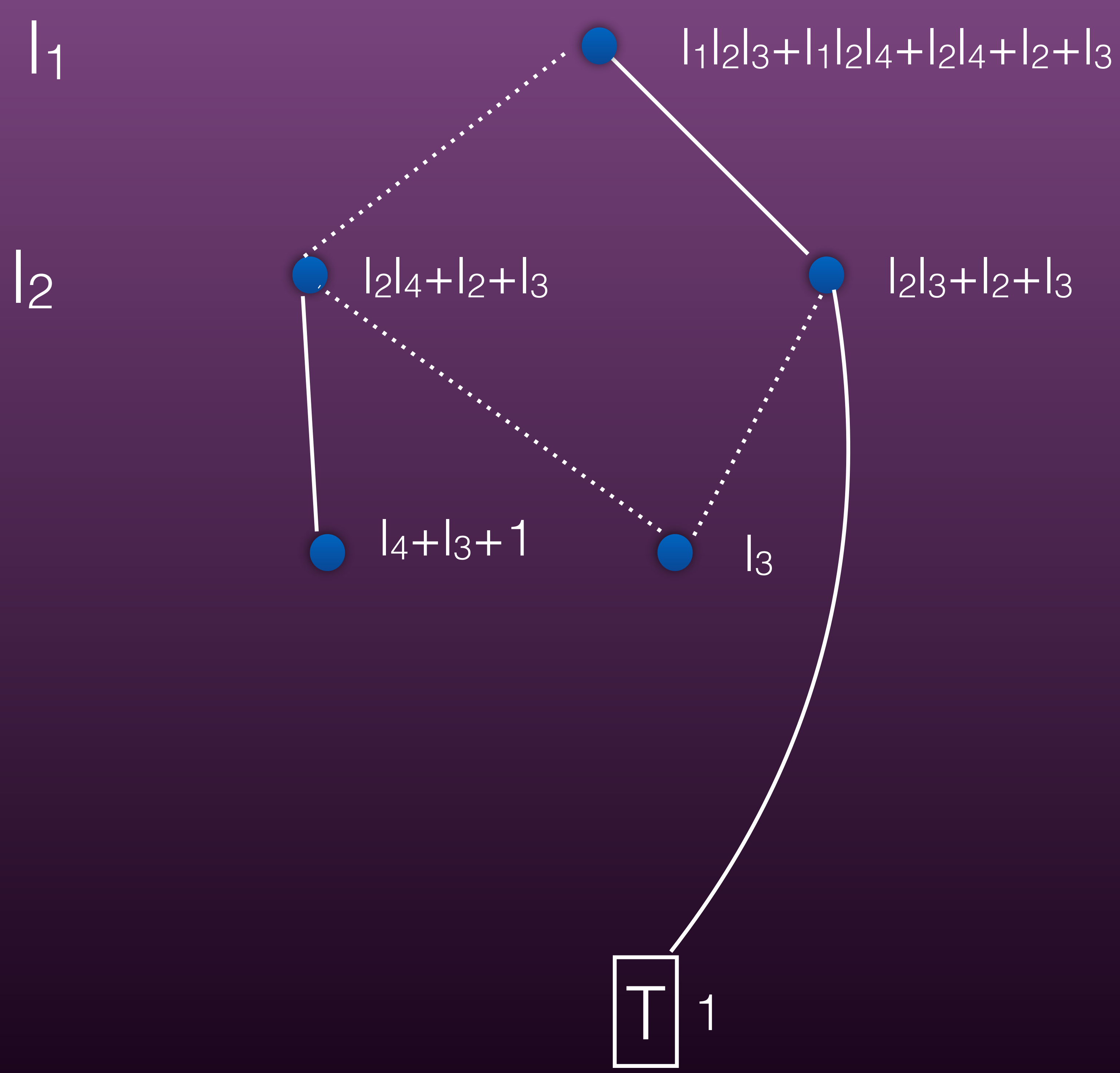
Example

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



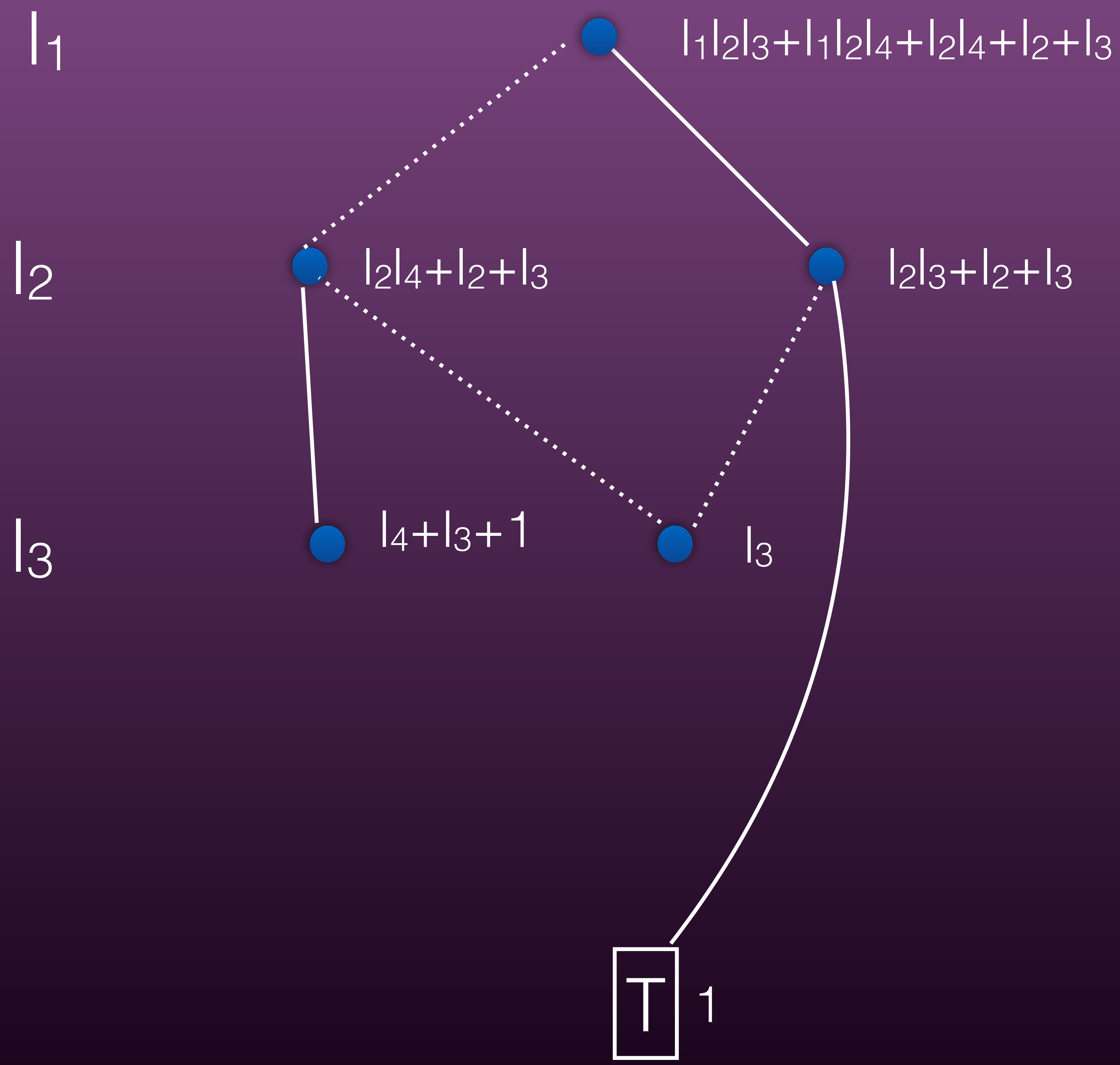
Example

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



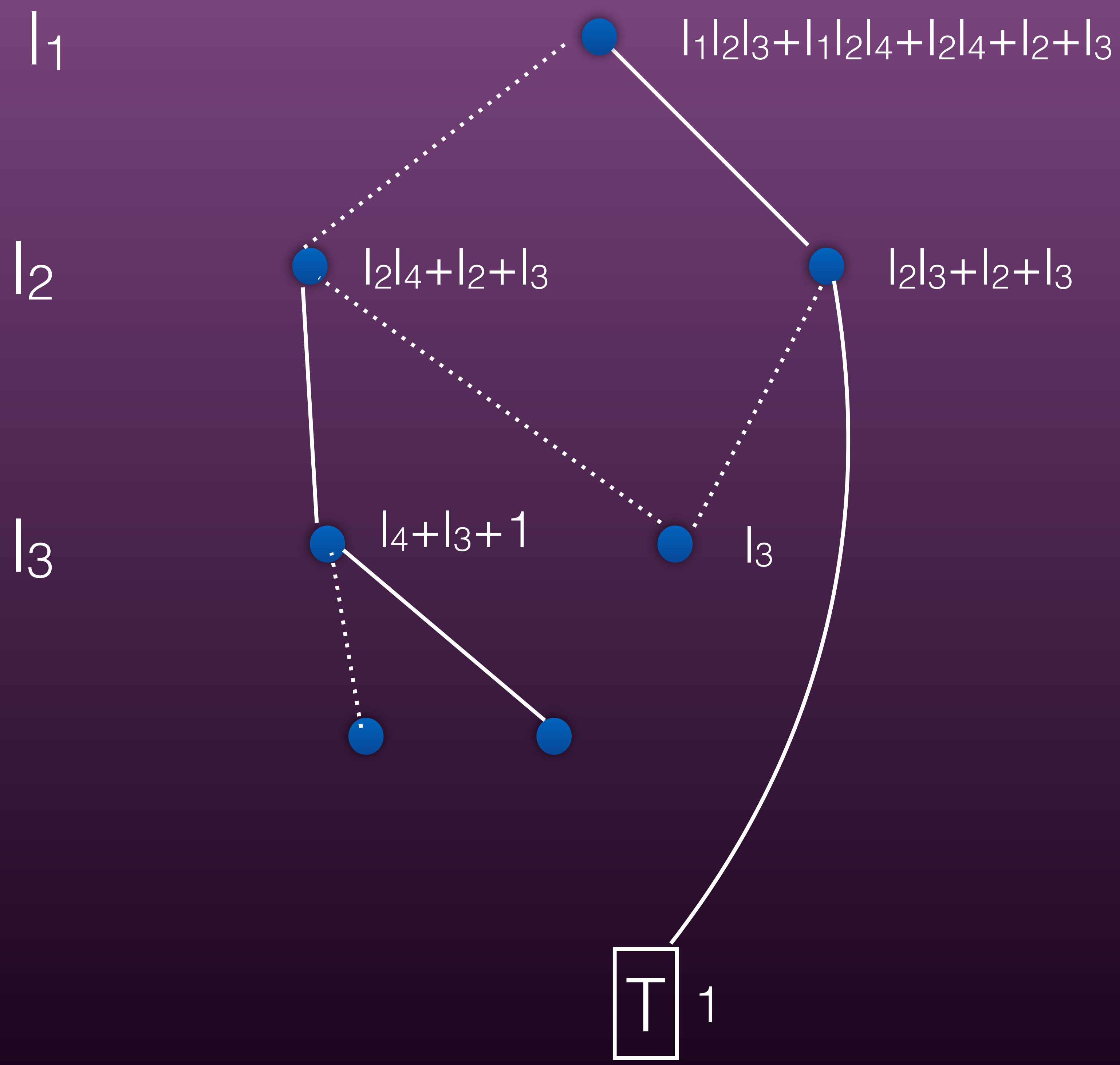
Example

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



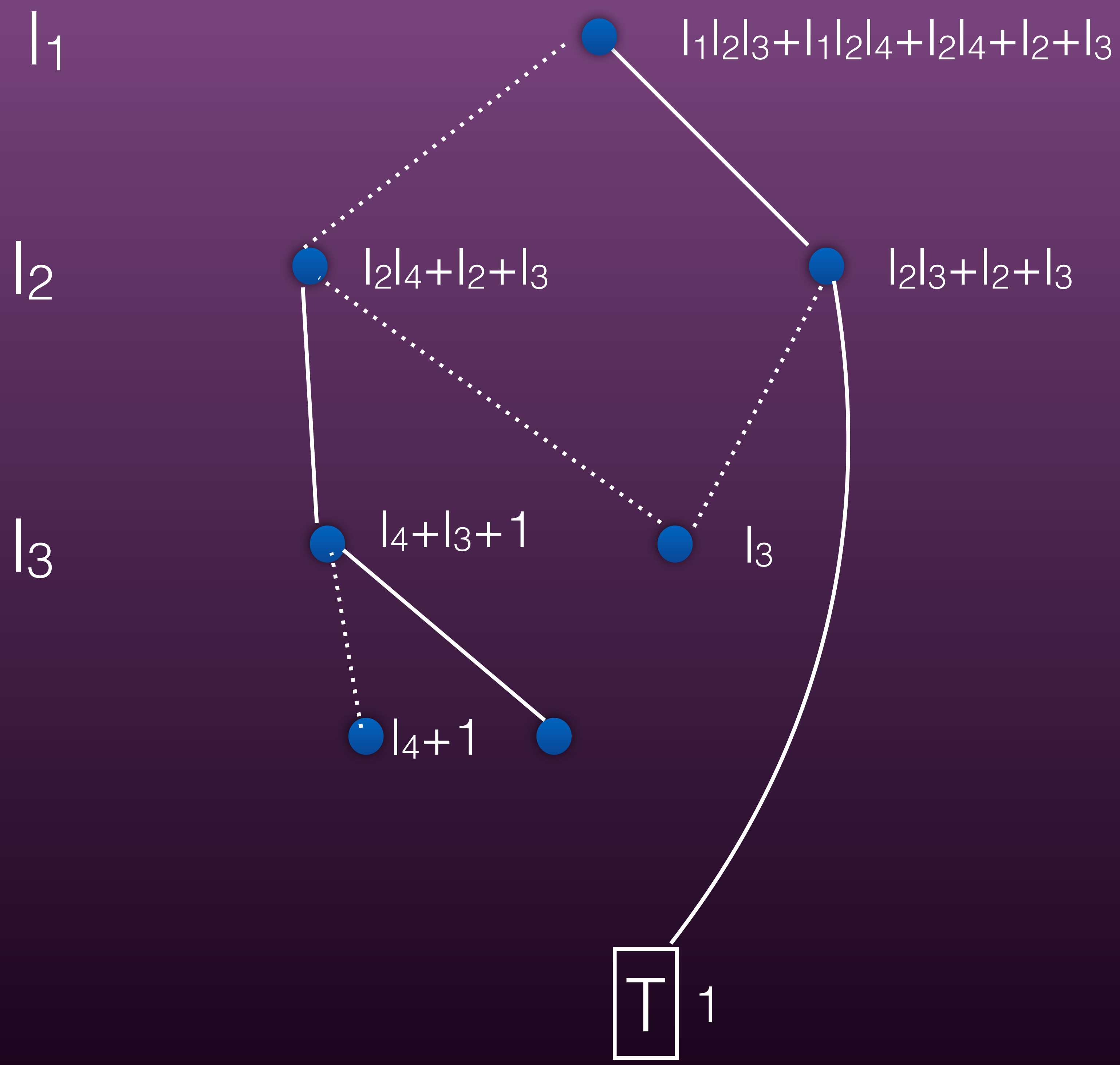
Example

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



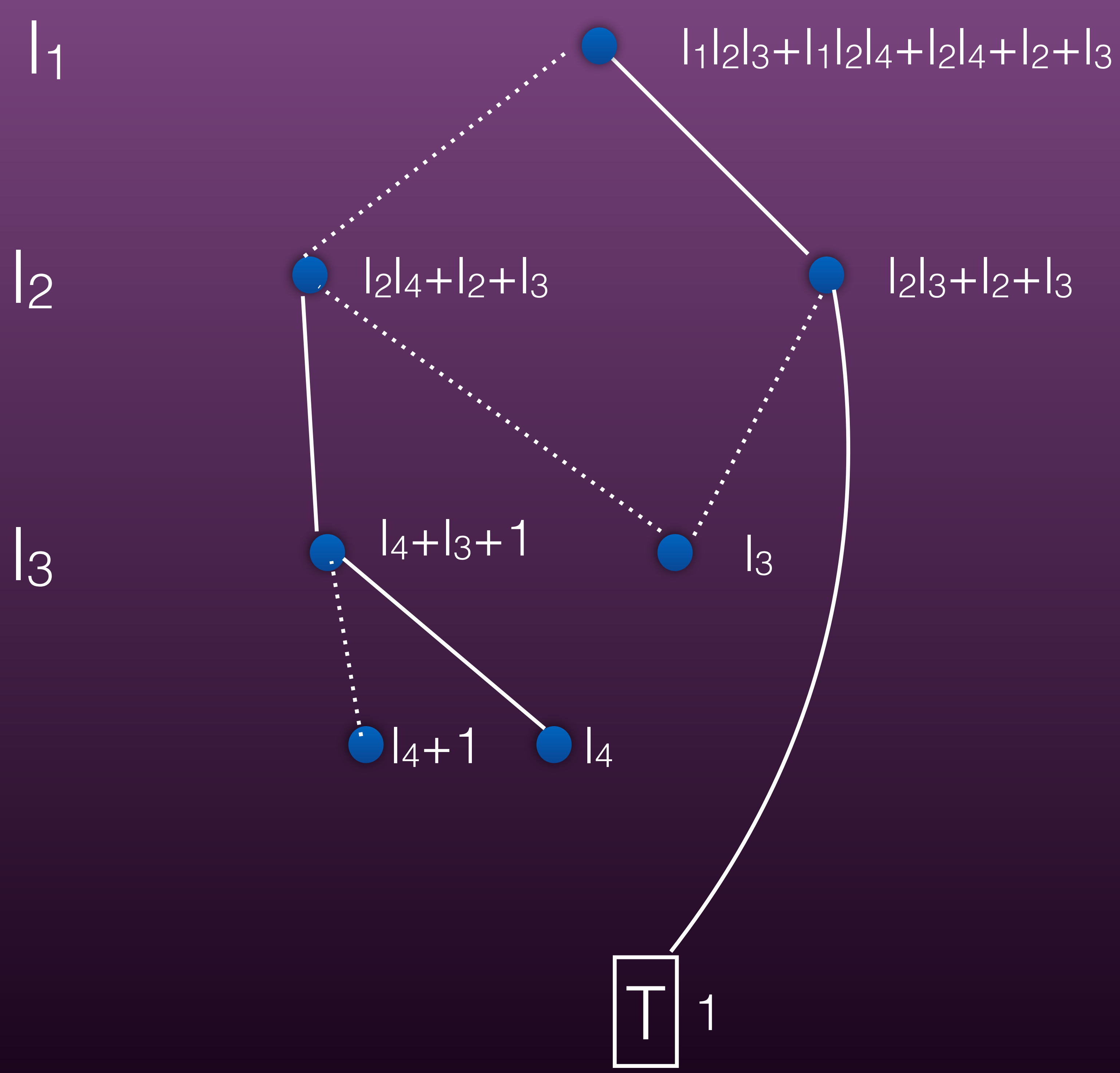
Example

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



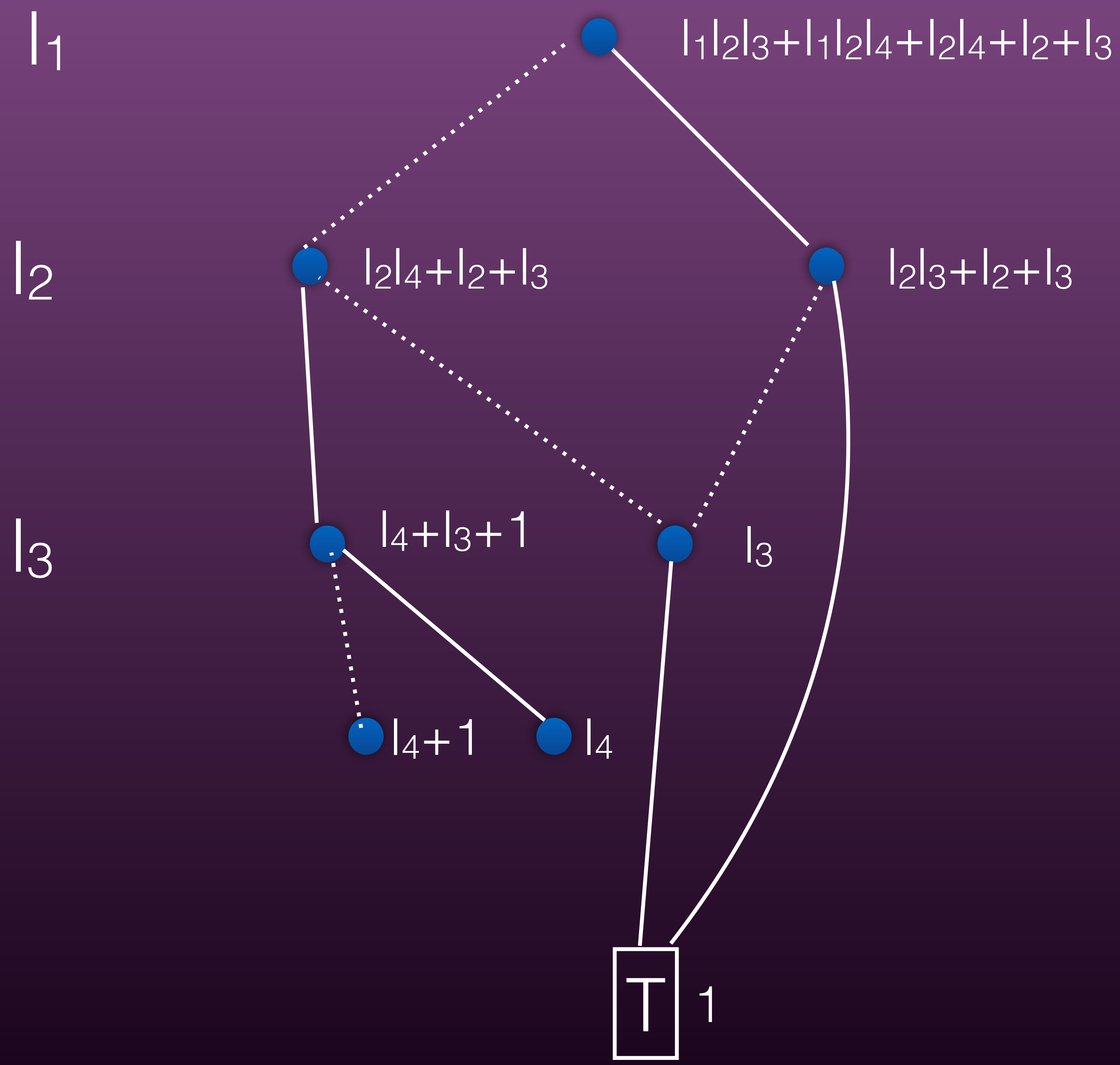
Example

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



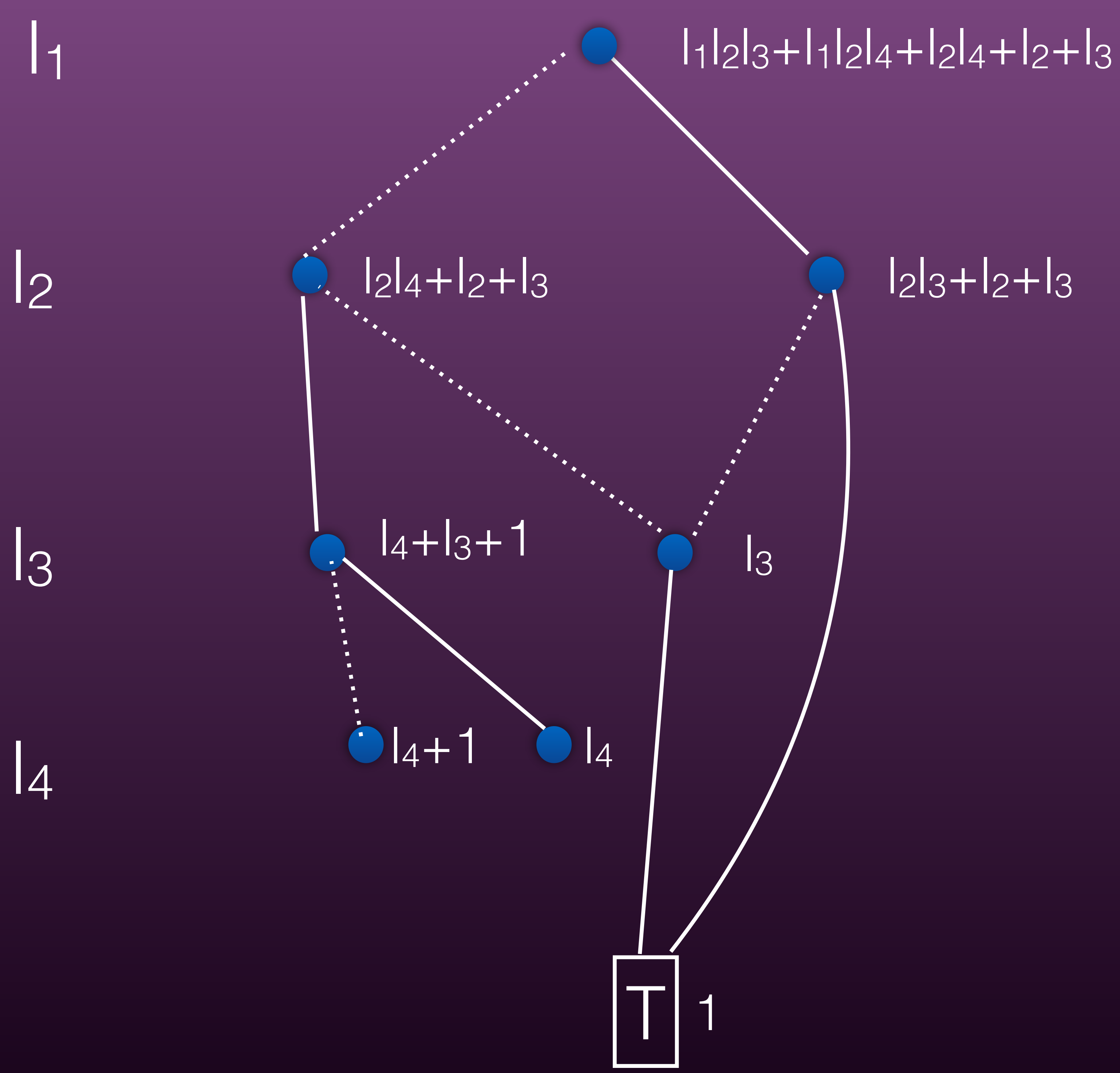
Example

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



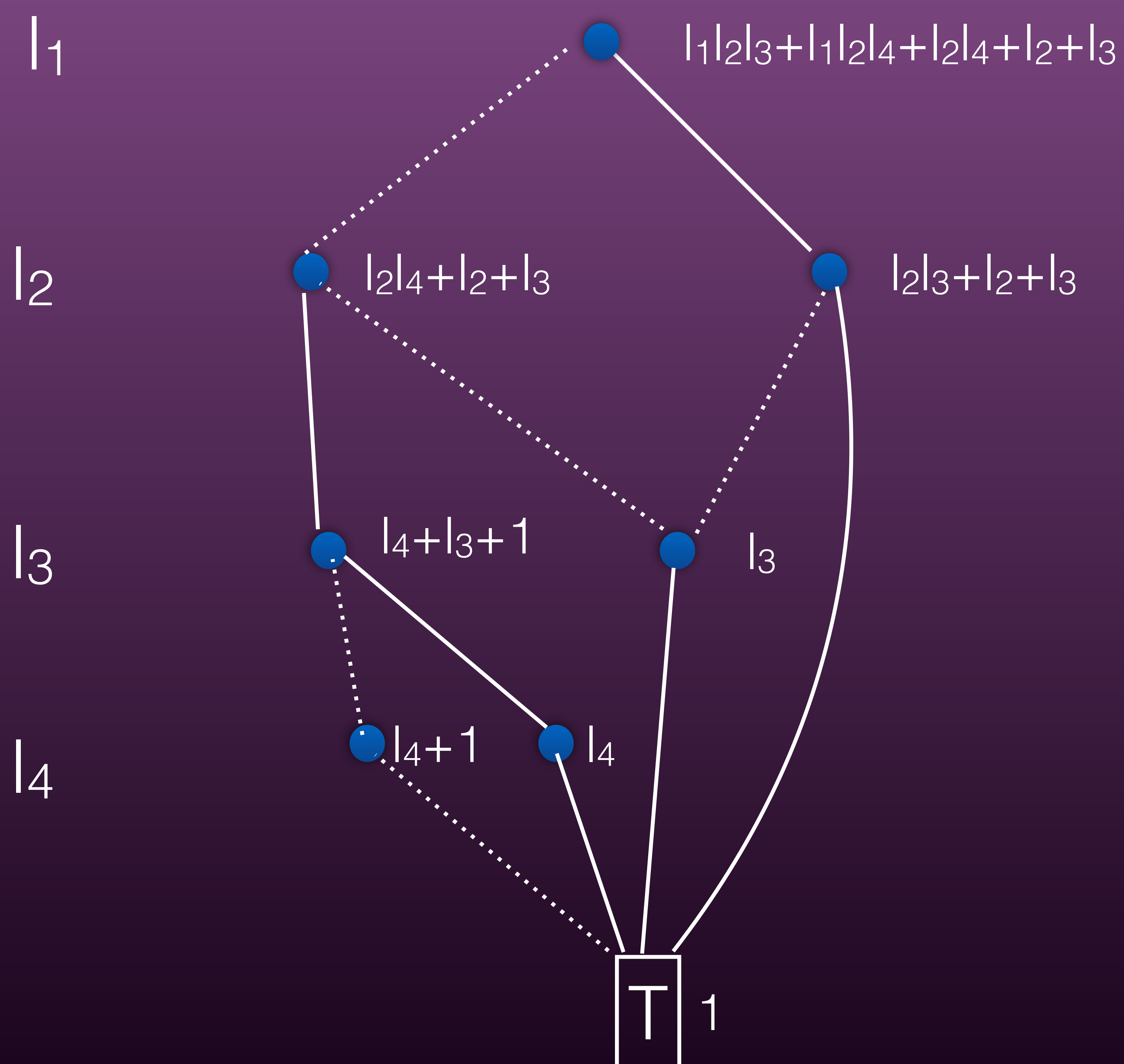
Example

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



Example

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



Constructing system

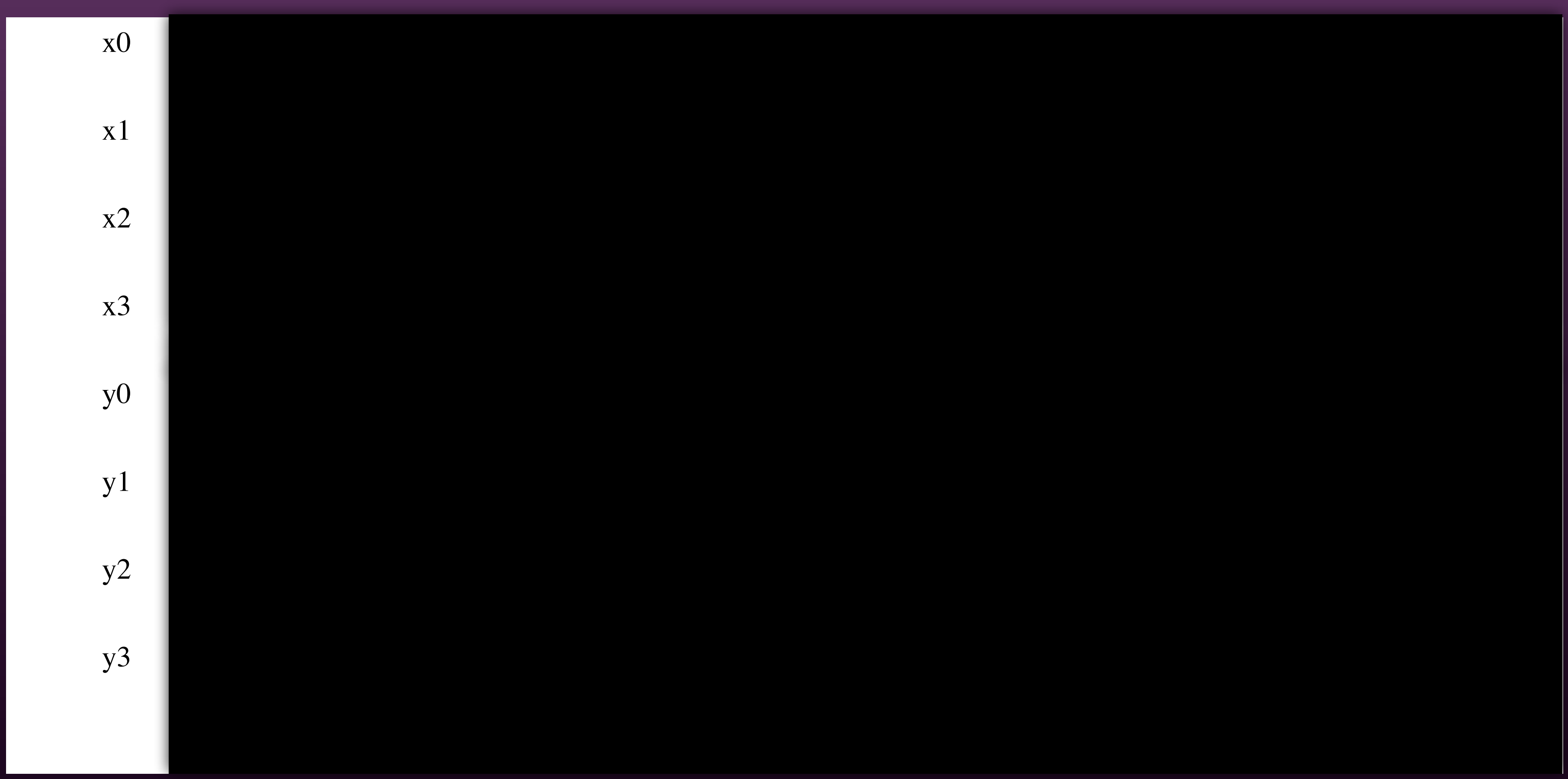
$$f_1(l_{11},\ldots,l_{1k}) = 1$$
$$f_2(l_{21},\ldots,l_{2k}) = 1$$
$$\ldots$$
$$f_n(l_{n1},\ldots,l_{nk}) = 1$$

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

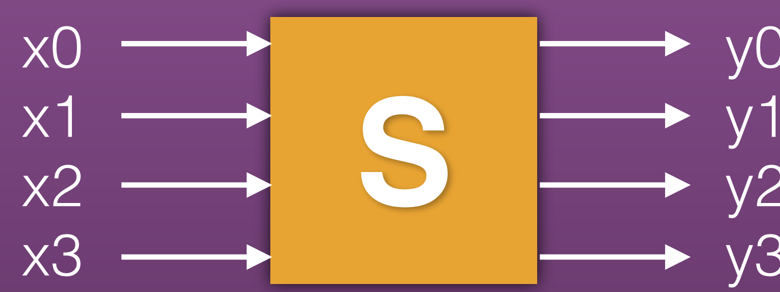
- Build one BDD for each f_i (or non-linear component)
- Set of BDDs = representation of equation system (cryptographic primitive)

BDD representing S-box

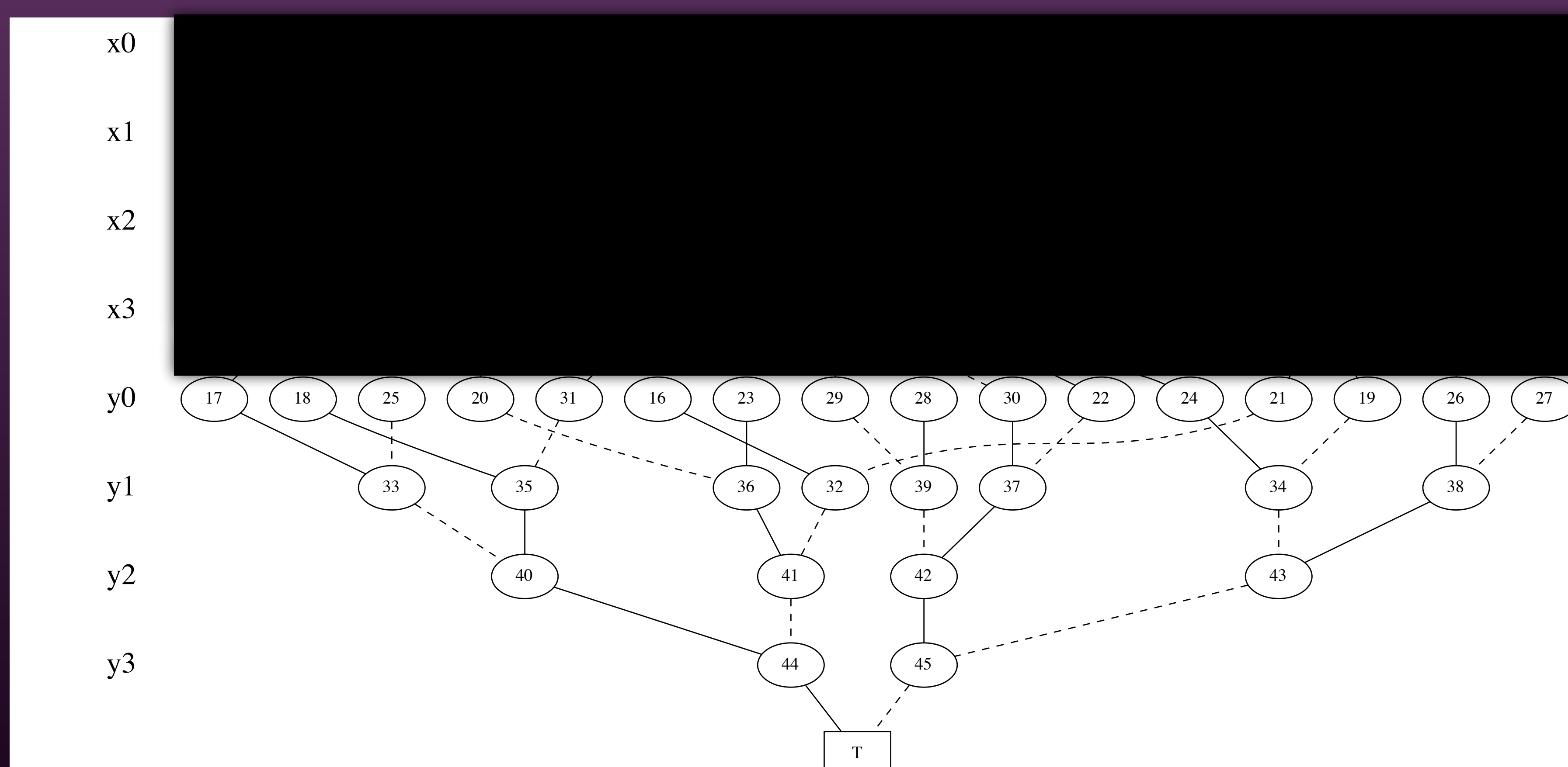
x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
y	5	C	8	F	9	7	2	B	6	A	0	D	E	4	3	1



BDD representing S-box

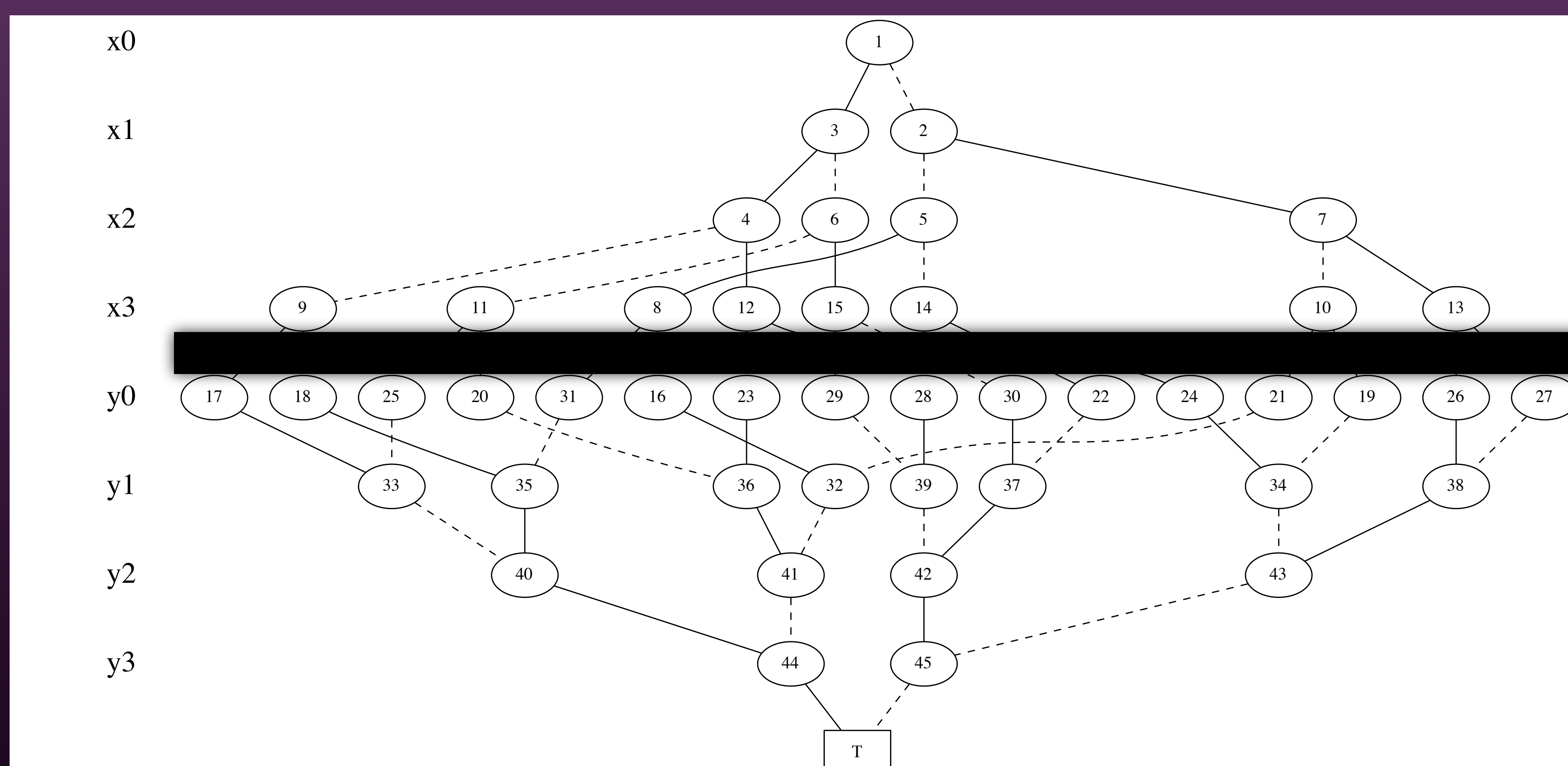


x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
y	5	C	8	F	9	7	2	B	6	A	0	D	E	4	3	1



BDD representing S-box

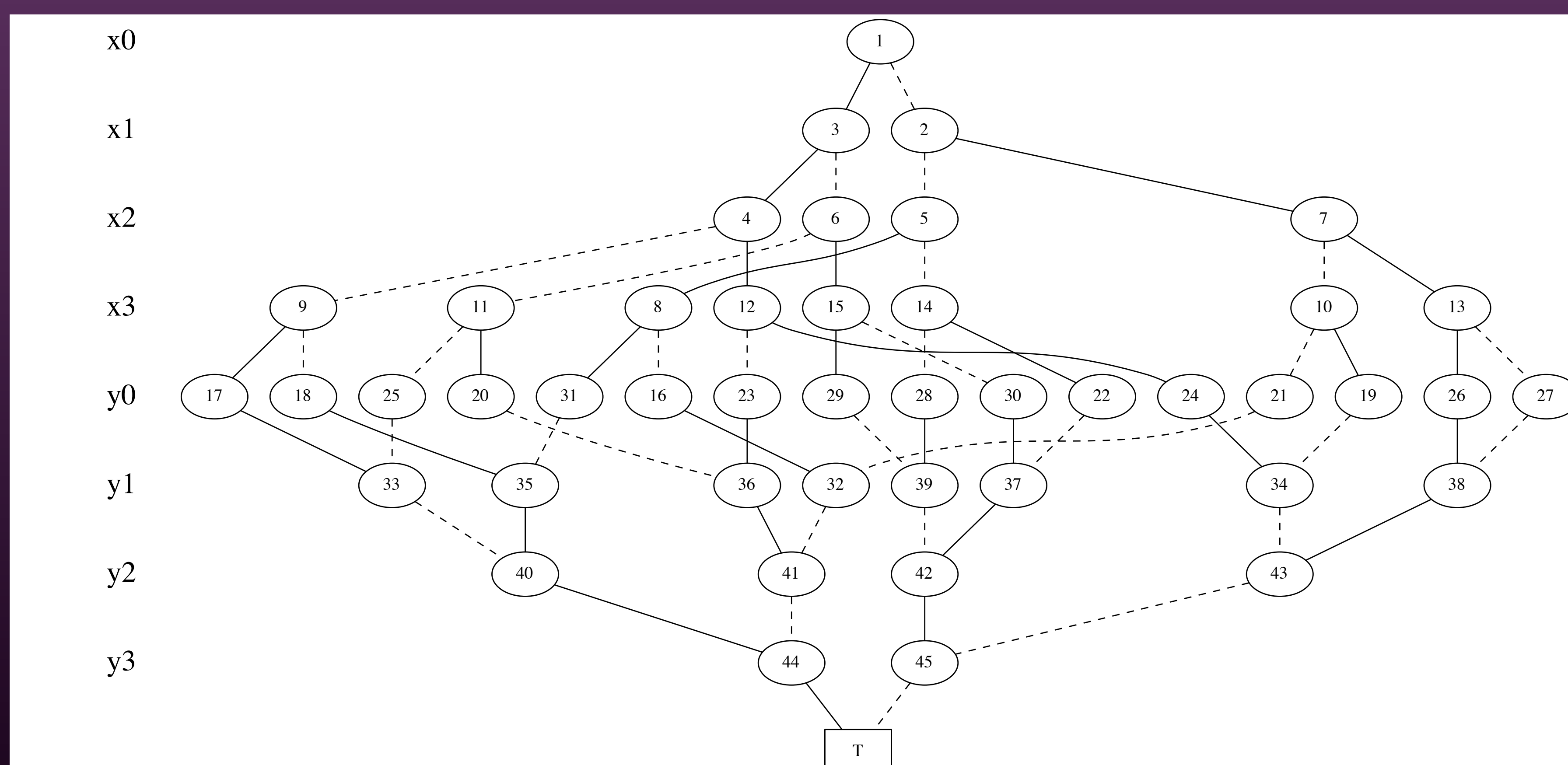
x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
y	5	C	8	F	9	7	2	B	6	A	0	D	E	4	3	1



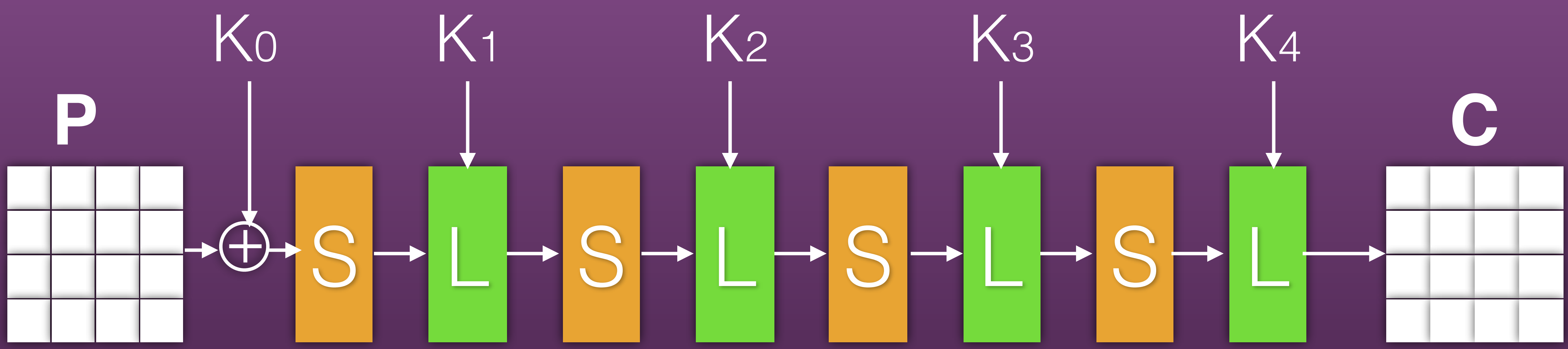
BDD representing S-box

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

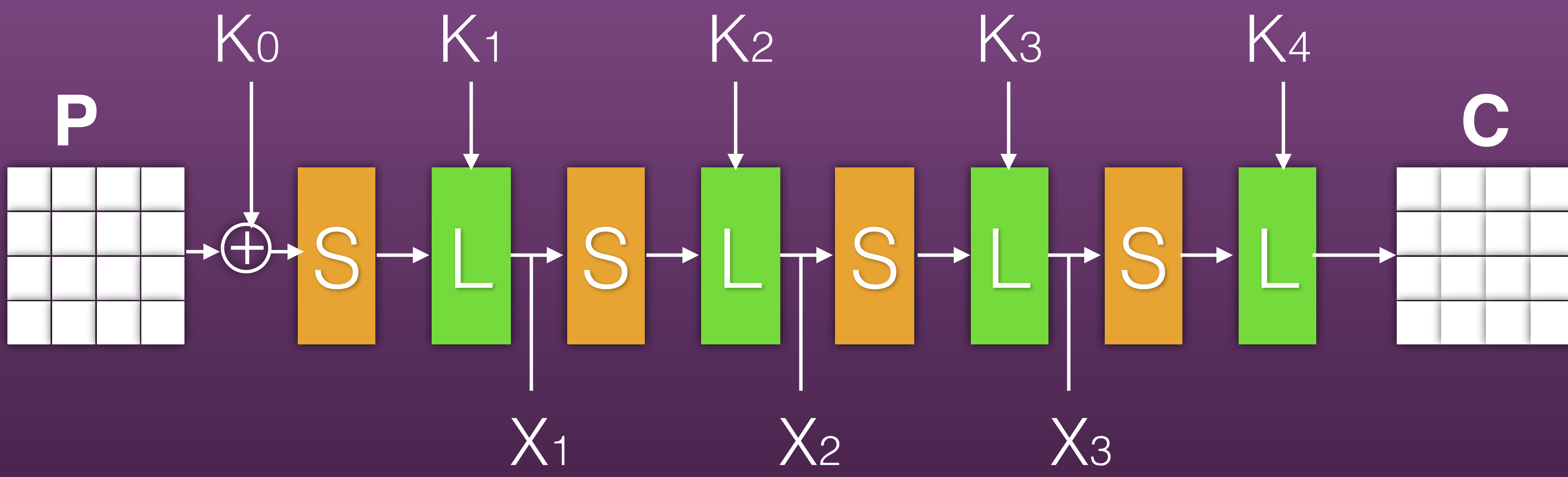
$\mathbf{y}=S(\mathbf{x})$



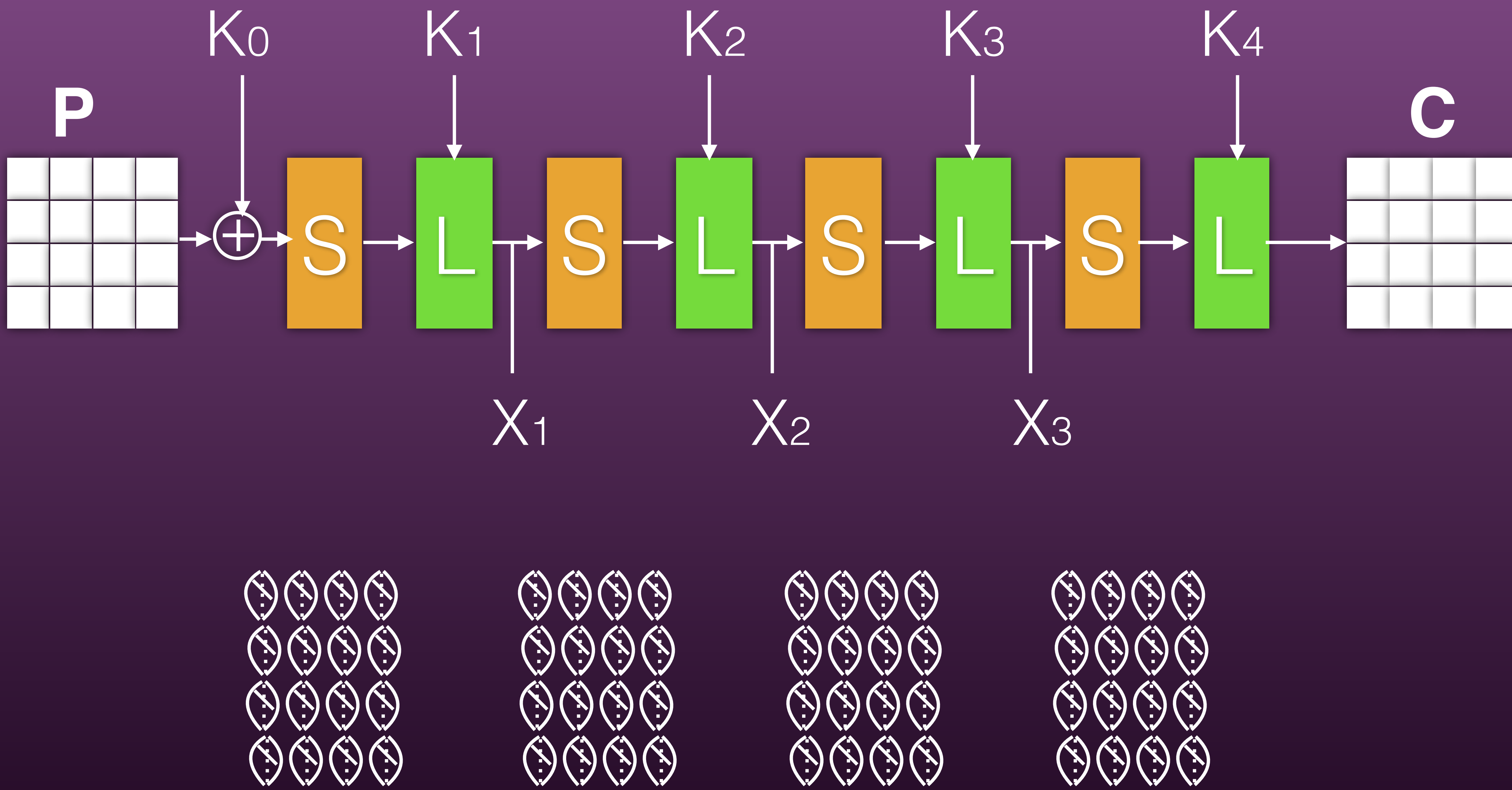
Example - 4 round AES



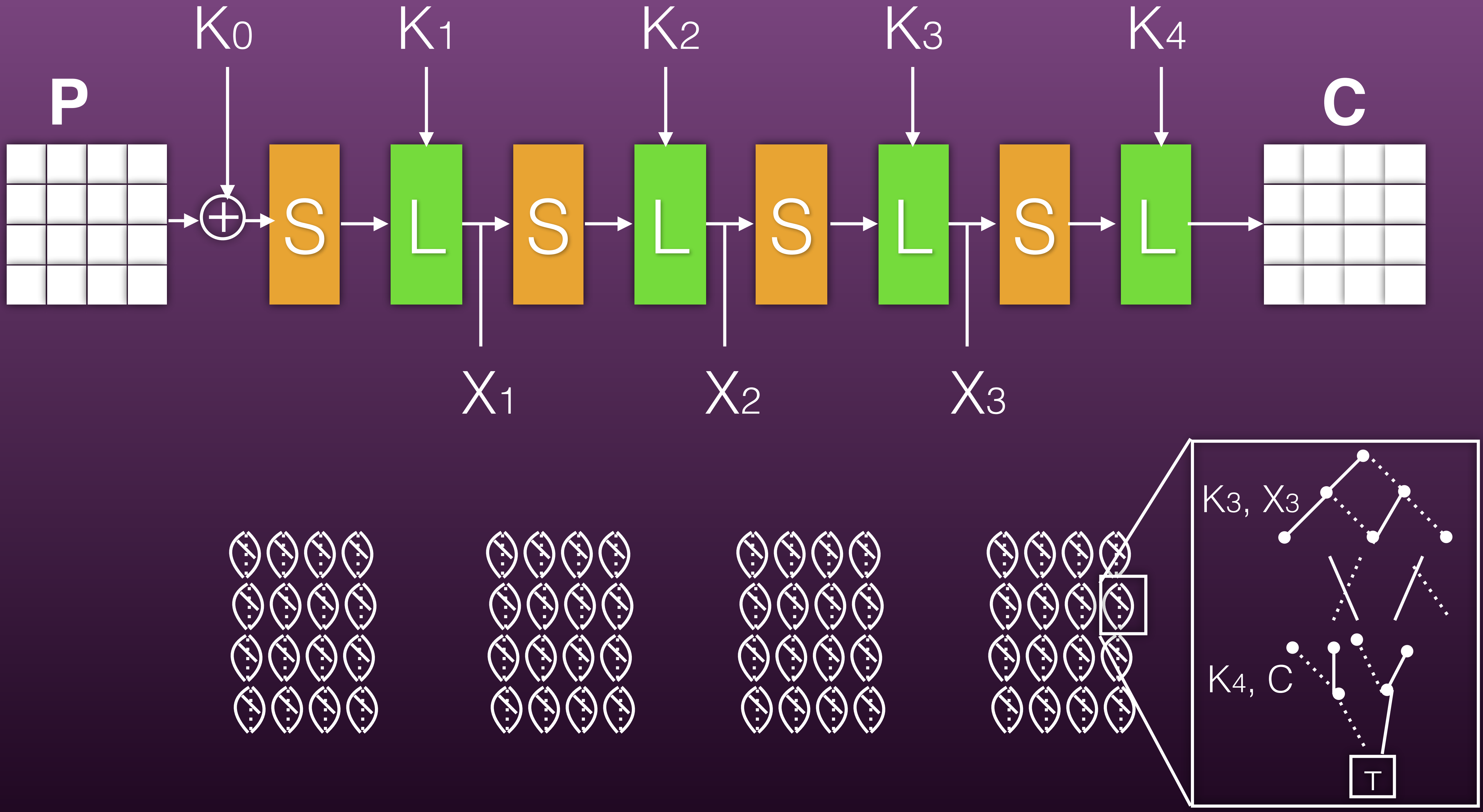
Example - 4 round AES



Example - 4 round AES



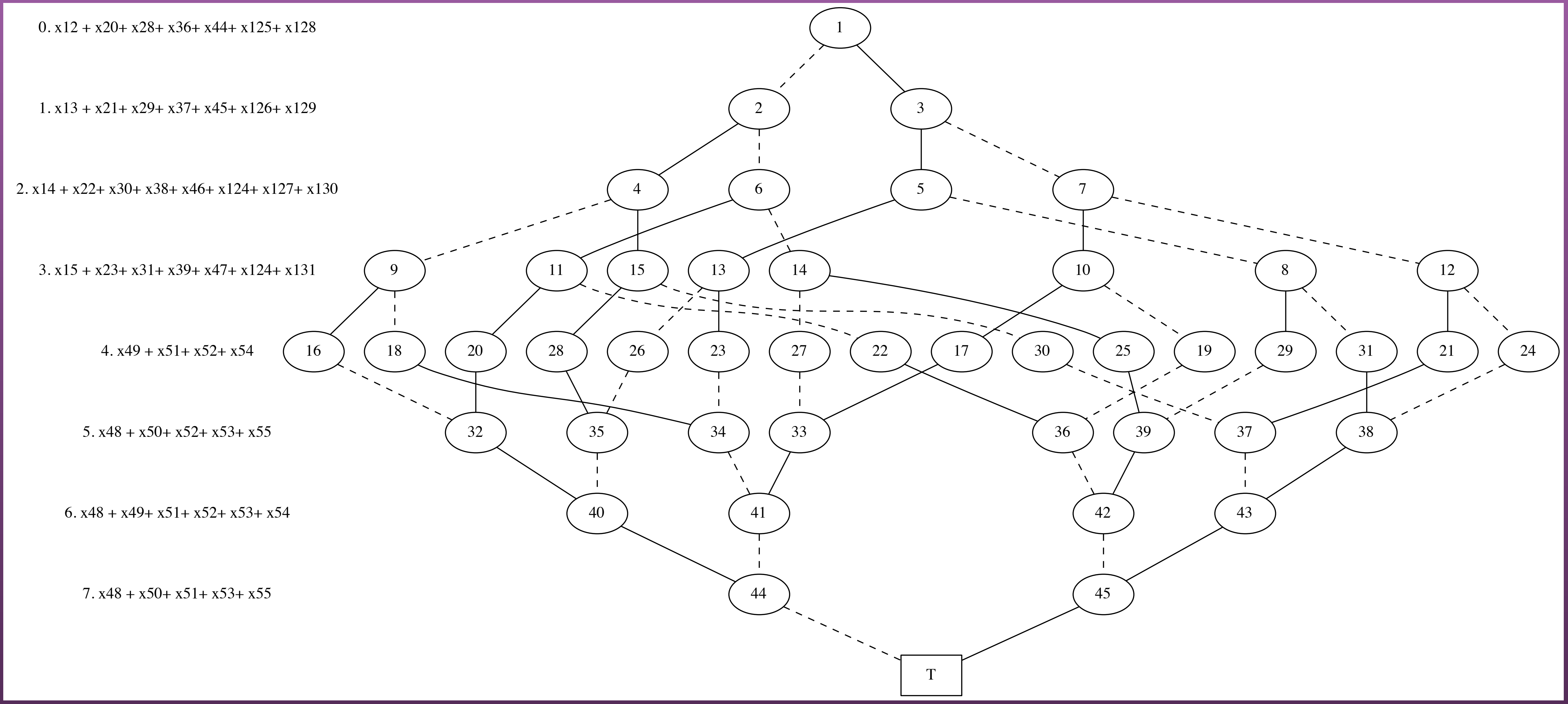
Example - 4 round AES

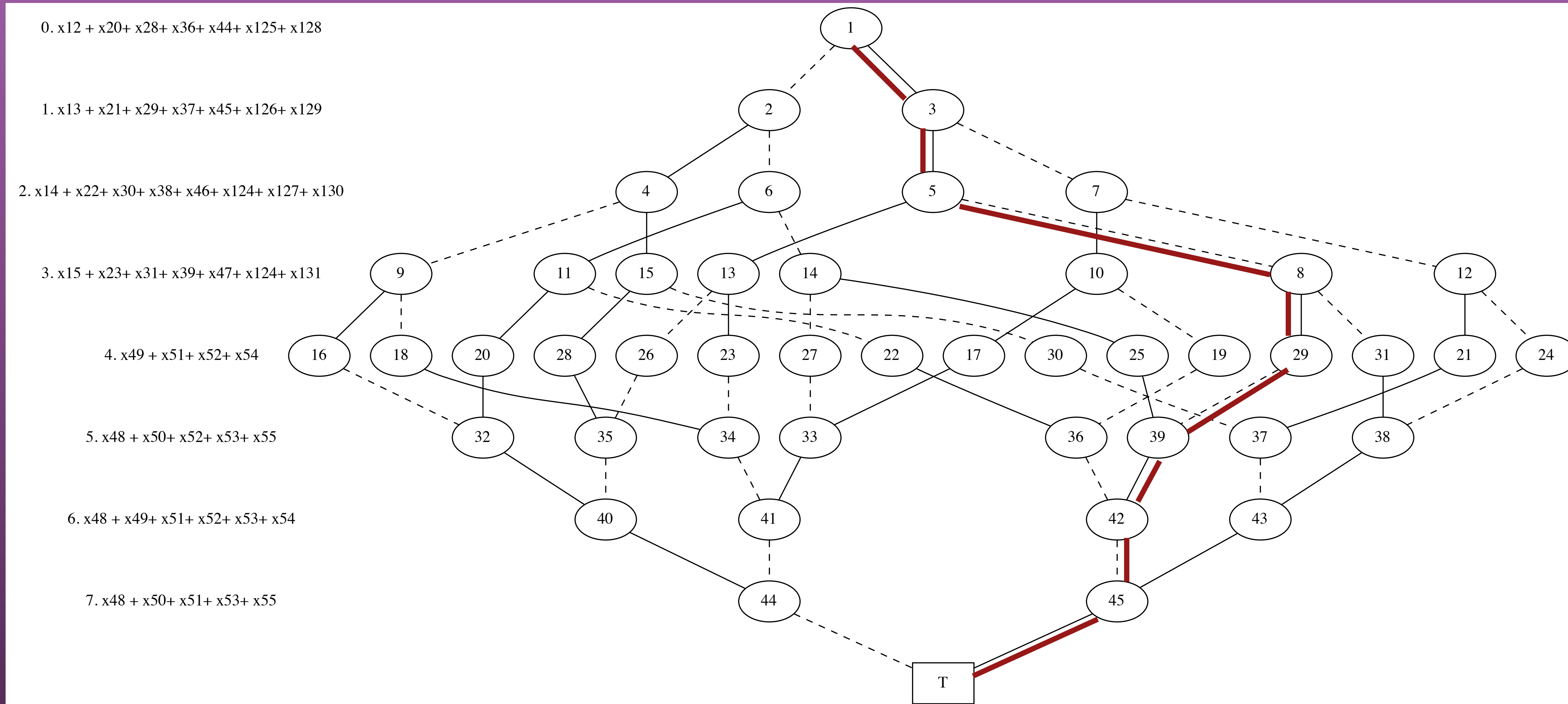


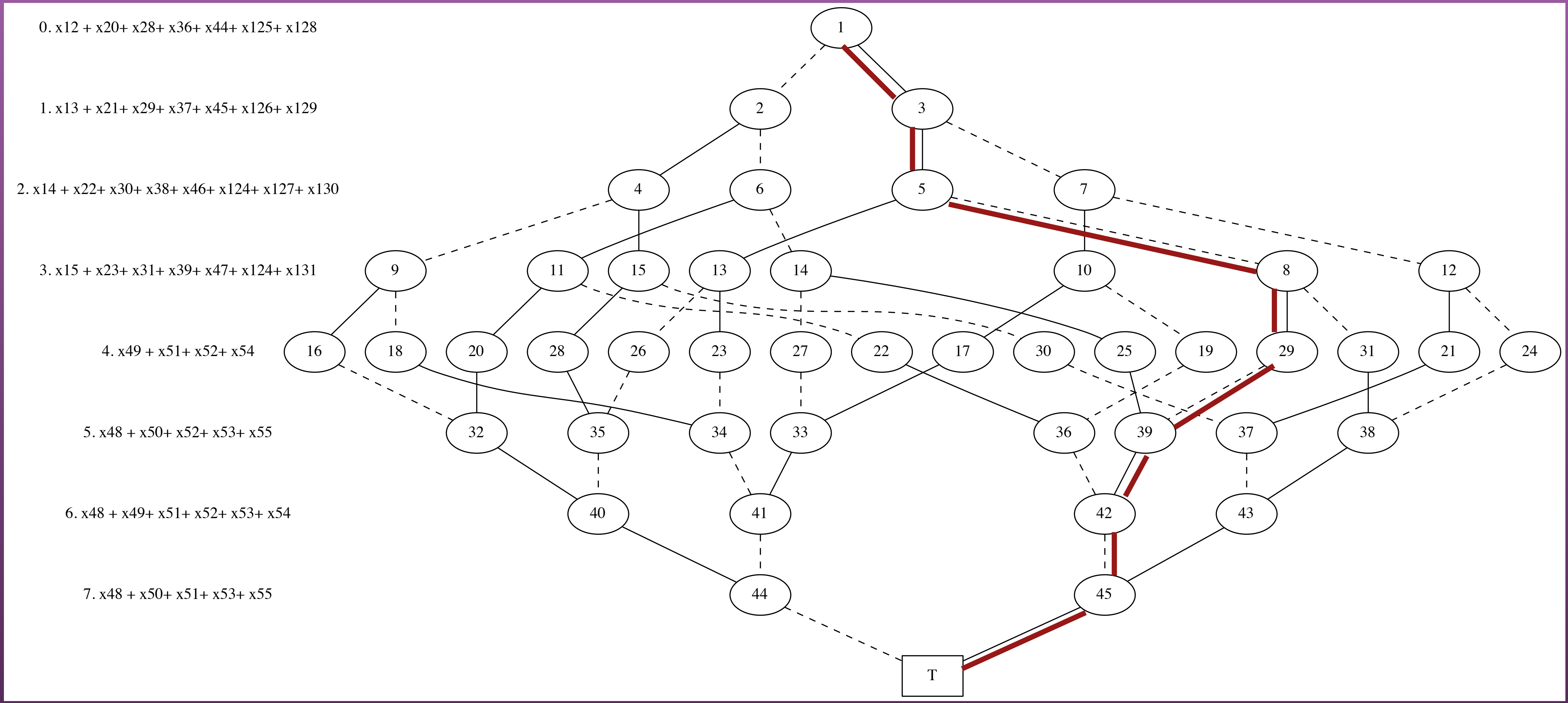
Solving BDD systems

Paths = valid assignments

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


0. x12 + x20+ x28+ x36+ x44+ x125+ x128
1. x13 + x21+ x29+ x37+ x45+ x126+ x129
2. x14 + x22+ x30+ x38+ x46+ x124+ x127+ x130
3. x15 + x23+ x31+ x39+ x47+ x124+ x131
4. x49 + x51+ x52+ x54
5. x48 + x50+ x52+ x53+ x55
6. x48 + x49+ x51+ x52+ x53+ x54
7. x48 + x50+ x51+ x53+ x55
1
2
3
4
6
5
7
9
11
15
13
14
10
8
12
16
18
20
28
26
23
27
22
17
30
25
19
29
31
21
24
32
35
34
33
36
39
37
38
40
41
42
43
44
45
T


0. x12 + x20+ x28+ x36+ x44+ x125+ x128
1. x13 + x21+ x29+ x37+ x45+ x126+ x129
2. x14 + x22+ x30+ x38+ x46+ x124+ x127+ x130
3. x15 + x23+ x31+ x39+ x47+ x124+ x131
4. x49 + x51+ x52+ x54
5. x48 + x50+ x52+ x53+ x55
6. x48 + x49+ x51+ x52+ x53+ x54
7. x48 + x50+ x51+ x53+ x55
1
2
3
4
6
5
7
9
11
15
13
14
10
8
12
16
18
20
28
26
23
27
22
17
30
25
19
29
31
21
24
32
35
34
33
36
39
37
38
40
41
42
43
44
45
T



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

Naive solving attempt

- Select a path from each BDD

Naive solving attempt

- Select a path from each BDD
- Collect linear systems from each BDD into one big linear system

Naive solving attempt

- Select a path from each BDD
- Collect linear systems from each BDD into one big linear system
- Solve big linear system

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

Naive failure

- Big linear system is overdefined, with lots of dependencies among linear combinations

Naive failure

- Big linear system is overdefined, with lots of dependencies among linear combinations
- Selected paths will, in all likelihood, lead to an inconsistent system

Naive failure

- Big linear system is overdefined, with lots of dependencies among linear combinations
- 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 linear combinations of two adjacent levels
  - Add (xor) the linear combinations of two adjacent levels

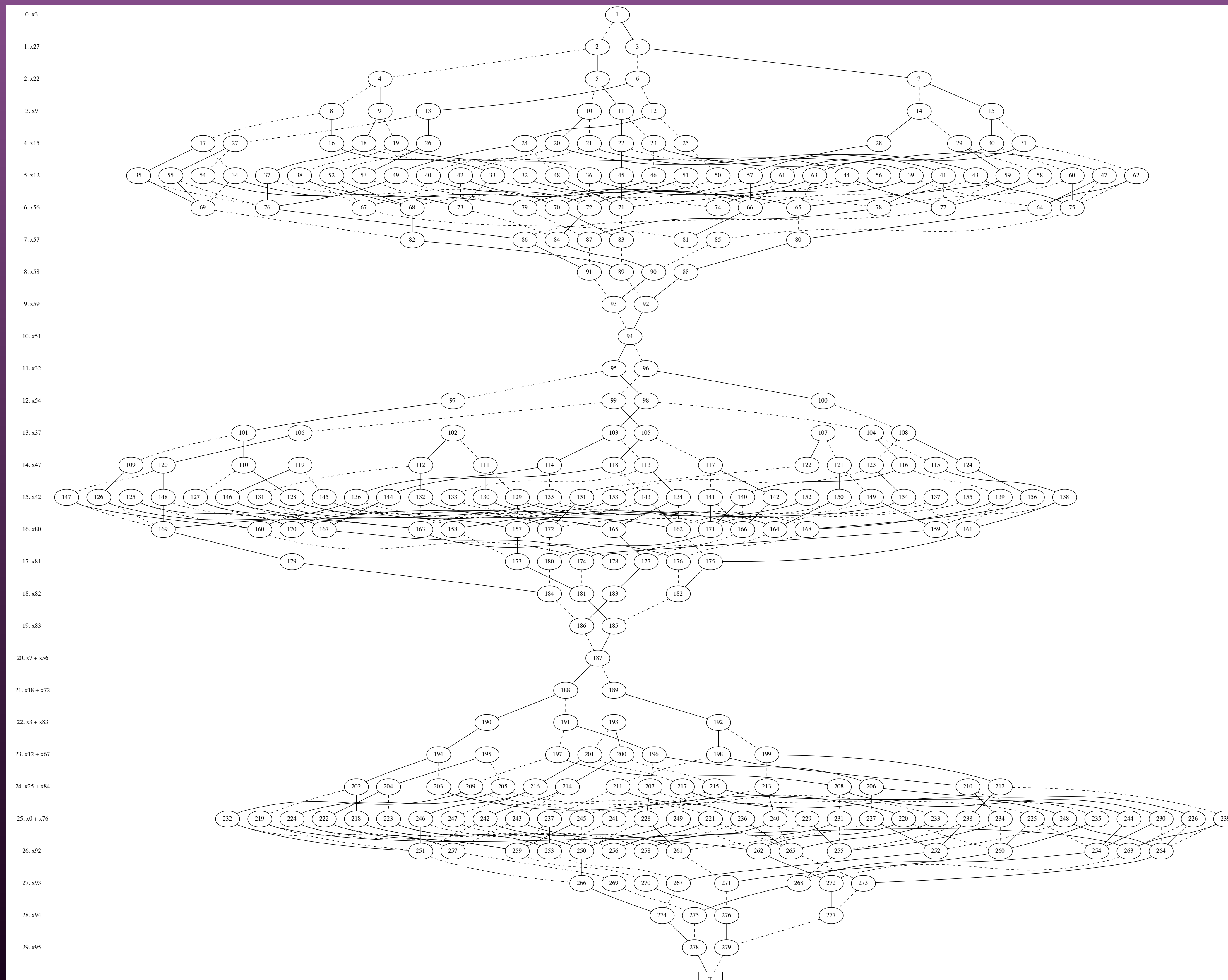
BDD Operations

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

x1 + x4

x2

x3 + x7

x3 + x4 + x6

x1 + x3 + x6

x5 + x7

Operations on a BDD

Linear absorption

$x2$

$x1 + x4$

$x3 + x7$

$x3 + x4 + x6$

$x1 + x3 + x6$

$x5 + x7$

Operations on a BDD

Linear absorption

$x2$

$x3 + x7$

$x1 + x4$

$x3 + x4 + x6$

$x1 + x3 + x6$

$x5 + x7$

Operations on a BDD

Linear absorption

x2

x3 + x7

x1 + x4

x1 + x3 +x 6

x1 + x3 + x6

x5 + x7

Operations on a BDD

Linear absorption

x2

x3 + x7

x1 + x4

x1 + x3 +x 6

0

x5 + x7

Operations on a BDD

Linear absorption

x2

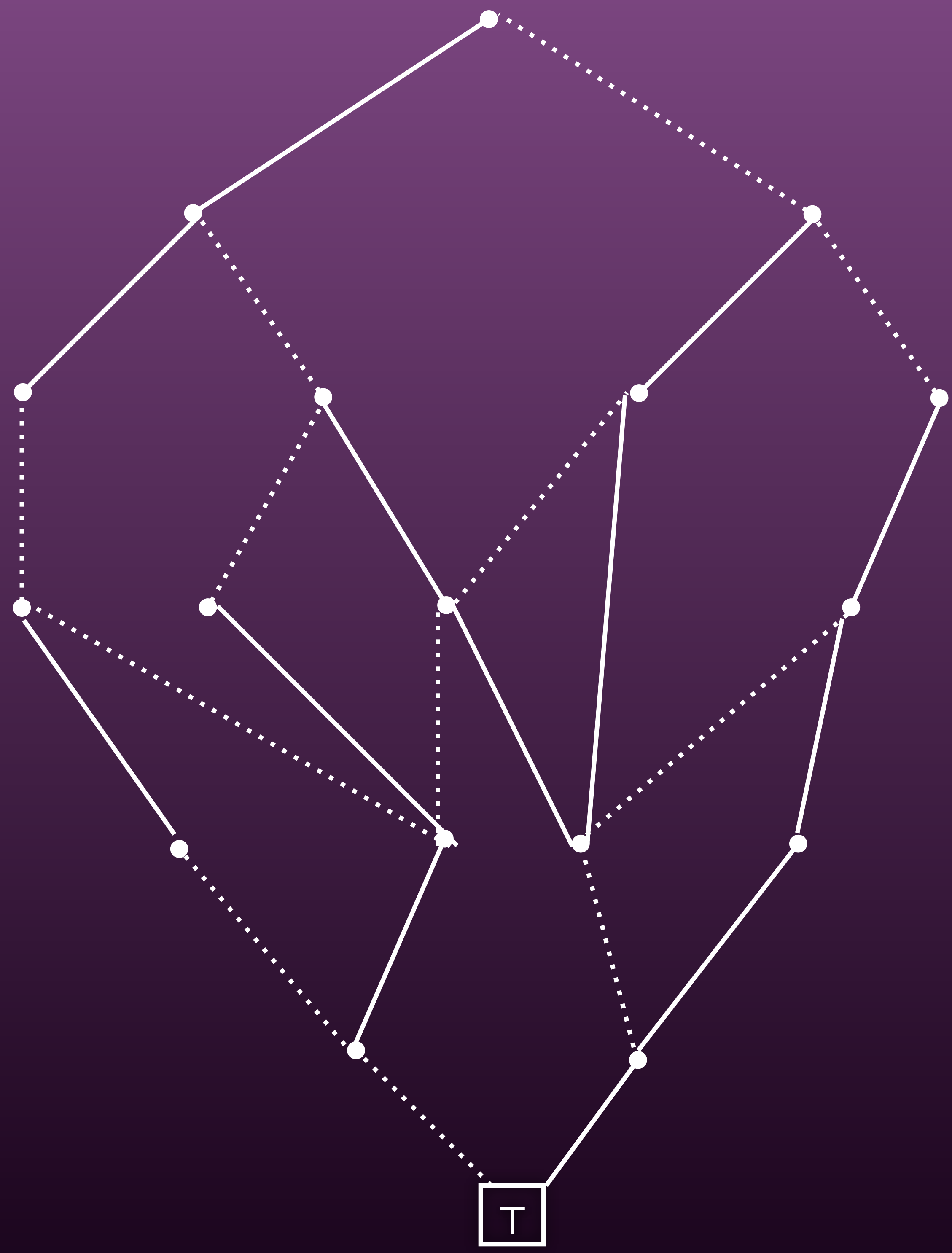
x3 + x7

x1 + x4

x1 + x3 +x 6

0

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

x2

x3 + x7

x1 + x4

x1 + x3 + x6

0

x5 + x7

Linear absorption

x2

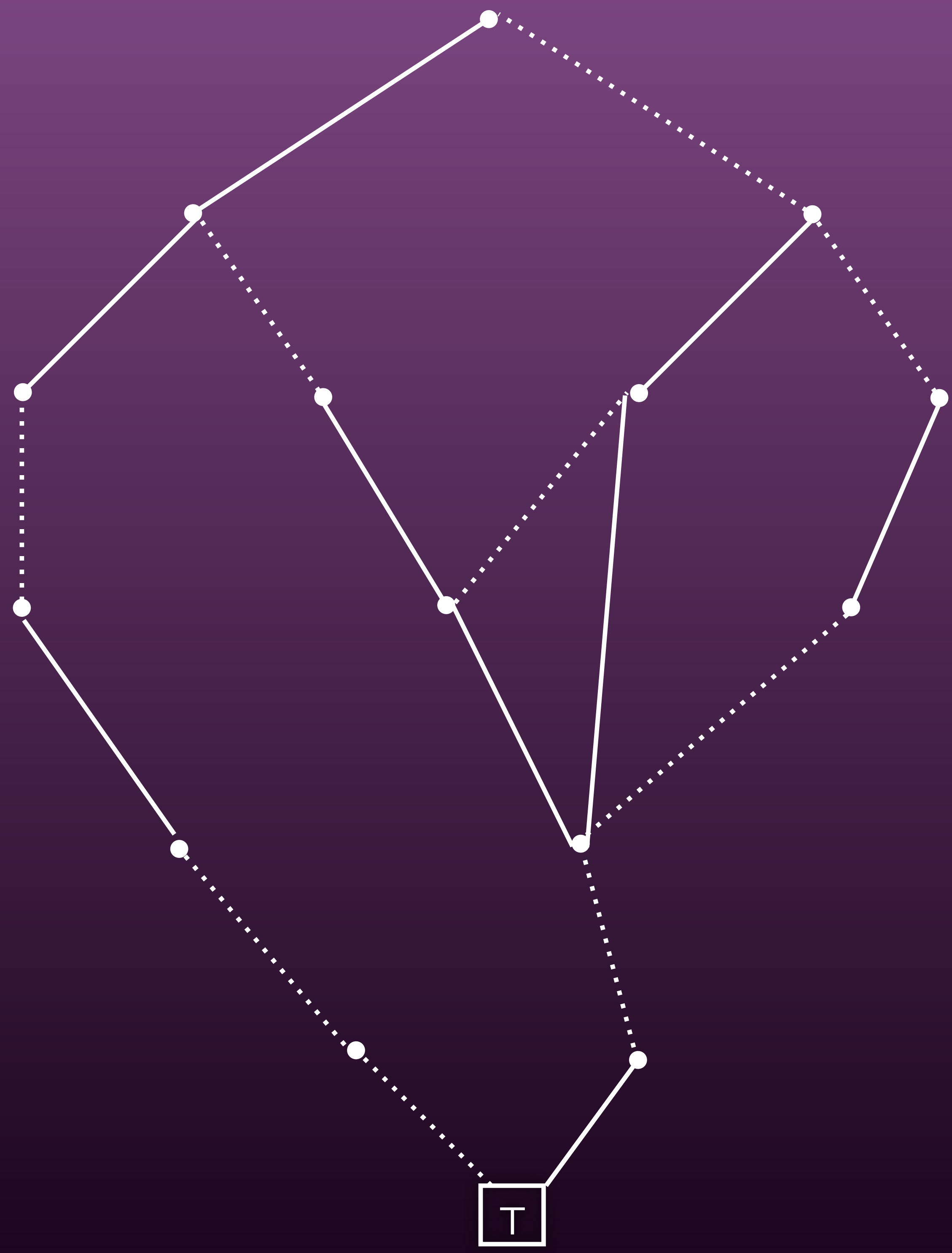
x3 + x7

x1 + x4

x1 + x3 + x6

0

x5 + x7



Linear absorption

x2

x3 + x7

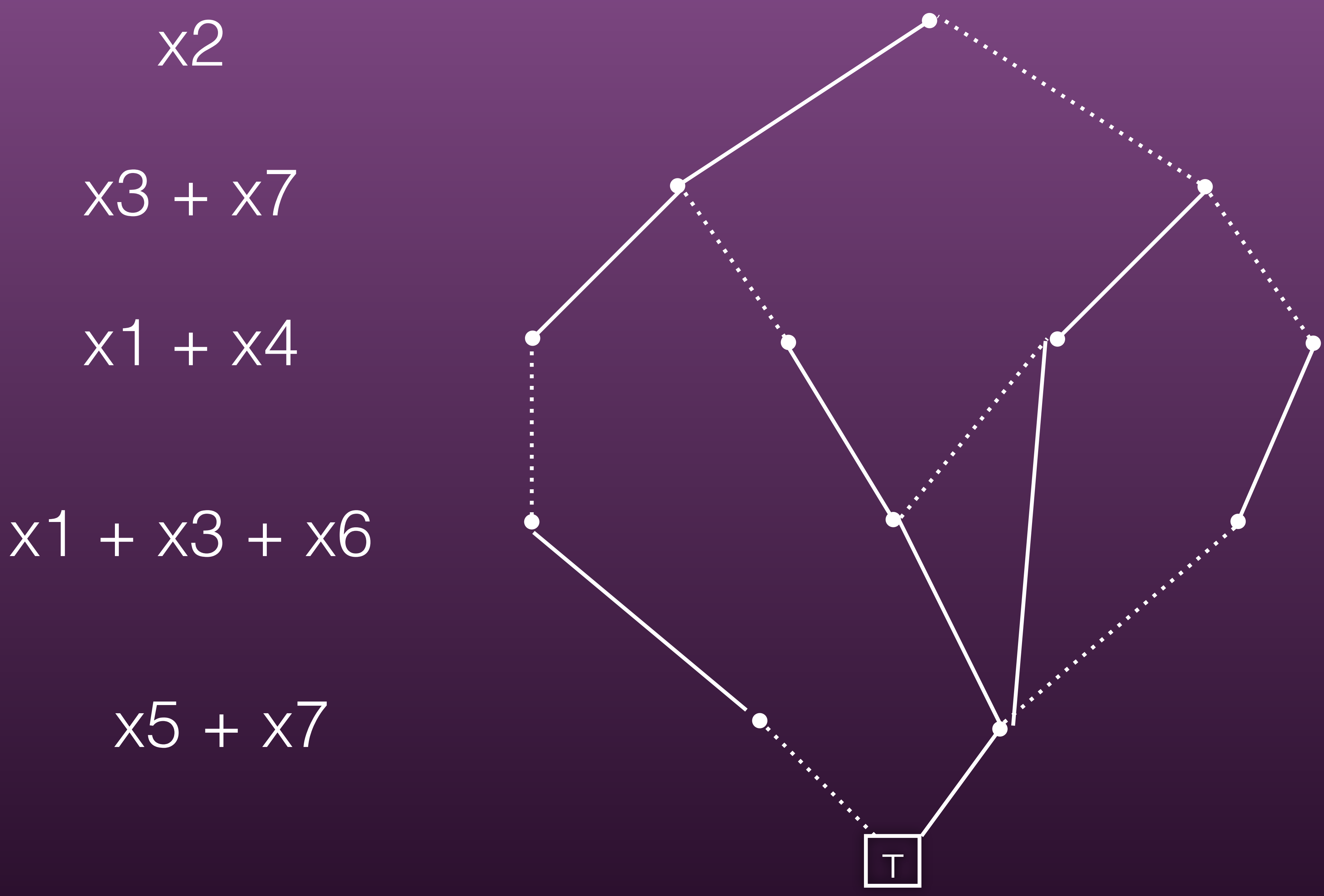
x1 + x4

x1 + x3 + x6

0

x5 + x7

Linear absorption



General solving algorithm

General solving algorithm

- While more than 1 BDD in system
  - Join some BDDs (in some order) creating a BDD with linear dependencies
  - Absorb linear dependencies

General solving algorithm

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

General solving algorithm

- While more than 1 BDD in system
  - ✦ Join some BDDs (in some order) creating a BDD with linear dependencies
  - ✦ Absorb linear 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

Practical results and examples

DES

DES

- 2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)

DES

- 2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)
- But… necessary to fix 20 bits of the key to correct values

DES

- 2007: an equation system for 6-round DES solved with MiniSat in 68 seconds (Courtois & Bard)
- But… necessary to 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)

MiniAES

MiniAES

- There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)

MiniAES

- There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)
- The best previous attack is only for 2 rounds

MiniAES

- There is no previous algebraic attacks on 10-round version (except CryptoMiniSAT)
- The best previous attack is only for 2 rounds
- BDD approach allows to break full version of MiniAES using only 1 known plaintext

Determining EA-equivalence

- Two vectorial Boolean functions F, G: $GF(2^n) \rightarrow GF(2^n)$ are EA-equivalent if for all x

$$F(x) = M_1 \cdot G(M_2 \cdot x + V_2) + M_3 \cdot x + V_1$$

- M_i are $n \times n$ matrices and V_j are n-bit vectors, M_1 and M_2 are 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	Gröbner basis	CryptoMiniSat
1	4	2	2	2	2
2	4	60	2	-	2
3	4	2	2	2	2
4	5	1	2	2	>2
5	5	155	2	-	>2

* Not finished after 78 hours

Conclusions

- New approaches to algebraic attacks development
- The BDD approach allows to reduce complexity of algebraic attack on DES by 2^{20}
- Practical algebraic attack on 10-round MiniAES was presented for the first time
- In some cases the BDD method is more universal and shows the best results of all known methods