Knowledge Compilation: Theory, Practice, and Applications

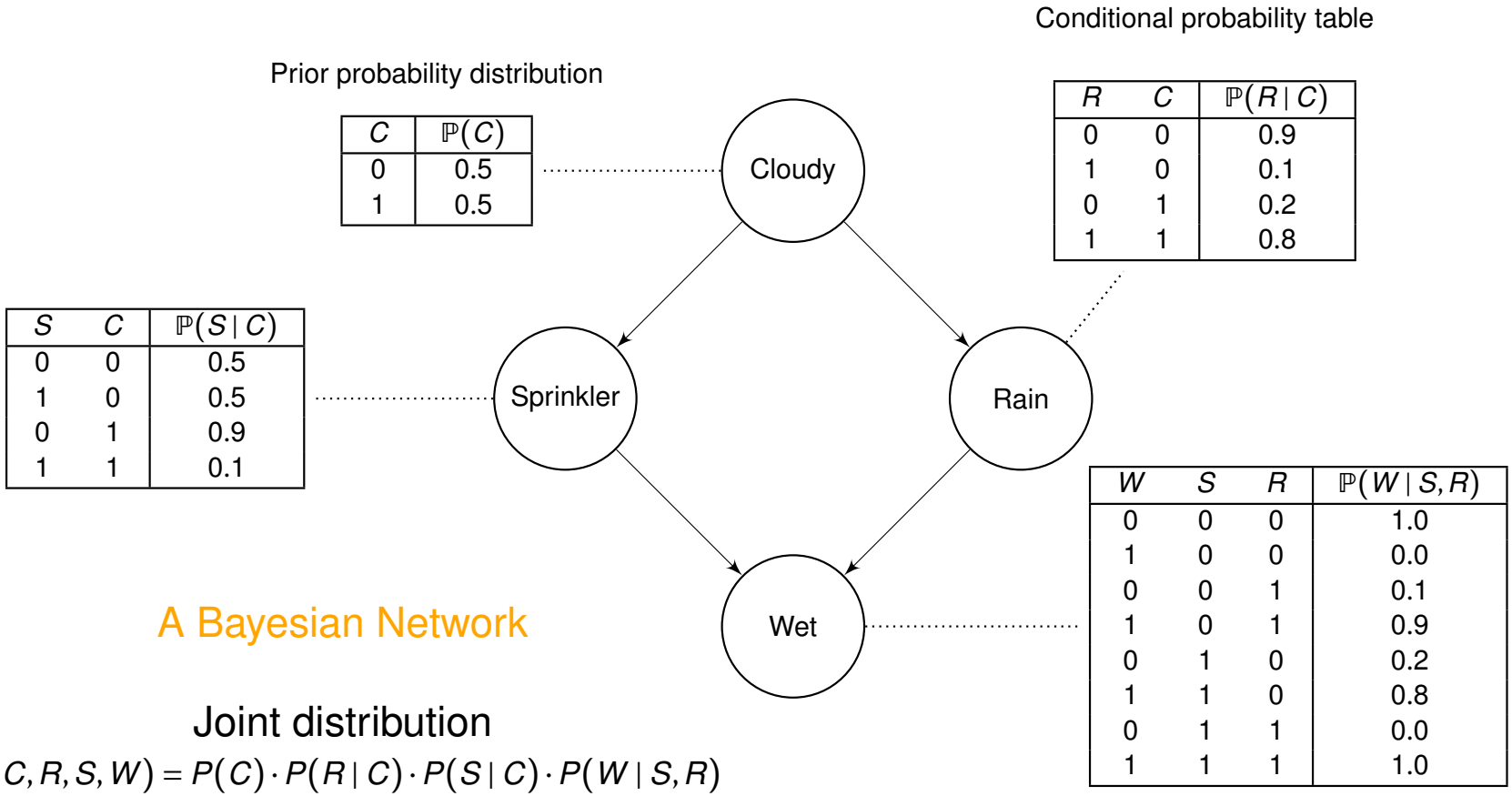
Jean-Marie Lagniez[1] Johannes Fichte[2]

[1] CRIL, Univ. Artois & CNRS, France
[2] Linköping University, Sweden

Outline

How to model using SAT

Graphical Models



C	$\mathbb{P}(C)$
0	0.5
1	0.5

R	C	$\mathbb{P}(R \mid C)$
0	0	0.9
1	0	0.1
0	1	0.2
1	1	0.8

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1

W	S	R	$\mathbb{P}(W \mid S,R)$
0	0	0	1.0
1	0	0	0.0
0	0	1	0.1
1	0	1	0.9
0	1	0	0.2
1	1	0	0.8
0	1	1	0.0
1	1	1	1.0

Joint distribution

$$P(C,R,S,W) = P(C) \cdot P(R \mid C) \cdot P(S \mid C) \cdot P(W \mid S,R)$$

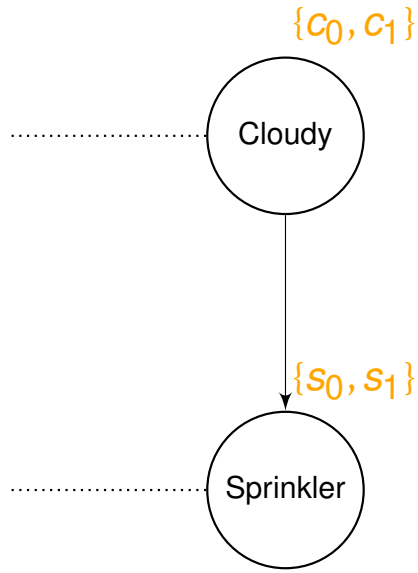
Graphical models are commonly separated into:

- directed models (a.k.a Bayesian networks), with conditional probability tables (CPTs) over a directed (acyclic) graph,
- undirected models (a.k.a. Markov networks), with energy functions over the cliques of an undirected graph.

ENC1 (Darwiche, 2002)

C	$\mathbb{P}(C)$
0	0.5
1	0.5

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1

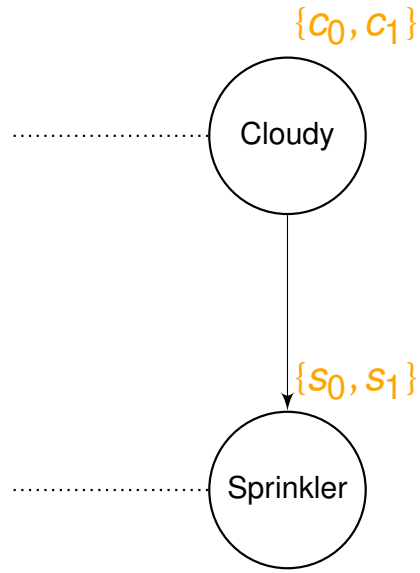


- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$;

ENC1 (Darwiche, 2002)

C	$\mathbb{P}(C)$
0	0.5
1	0.5

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1

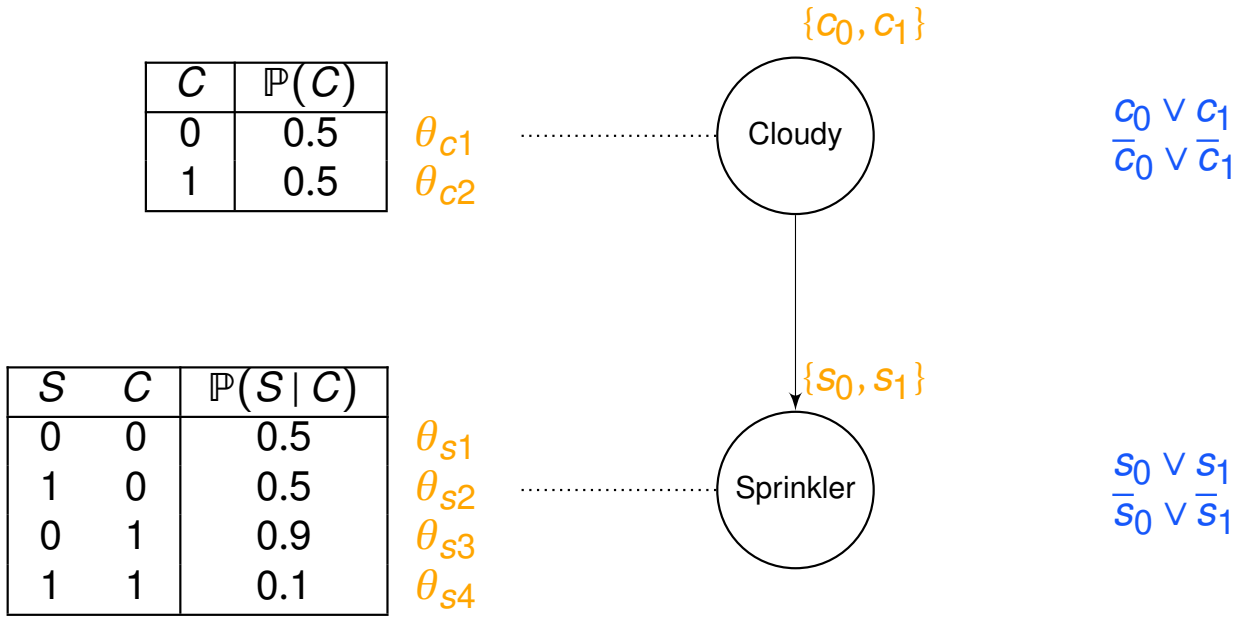


$c_0 \vee c_1$
$\overline{c}_0 \vee \overline{c}_1$

$s_0 \vee s_1$
$\overline{s}_0 \vee \overline{s}_1$

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$;
- A set of indicator clauses for each random variable X_i (direct encoding);

ENC1 (Darwiche, 2002)



C	$\mathbb{P}(C)$	
0	0.5	θ_{c1}
1	0.5	θ_{c2}

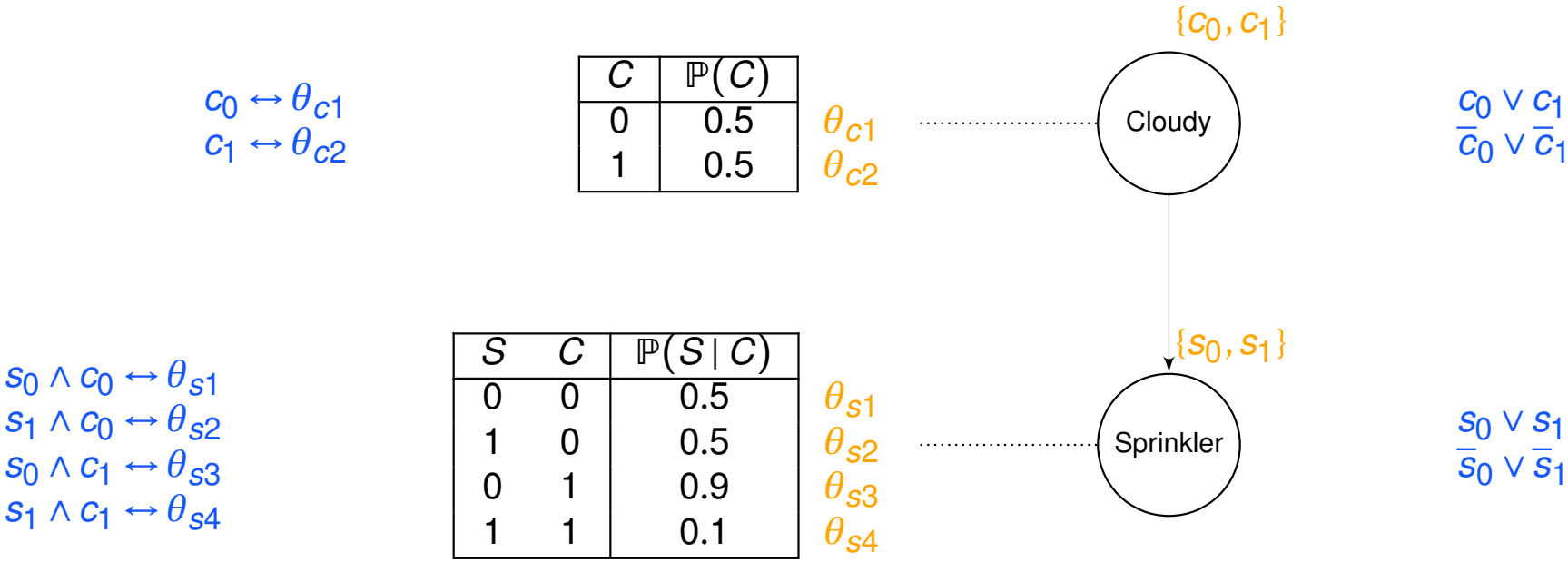
$c_0 \vee c_1$
$\overline{c}_0 \vee \overline{c}_1$

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	θ_{s1}
1	0	0.5	θ_{s2}
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

$s_0 \vee s_1$
$\overline{s}_0 \vee \overline{s}_1$

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$;
- A set of indicator clauses for each random variable X_i (direct encoding);
- A parameter variable θ_{ir} for each row r in the table of X_i, and a weight $w(\theta_{ir}) = \mathbb{P}(x_i \mid \boldsymbol{x}_r)$;

ENC1 (Darwiche, 2002)



C	$\mathbb{P}(C)$	
0	0.5	θ_{c1}
1	0.5	θ_{c2}

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	θ_{s1}
1	0	0.5	θ_{s2}
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$;
- A set of indicator clauses for each random variable X_i (direct encoding);
- A parameter variable θ_{ir} for each row r in the table of X_i, and a weight $w(\theta_{ir}) = \mathbb{P}(x_i \mid \boldsymbol{x}_r)$;
- A set of equivalences describing the semantics of each row in the table of X_i.

Local Structure

- In practice, variables are not necessary binary and CPTs can be quite large

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- Thus, the generated CNF encoding can be large, challenging state-of-the-art weighted model counters

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- The weight of a model is obtained by multiplying the weights of its positive parameters
- Then, all models which contain a positive parameter with a weight of 0 can be removed

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- The weight of a model is obtained by multiplying the weights of its positive parameters
- Then, all models which contain a positive parameter with a weight of 0 can be removed

$$\neg a_1 \vee \neg b_1 \vee \neg c_1 \qquad \neg a_2 \vee \neg b_1 \vee \neg c_1 \qquad \neg a_2 \vee \neg b_1 \vee \neg c_2$$

Equal Parameters

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

Equal Parameters

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- ENC1 will generate 9 parameter variables
- Group parameters that are equal and use a unique parameter variable by group

$$a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1}$$
$$a_1 \wedge b_1 \wedge c_3 \rightarrow \theta_{c1}$$
$$a_1 \wedge b_2 \wedge c_3 \rightarrow \theta_{c1}$$
$$a_2 \wedge b_2 \wedge c_3 \rightarrow \theta_{c1}$$

with $w(\theta_{c1}) = 0.5$ and we have to filter the models

Decomposability

- Model counters (and compilers) typically look for dependent components
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d\|b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$ $\quad \neg a_2 \vee \neg b_1 \vee \neg c_1$
$a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$
$a_1 \wedge b_2 \wedge c_1 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$
$a_1 \wedge b_2 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$

$b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$ $\quad b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$
$b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$ $\quad b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$
$b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$ $\quad b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$
$b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$ $\quad b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

Decomposability

- Model counters (and compilers) typically look for dependent components
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d\|b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$ $\quad \neg a_2 \vee \neg b_1 \vee \neg c_1$ $\quad b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$ $\quad b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$

$a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$ $\quad b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$ $\quad b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$

$a_1 \wedge b_2 \wedge c_1 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$ $\quad b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$ $\quad b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$

$a_1 \wedge b_2 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$ $\quad b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$ $\quad b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

- Computing implicants before translating into CNF promotes the identification of independent components

Decomposability

- Model counters (and compilers) typically look for dependent components
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d\|b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$ $\quad \neg a_2 \vee \neg b_1 \vee \neg c_1$ $\quad b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$ $\quad b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$

$a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$ $\quad b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$ $\quad b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$

$a_1 \wedge b_2 \wedge c_1 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$ $\quad b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$ $\quad b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$

$a_1 \wedge b_2 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$ $\quad b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$ $\quad b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

- Computing implicants before translating into CNF promotes the identification of independent components

Decomposability

- Model counters (and compilers) typically look for dependent components
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d\|b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \rightarrow \theta_{c1}$

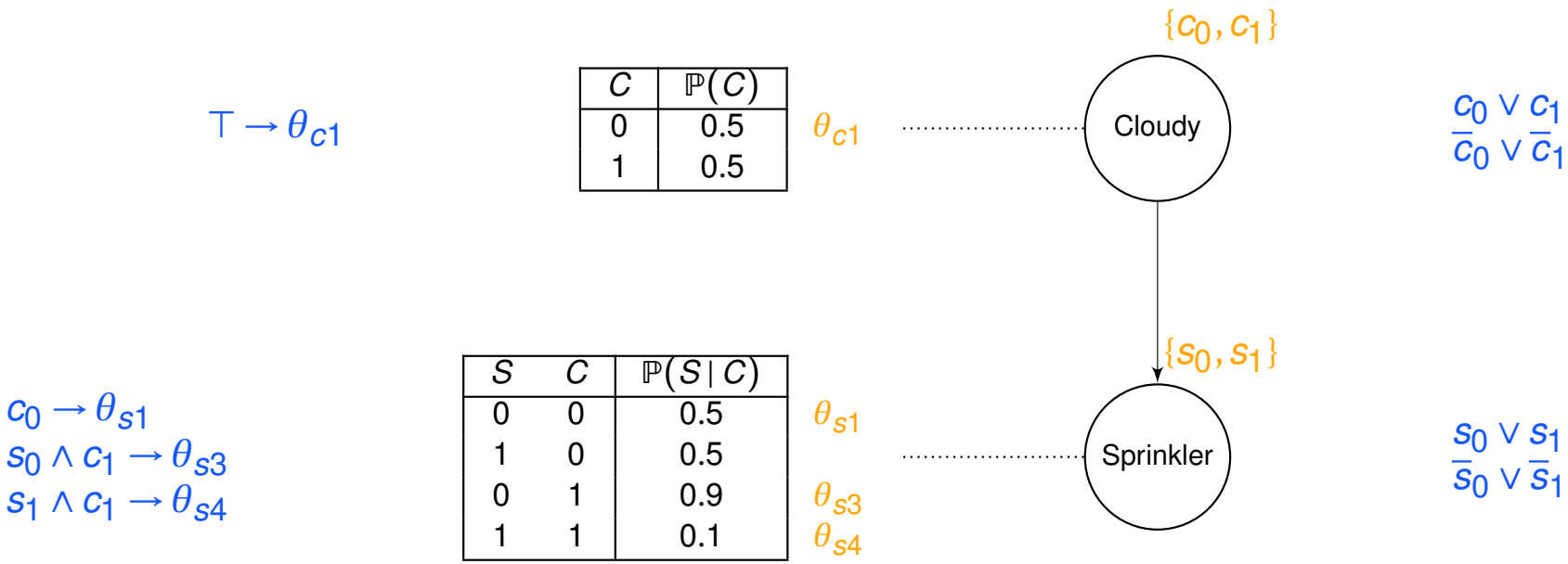
$\neg a_2 \vee \neg b_1 \vee \neg c_1$
$a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$
$a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$
$a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$

$b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$
$b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$
$b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$
$b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$

$b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$
$b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$
$b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$
$b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

- Computing implicants before translating into CNF promotes the identification of independent components

Chavira and Darwiche's Translation (Chavira and Darwiche, 2005)



A more compact encoding of tables:

- Group rows with the same probability, and compress them (into a prime DNF);
- Use implications instead of equivalences.

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

green
blue
gray
purple
black
white

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

green	0
blue	1
gray	2
purple	3
black	4
white	5

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0
blue	1
gray	2
purple	3
black	4
white	5

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0	$\neg c_2 \wedge \neg c_1 \wedge \neg c_0$
blue	1	$\neg c_2 \wedge \neg c_1 \wedge c_0$
gray	2	$\neg c_2 \wedge c_1 \wedge \neg c_0$
purple	3	$\neg c_2 \wedge c_1 \wedge c_0$
black	4	$c_2 \wedge \neg c_1 \wedge \neg c_0$
white	5	$c_2 \wedge \neg c_1 \wedge c_0$

Log Encoding (Bart et al., 2016)

- In the log encoding domains are represented with $m = \lceil \log_2(d) \rceil$ propositional variables
- Each of the 2^m combinations represents a possible assignment
- Introduce less indicator variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0	$\neg c_2 \wedge \neg c_1 \wedge \neg c_0$
blue	1	$\neg c_2 \wedge \neg c_1 \wedge c_0$
gray	2	$\neg c_2 \wedge c_1 \wedge \neg c_0$
purple	3	$\neg c_2 \wedge c_1 \wedge c_0$
black	4	$c_2 \wedge \neg c_1 \wedge \neg c_0$
white	5	$c_2 \wedge \neg c_1 \wedge c_0$

- We need to exclude the values in excess with a logarithmic number of clauses

$$\neg c_2 \vee \neg c_1 \vee \neg c_0 \qquad \neg c_2 \vee \neg c_1 \vee c_0$$

- Domains having 2^d values don't need indicator clauses!

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit
- Instantiations with default value are left implicit

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

0.3

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit
- Instantiations with default value are left implicit

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

0.3

$\frac{0.2}{0.3}$

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit
- Instantiations with default value are left implicit

A	B	C	$\mathbb{P}$(C|A,B)	0.3
0	0	0	0.2	$\frac{0.2}{0.3}$
0	0	1	0.3	
0	1	0	0.3	
0	1	1	0.2	$\frac{0.2}{0.3}$
1	0	0	0	
1	0	1	0	
1	1	0	0.6	$\frac{0.6}{0.3}$
1	1	1	0.4	$\frac{0.4}{0.3}$

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit
- Instantiations with default value are left implicit

	A	B	C	$\mathbb{P}$(C|A,B)	0.3
θ_{c1}	0	0	0	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	0	0	1	0.3	
	0	1	0	0.3	
θ_{c1}	0	1	1	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	1	0	0	0	
	1	0	1	0	
θ_{c2}	1	1	0	0.6	$\frac{0.6}{0.3} = w(\theta_{c2})$
θ_{c3}	1	1	1	0.4	$\frac{0.4}{0.3} = w(\theta_{c3})$

$a_0 \wedge b_0 \wedge c_0 \rightarrow \theta_{c1}$ $\quad a_0 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$

$a_1 \wedge b_1 \wedge c_0 \rightarrow \theta_{c2}$ $\quad a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c3}$

Scaling Factor (Bart et al., 2016)

- For each instantiation, exactly one parameter variable will be assigned to true
- Then, we can keep for each CPT R one parameter variable θ_R implicit
- Instantiations with default value are left implicit

	A	B	C	$\mathbb{P}$(C|A,B)	0.3
θ_{c1}	0	0	0	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	0	0	1	0.3	
	0	1	0	0.3	
θ_{c1}	0	1	1	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	1	0	0	0	
	1	0	1	0	
θ_{c2}	1	1	0	0.6	$\frac{0.6}{0.3} = w(\theta_{c2})$
θ_{c3}	1	1	1	0.4	$\frac{0.4}{0.3} = w(\theta_{c3})$

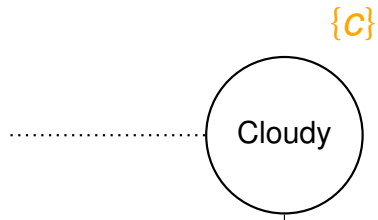
$a_0 \wedge b_0 \wedge c_0 \rightarrow \theta_{c1}$ $\quad a_0 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$

$a_1 \wedge b_1 \wedge c_0 \rightarrow \theta_{c2}$ $\quad a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c3}$

- To make the translation faithful we now assign a specific weight to the negative parameter literals: $w(\neg\theta_{ij}) = 1 - w(\theta_{ij})$
  Accounts for the instantiations with default implicit (0.3) and the factor for the remaining ones

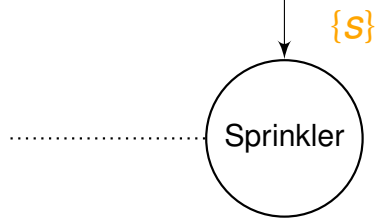
Running Example

C	$\mathbb{P}(C)$
0	0.5
1	0.5



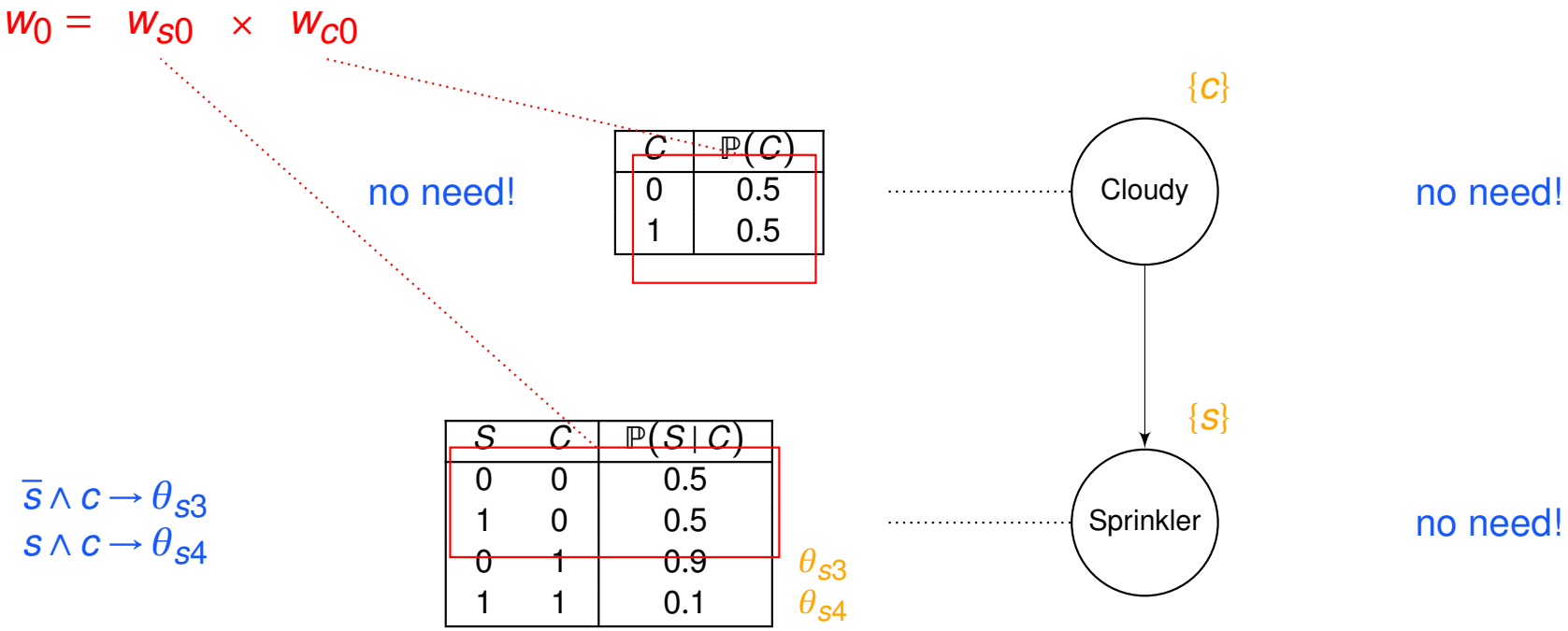
no need!

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	
1	0	0.5	
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

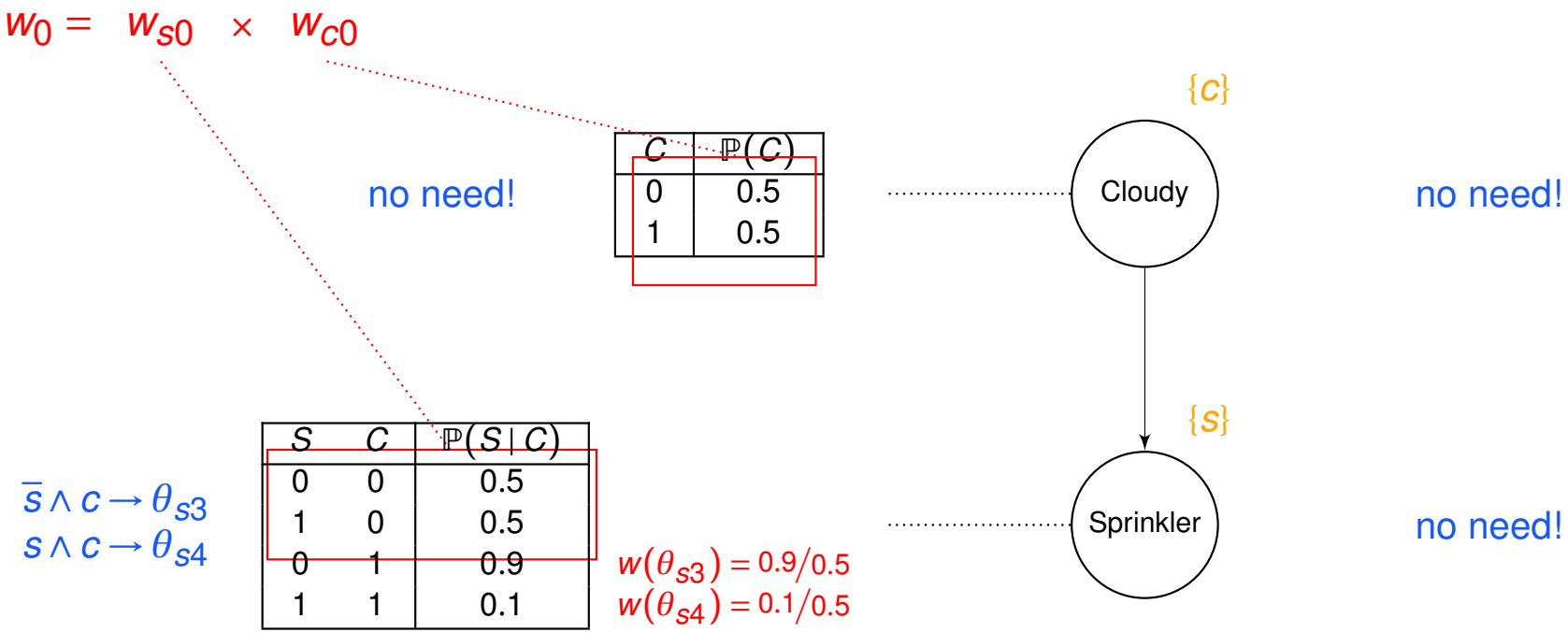


no need!

Running Example



Running Example



- The weighted model count of (F, w, w_0) becomes:

$$W(F) = w_0 \left(\sum_{\boldsymbol{v} \in \{0,1\}^n, \boldsymbol{v} \models F} W(\boldsymbol{v}) \right)$$

Resources for SAT Modeling

For deeper insights into modeling problems using propositional logic and evaluating modern solvers, consider the following standard resources:

Satisfiability (SAT)

- **Handbook of Satisfiability** Biere et al. (2021)
  The definitive guide to SAT theory, encoding techniques, and applications.
- **The SAT Competition**
  Annual evaluation of state-of-the-art solvers.
  `https://satcompetition.github.io/`

Model Counting (#SAT)

- **Model Counting Competition (MC)**
  The premier benchmark track for exact and approximate #SAT solvers.
  `https://mccompetition.org/`

Outline

Knowledge Compilation

Outline

Motivations for Knowledge Compilation

- **The Bottleneck:** SAT and #SAT are NP-complete and #P-complete. In practice, there is no guarantee of solving queries within a strict time limit.

The KC Paradigm: Split the Effort

1. **Offline Phase (Compilation):** Translate the instance Σ into a representation $\mathscr{C}$ from a target language $\mathscr{L}$. This step is computationally expensive.
2. **Online Phase (Querying):** Use $\mathscr{C}$ to answer difficult queries (like Model Counting) in **polynomial time**.

- **When is it useful?** When the offline cost is amortized over *many* queries on the same fixed knowledge base.
- **The Question:** Which language $\mathscr{L}$ should we choose? We use the *Knowledge Compilation Map*.

Outline

KC Map: Evaluating a Language $\mathscr{L}$

To choose a language, we evaluate its tractability on two axes:

1. Queries
(Extracting Information)

- **CO** (Consistency / SAT)
- **CE** (Clause Entailment)
- **VA** (Validity)
- **EQ** (Equivalence)
- **SE** (Sentential Entailment)
- **IM** (Implicants)
- **CT** (Model Counting)
- **ME** (Model Enumeration)

2. Transformations
(Modifying the Knowledge)

- **CD** (Conditioning)
- $\wedge$**C**, $\wedge$**BC** (Closure under $\wedge$)
- $\vee$**C**, $\vee$**BC** (Closure under $\vee$)
- $\neg$**C** (Closure under $\neg$)
- **FO**, **SFO** (Forgetting)

Target Languages: The Baselines

Before evaluating their properties, let us define our starting languages.

`Circ` (Boolean Circuits)

A Directed Acyclic Graph (DAG) where internal nodes are Boolean gates ($\wedge, \vee, \neg$) and leaves are variables or constants.

- **Pros:** Can represent any Boolean function compactly.
- **Cons:** Almost all queries (SAT, #SAT) remain NP-hard / #P-hard.

`CNF`

An AND of ORs (clauses).

- **Ex:** $(a \vee \neg b) \wedge (\neg a \vee c)$
- Great for modeling (like our Zykov encoding).
- **CO** (Consistency) is hard.

`DNF`

An OR of ANDs (terms).

- **Ex:** $(a \wedge \neg b) \vee (\neg a \wedge c)$
- Explicitly lists partial models.
- **CO** is easy ($\surd$), but **CT** (Counting) is still hard!

Fragment of the KC Map

- $\surd$: Polynomial-time algorithm exists.
- $\circ$: No poly-time algorithm (unless P = NP).

Queries

$\mathscr{L}$	CO	VA	CE	IM	EQ	SE	CT	ME
Circ	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$
CNF	$\circ$	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$
DNF	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$

Transformations

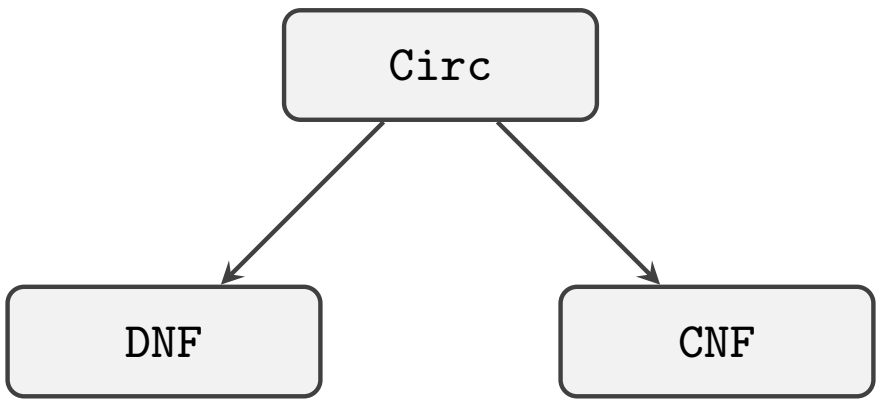
$\mathscr{L}$	CD	FO	SFO	$\wedge$C	$\wedge$BC	$\vee$C	$\vee$BC	$\neg$C
Circ	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
CNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$
DNF	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$

Outline

Succinctness ($\leq_s$)

Succinctness captures the ability of a language to represent information **compactly**.

- $\mathscr{L}_1 \leq_s \mathscr{L}_2$ ($\mathscr{L}_1$ is *at least as succinct as* $\mathscr{L}_2$) if there exists a polynomial p such that for every formula $\alpha \in \mathscr{L}_2$, there is an equivalent $\beta \in \mathscr{L}_1$ where $|\beta| \leq p(|\alpha|)$.
- An arrow $\mathscr{L}_1 \to \mathscr{L}_2$ strictly means $\mathscr{L}_1 <_s \mathscr{L}_2$ (`Circ` is strictly more succinct than the others)



Outline

Leveraging Modern SAT Solvers

- Over the last two decades, SAT solvers (CDCL) have become incredibly efficient at finding a single satisfying assignment for massive `CNF` formulas.
- **The Question:** Instead of building a custom compilation algorithm from scratch, can we treat an off-the-shelf SAT solver as an *oracle* to compile our formula?

The Black-Box Compilation Approach

Iteratively query the SAT solver. When it finds a solution, record it, *block it* from the search space, and query the solver again until the formula becomes unsatisfiable.

- **What target languages does this produce?**
  - If we extract full assignments (models), we generate `MODS`.
  - If we extract partial assignments (prime implicants), we generate `ODNF`.
- To understand this approach, let us first review how modern CDCL SAT solvers work.

The Resolution Rule

- **The Rule:** Two clauses containing a variable x with *clashing literals* $(x, \neg x)$ imply a *resolvent* clause containing the union of the remaining literals.

$$\frac{x \vee \underbrace{y \vee \neg z}_{C_1} \qquad \neg x \vee \underbrace{t \vee u}_{C_2}}{y \vee \neg z \vee t \vee u}$$

- **Intuition:** If the formula is true, x must be either True or False. In either case, one of the remaining parts (C_1 or C_2) must hold.
- **Variable Elimination:** Performing all possible resolutions on x is equivalent to *forgetting* x (existential quantification):

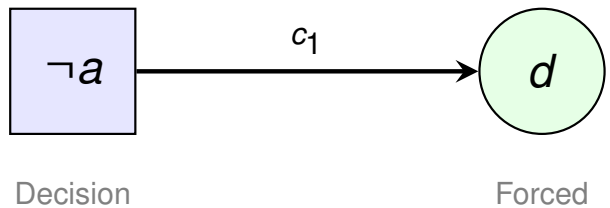
$$\exists x.\Sigma \equiv (\Sigma|_x) \vee (\Sigma|_{\neg x})$$

- **Property:** Resolution is a refutation-complete proof system for CNF formulas.

Ingredients of a CDCL Solver

- **1. Decisions & Propagation**
  - **Branching Heuristics:** Choosing the next variable (e.g., VSIDS) and its polarity.
  - **Unit Propagation (BCP):** Deduce values forced by clauses.

$$\phi = \{c_1 : a \vee d\}$$



- **2. Conflict Analysis & Learning**
  - Building the *Implication Graph* upon conflict.
  - *Clause Learning*: Deriving a new clause to prevent this mistake.
  - *Backjumping*: Non-chronological backtracking.

Key Mechanism

Constructing the **Implication Graph** allows the solver to learn from failure.

Constructing the Implication Graph

Current Formula State

$c_1 : a \vee d$ $\quad$ $c_2 : a \vee \neg c \vee \neg f$ $\quad$ $c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad$ $c_5 : \neg c \vee \neg e \vee i$ $\quad$ $c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad$ $c_8 : e \vee \neg h \vee k$ $\quad$ $c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1

DL 2

DL 3

DL 4

Constructing the Implication Graph

Current Formula State

$c_1 : a \vee d$ $\quad$ $c_2 : a \vee \neg c \vee \neg f$ $\quad$ $c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad$ $c_5 : \neg c \vee \neg e \vee i$ $\quad$ $c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad$ $c_8 : e \vee \neg h \vee k$ $\quad$ $c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1 $\boxed{\neg a}$

DL 2

DL 3

DL 4

Constructing the Implication Graph

Current Formula State

$c_1 : a \vee d$ $\quad$ $c_2 : a \vee \neg c \vee \neg f$ $\quad$ $c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad$ $c_5 : \neg c \vee \neg e \vee i$ $\quad$ $c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad$ $c_8 : e \vee \neg h \vee k$ $\quad$ $c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1 $\neg a$

DL 2

DL 3

DL 4

Constructing the Implication Graph

Current Formula State

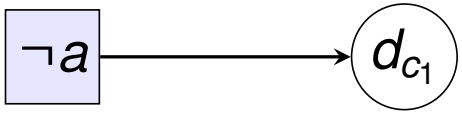
$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1



DL 2

DL 3

DL 4

Constructing the Implication Graph

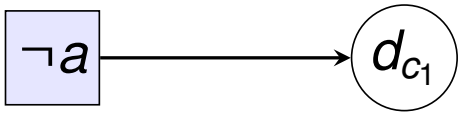
Current Formula State

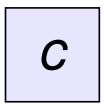
$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1 

DL 2 

DL 3

DL 4

Constructing the Implication Graph

Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:

DL 1 $\neg a \rightarrow d_{c_1}$

DL 2 c

DL 3

DL 4

Constructing the Implication Graph

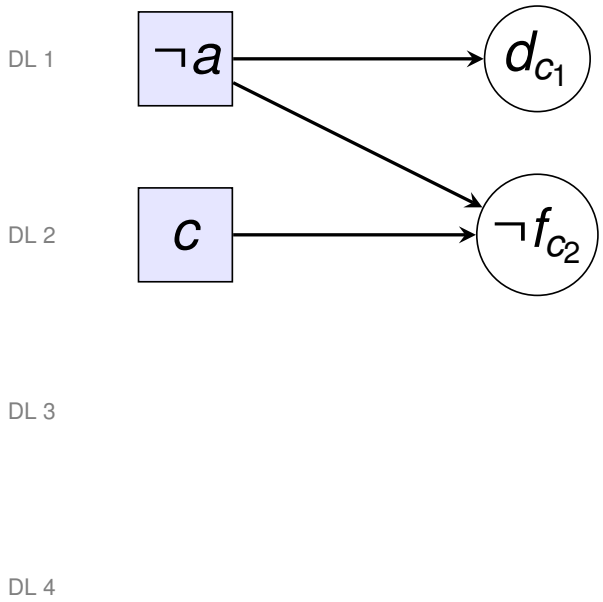
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

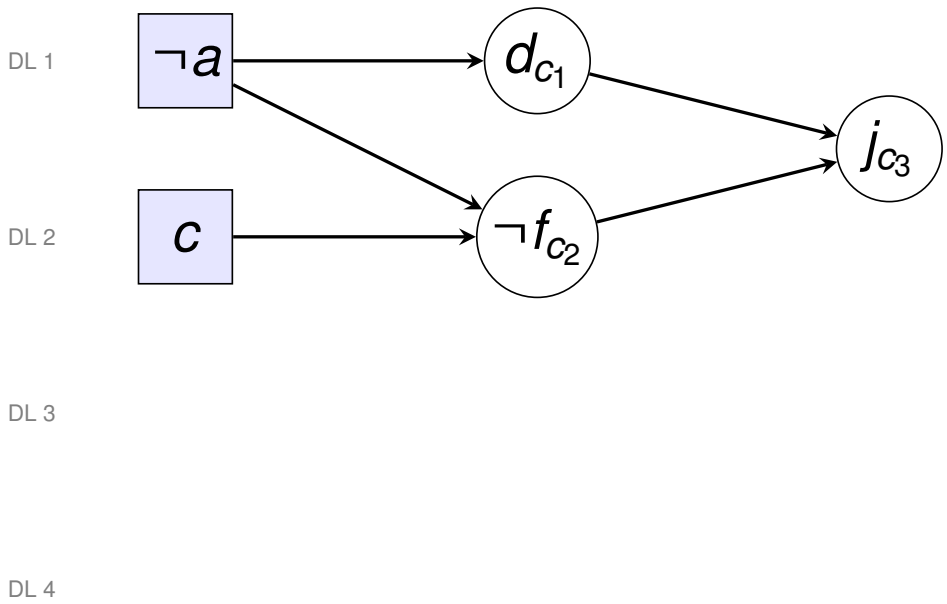
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

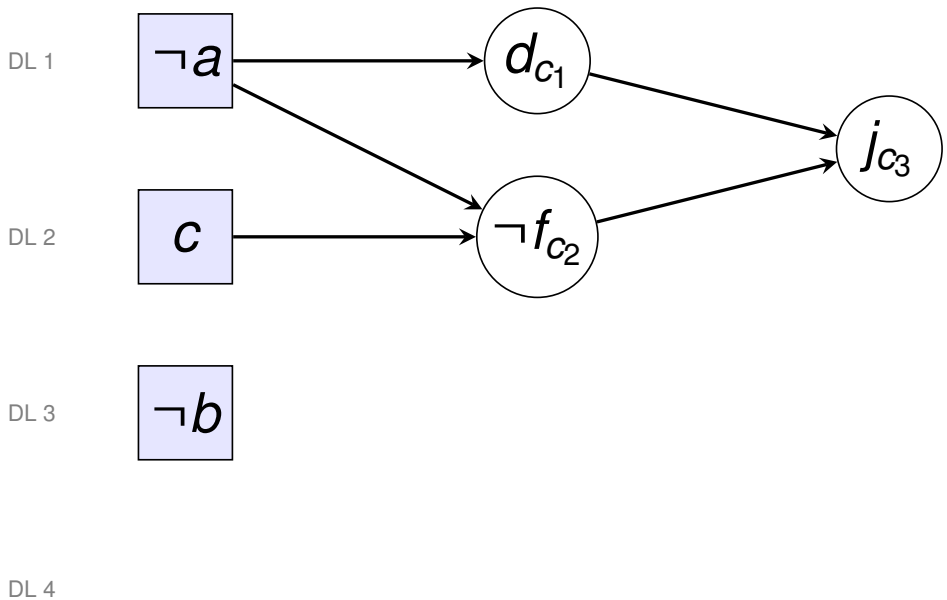
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

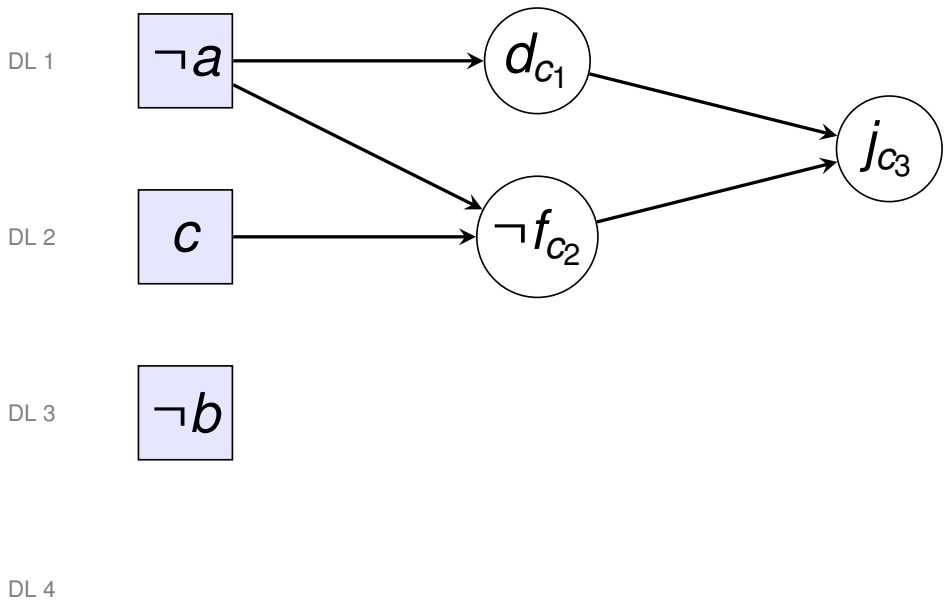
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

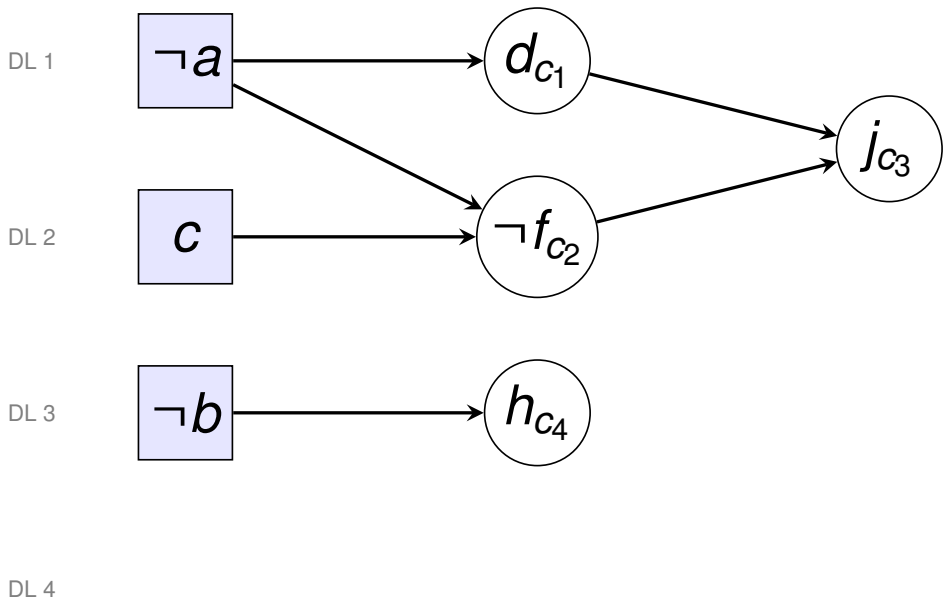
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

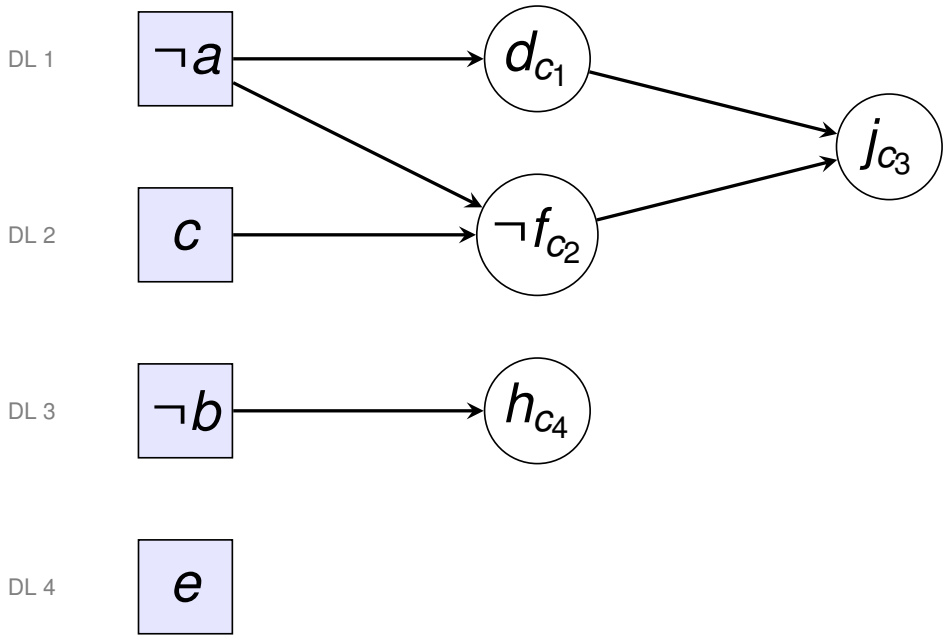
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

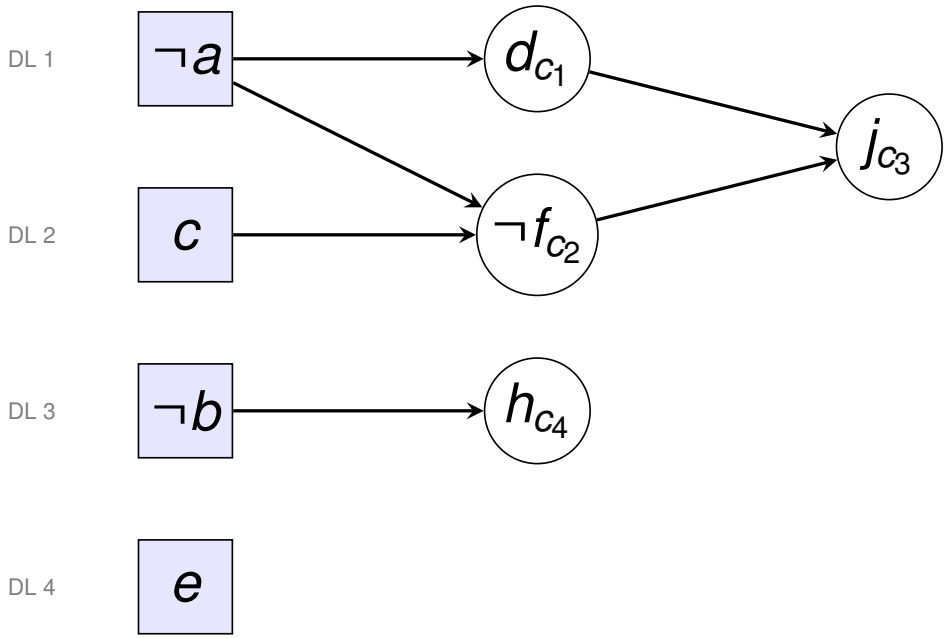
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

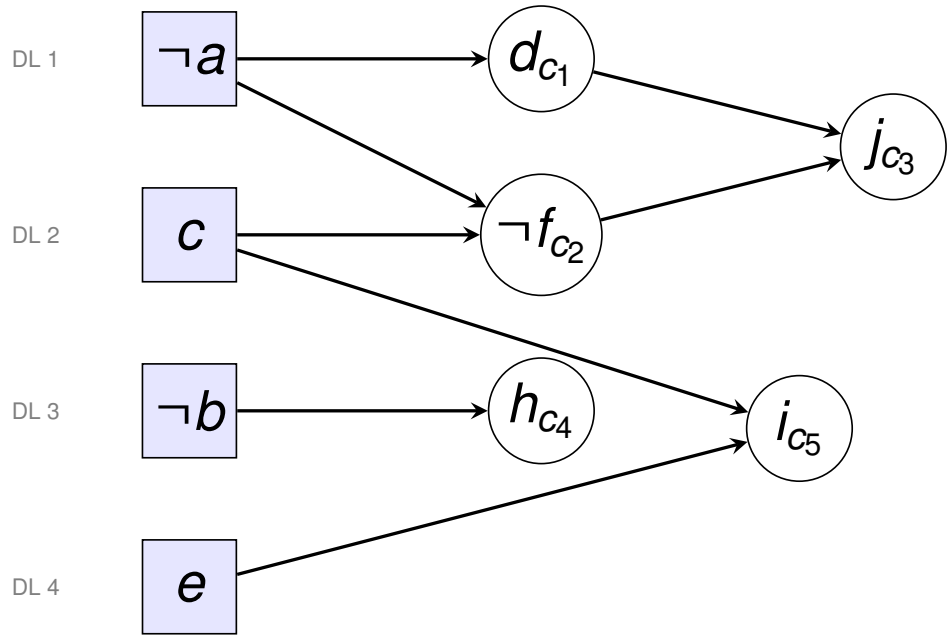
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

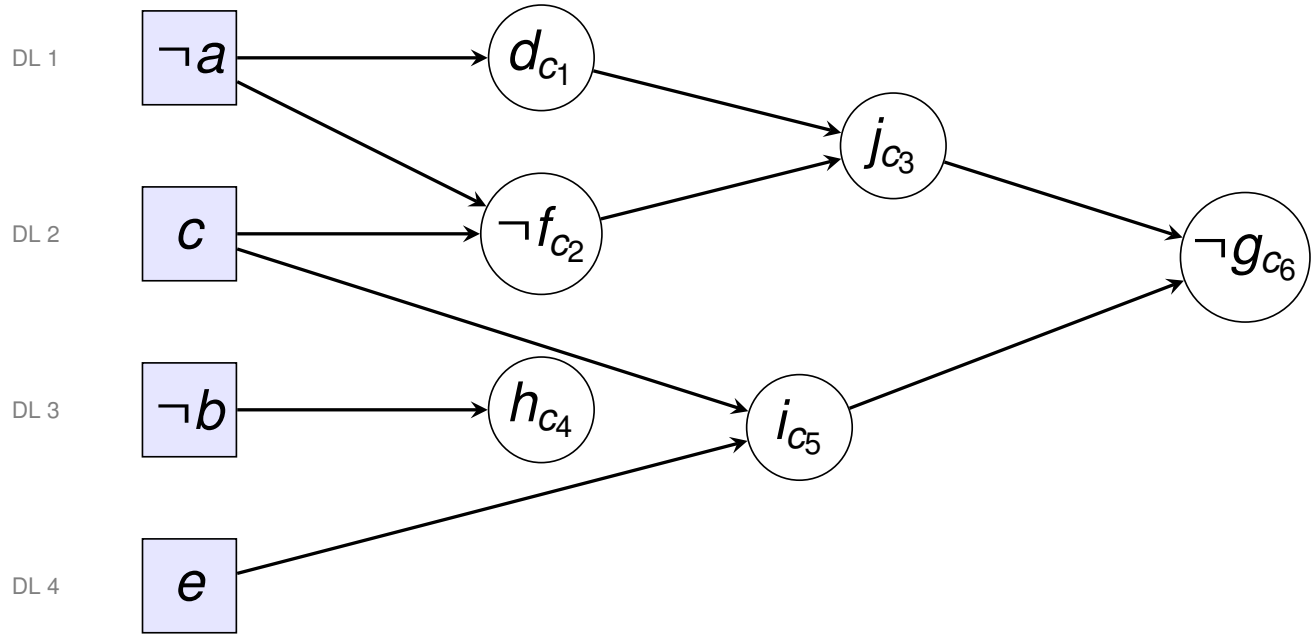
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

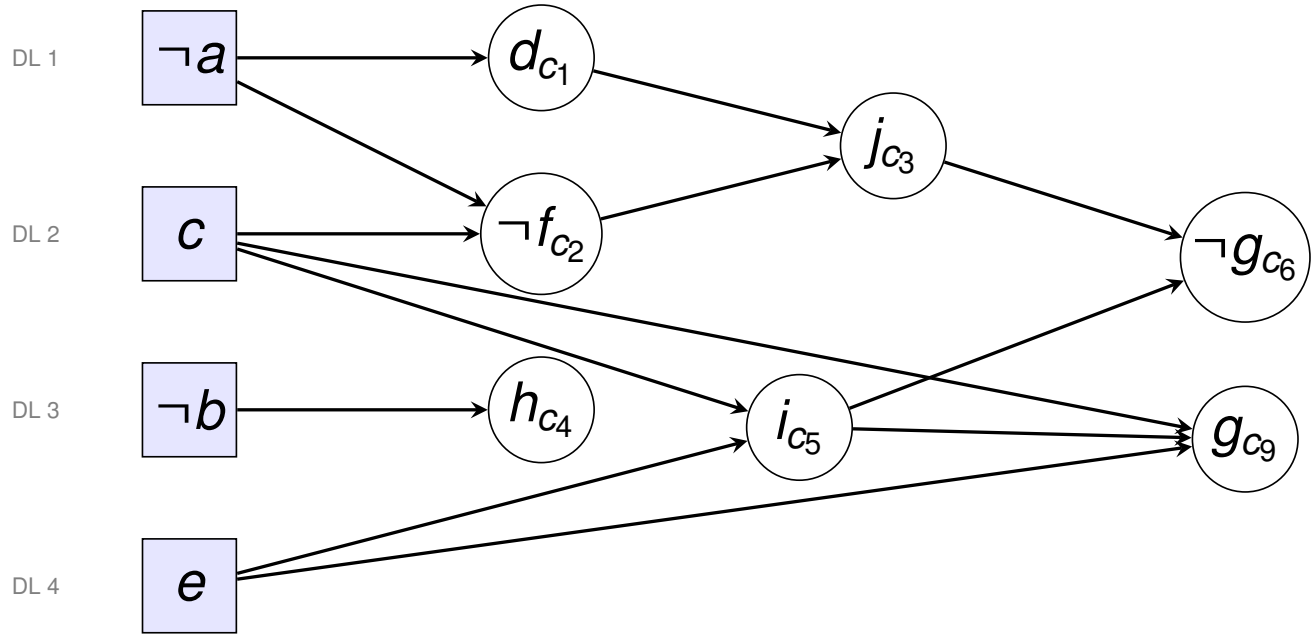
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

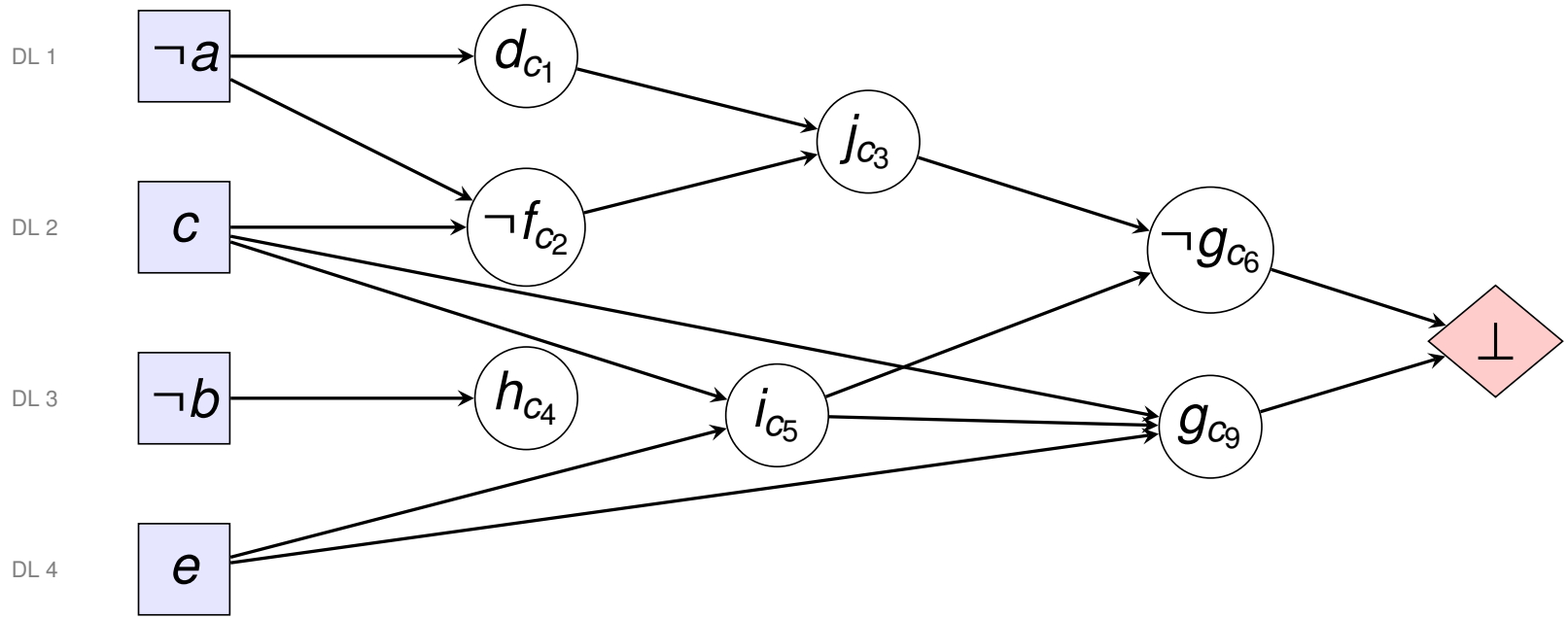
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

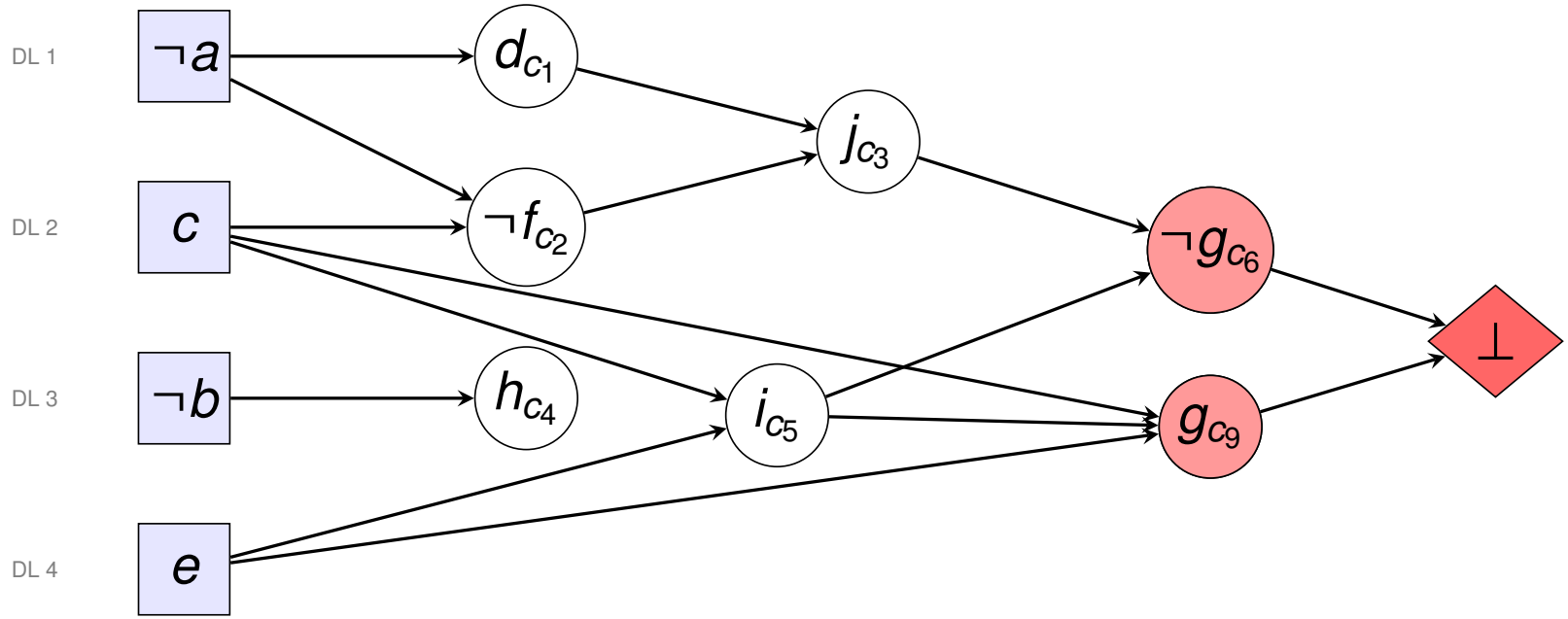
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

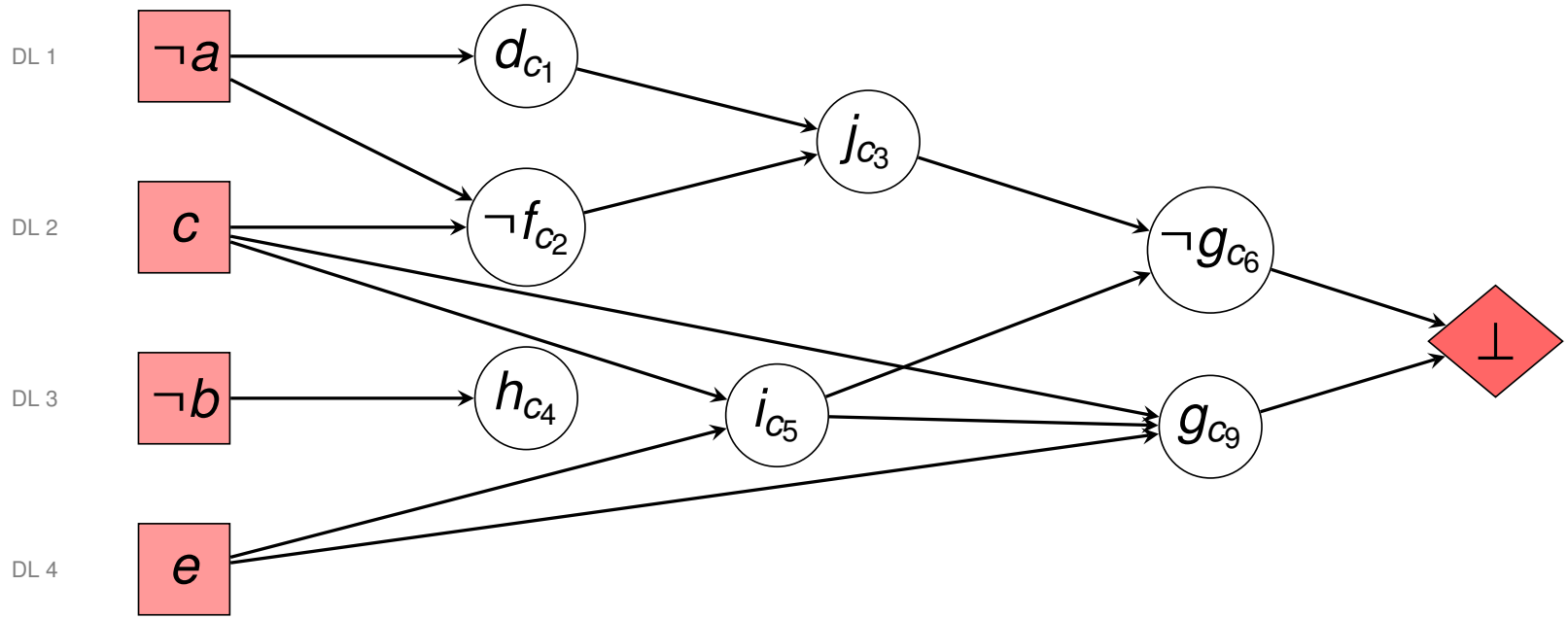
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



Constructing the Implication Graph

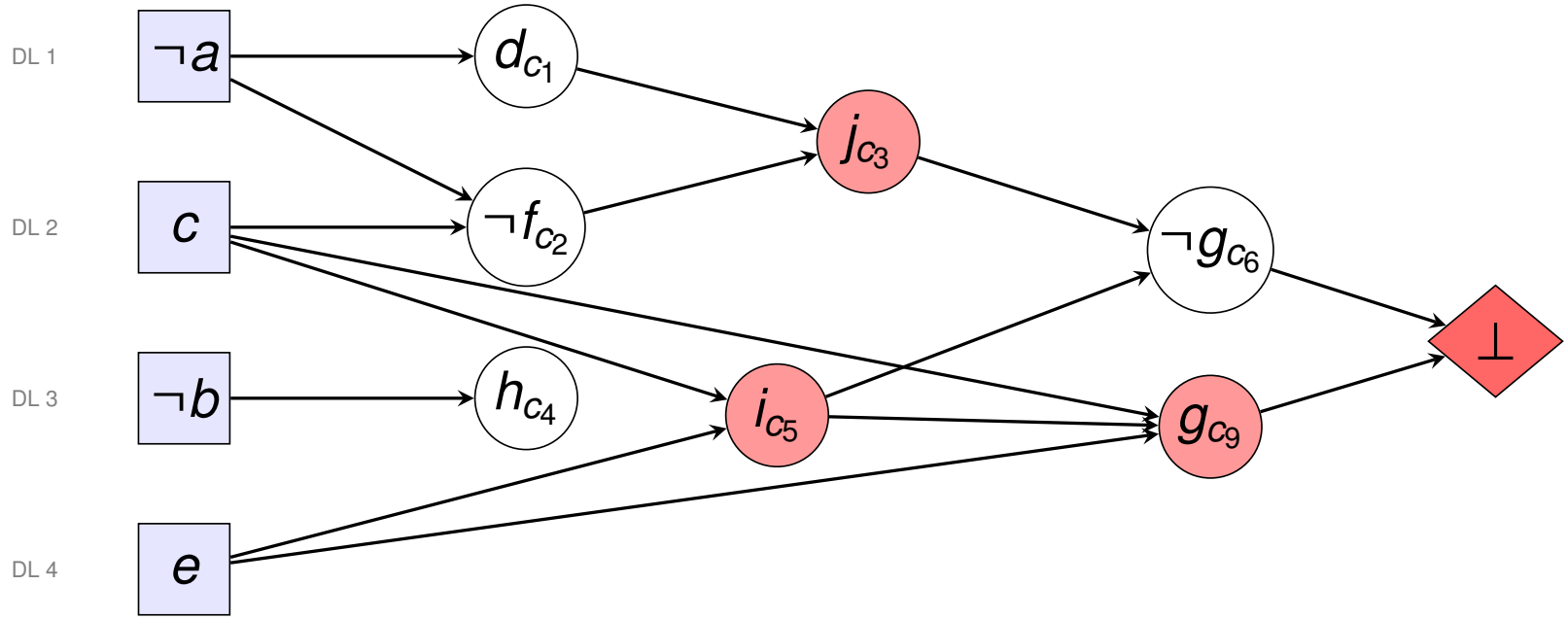
Current Formula State

$c_1 : a \vee d$ $\quad c_2 : a \vee \neg c \vee \neg f$ $\quad c_3 : \neg d \vee j \vee f$

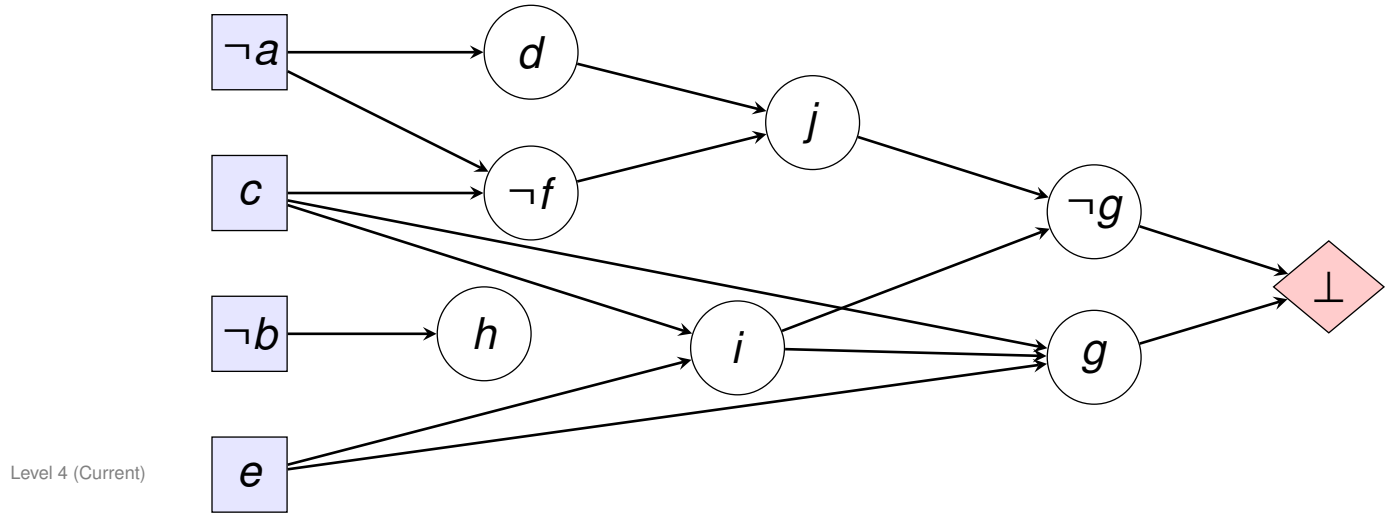
$c_4 : b \vee h$ $\quad c_5 : \neg c \vee \neg e \vee i$ $\quad c_6 : \neg i \vee \neg j \vee \neg g$

$c_7 : e \vee \neg k$ $\quad c_8 : e \vee \neg h \vee k$ $\quad c_9 : \neg c \vee \neg e \vee \neg i \vee g$

Decisions & Propagation Trace:



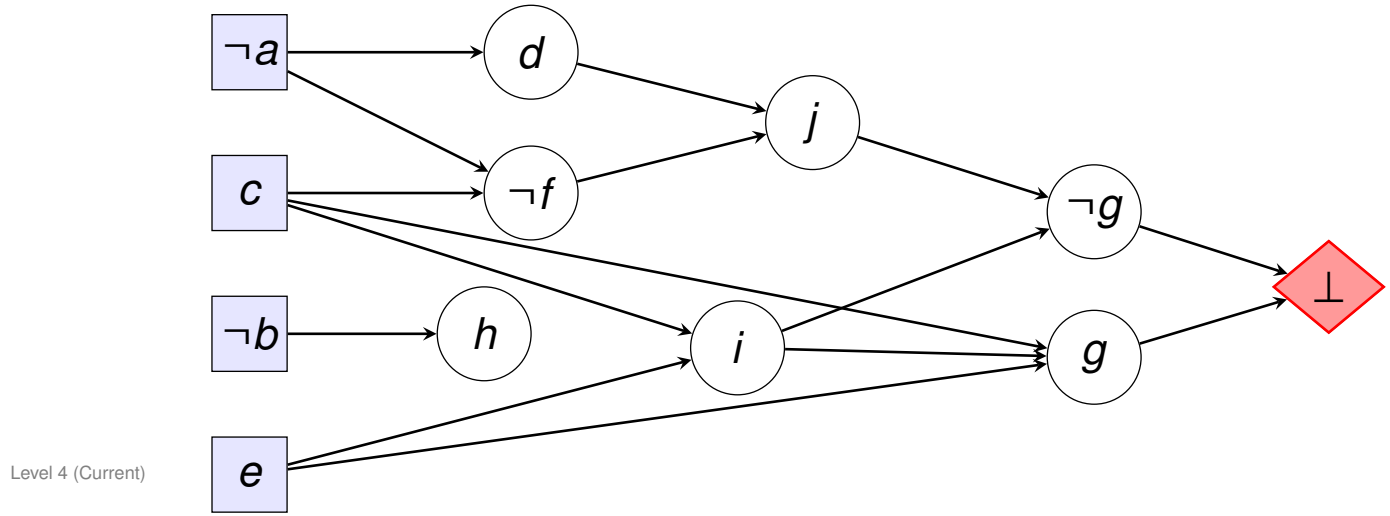
Conflict Analysis & Clause Learning



Resolution Trace (Deriving δ)

Conflict detected at level 4.

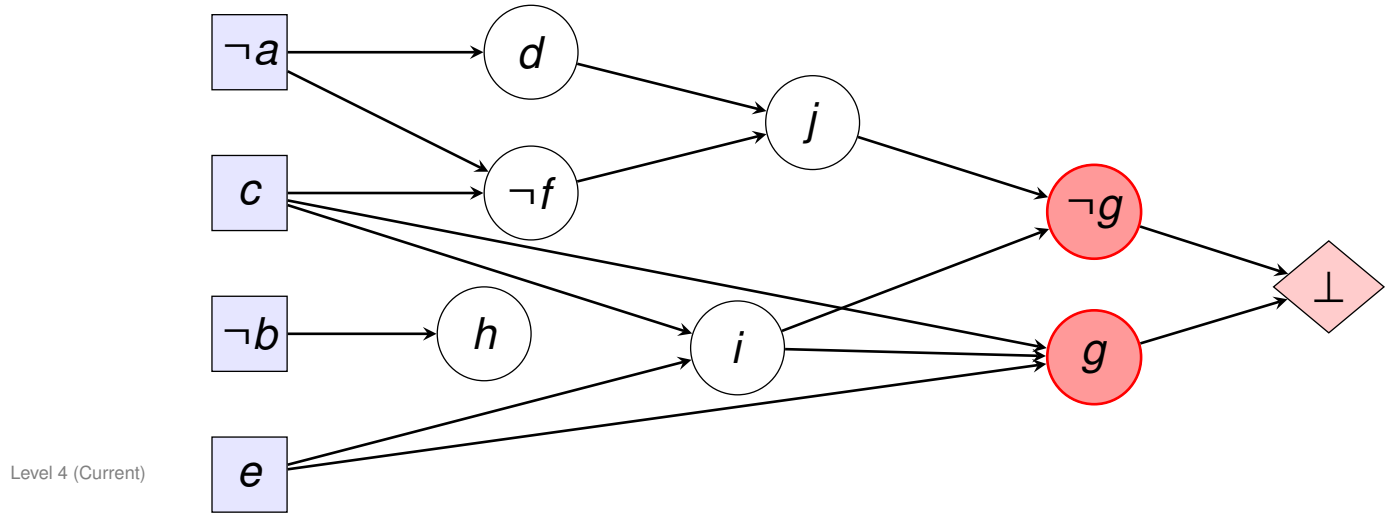
Conflict Analysis & Clause Learning



Resolution Trace (Deriving δ)

Start with conflict: $\bot$

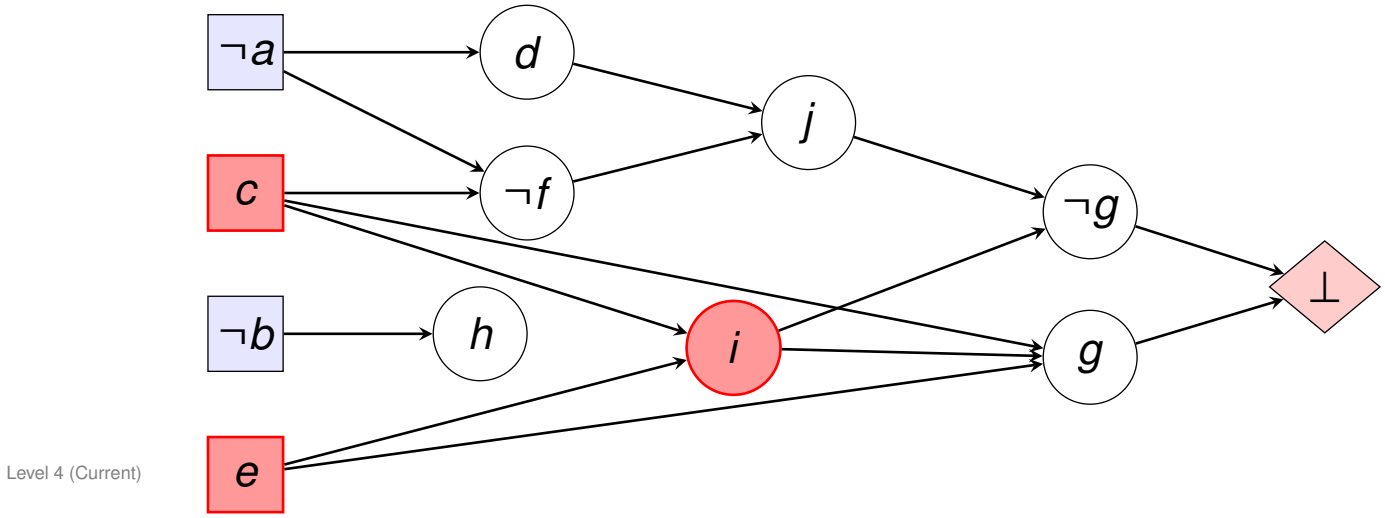
Conflict Analysis & Clause Learning



Resolution Trace (Deriving δ)

Resolve on g: $g \vee \neg g$

Conflict Analysis & Clause Learning

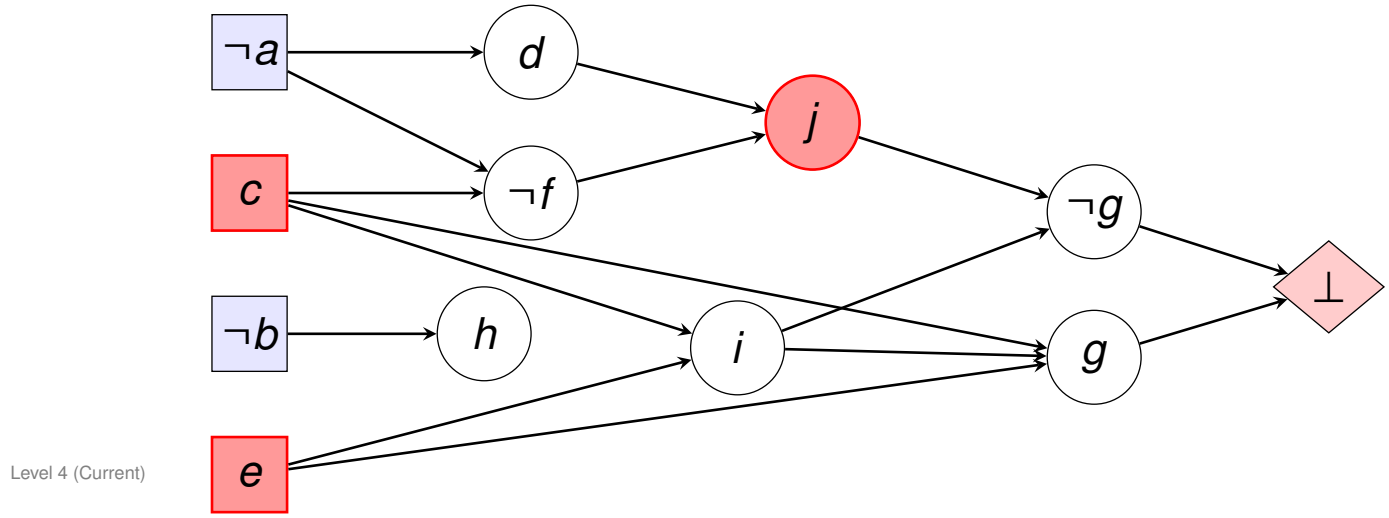


Resolution Trace (Deriving δ)

Resolve inputs of g: $\neg c \vee \underbrace{\neg e \vee \neg \mathbf{i}}_{\text{Level 4}} \vee \neg j = \neg i \vee \neg j \vee \neg g \oplus \neg c \vee \neg e \vee \neg i \vee g$

(Two literals at Level 4: e, i. **Continue.**)

Conflict Analysis & Clause Learning



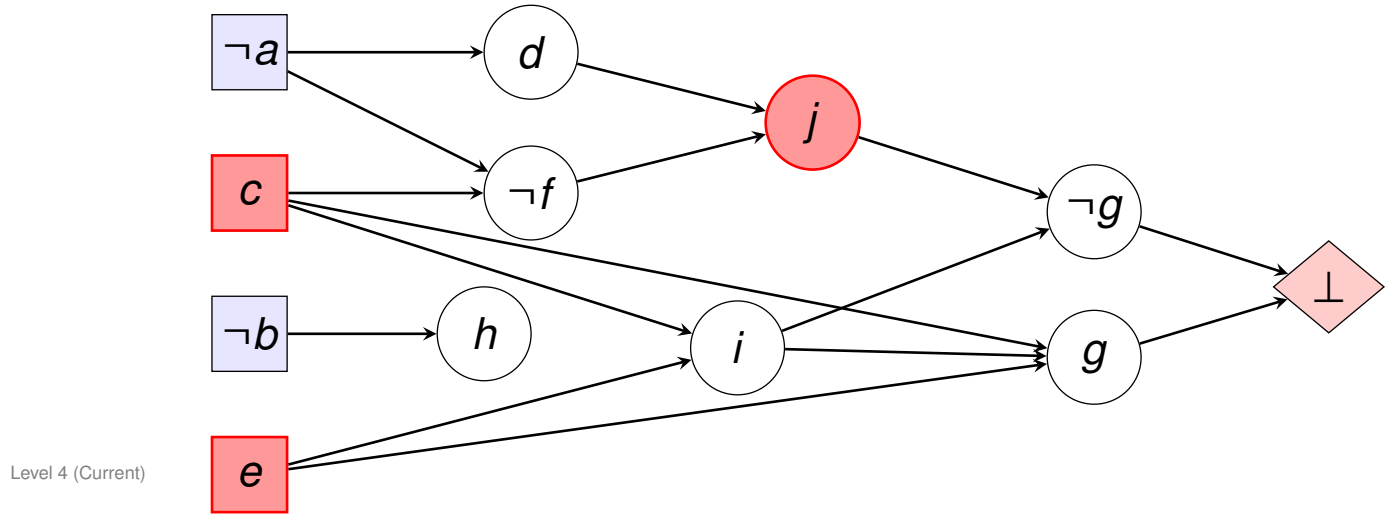
Resolution Trace (Deriving δ)

Resolve on i: $\neg c \vee \neg e \vee \neg j = \neg c \vee \neg e \vee \neg i \vee \neg j \oplus \neg c \vee \neg e \vee i$

(**Stop:** Only one literal at Level 4: e)

- **1-UIP Condition:** Stop when the resolvent contains exactly *one literal* from the current decision level (here e).

Conflict Analysis & Clause Learning



Resolution Trace (Deriving δ)

Resolve on i: $\neg c \vee \neg e \vee \neg j = \neg c \vee \neg e \vee \neg i \vee \neg j \oplus \neg c \vee \neg e \vee i$

(**Stop:** Only one literal at Level 4: e)

- **1-UIP Condition:** Stop when the resolvent contains exactly *one literal* from the current decision level (here e).
- **Learning:** Add clause $\delta = (\neg c \vee \neg e \vee \neg j)$ to the database.

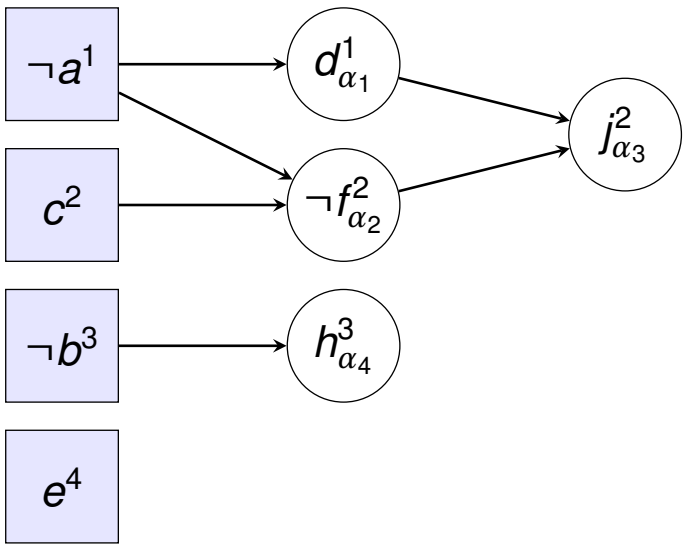
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^4$

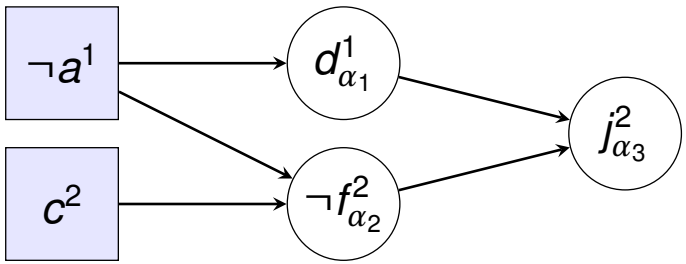
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e$

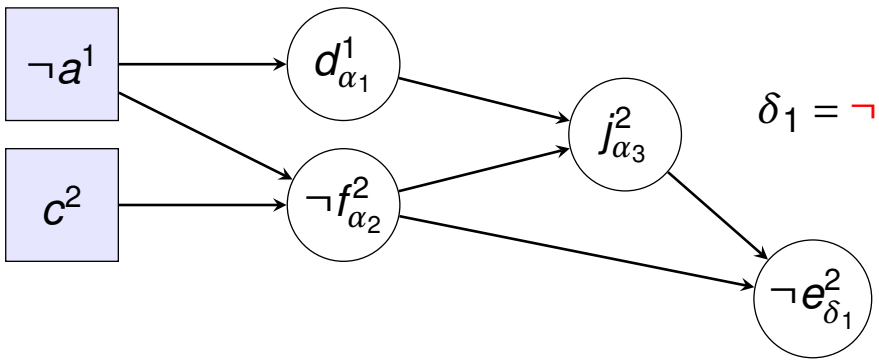
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$

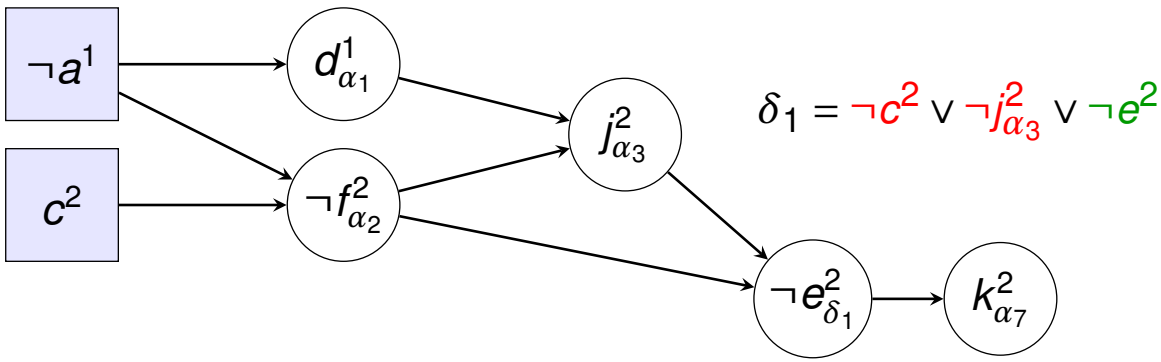
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



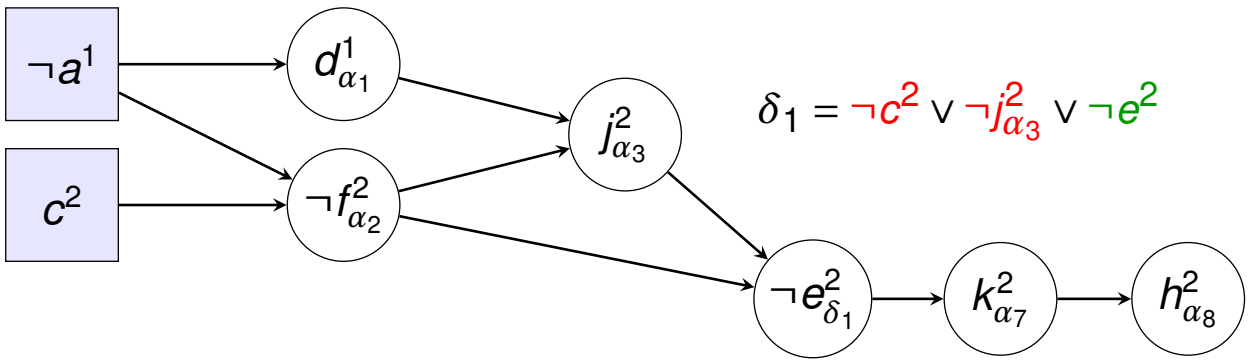
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



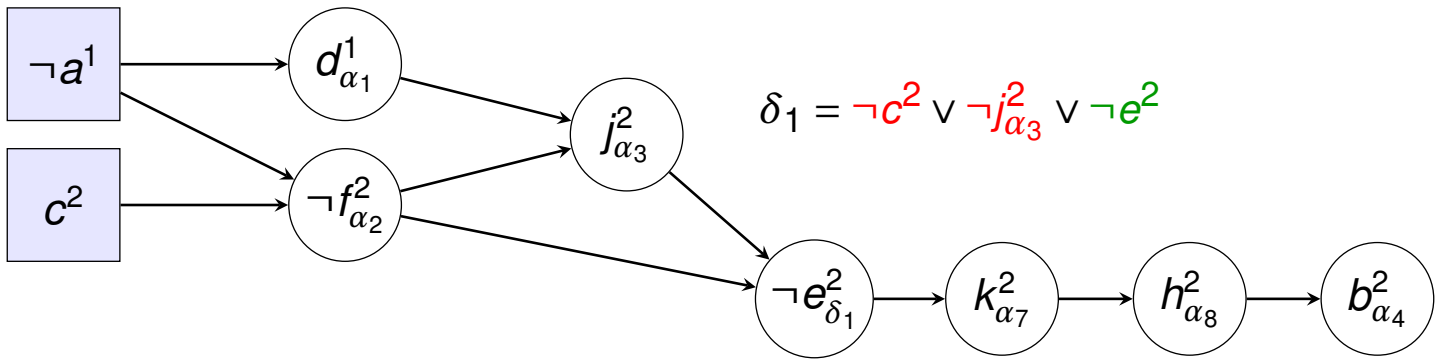
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



$\delta_1 = \neg c^2 \vee \neg j^2_{\alpha_3} \vee \neg e^2$

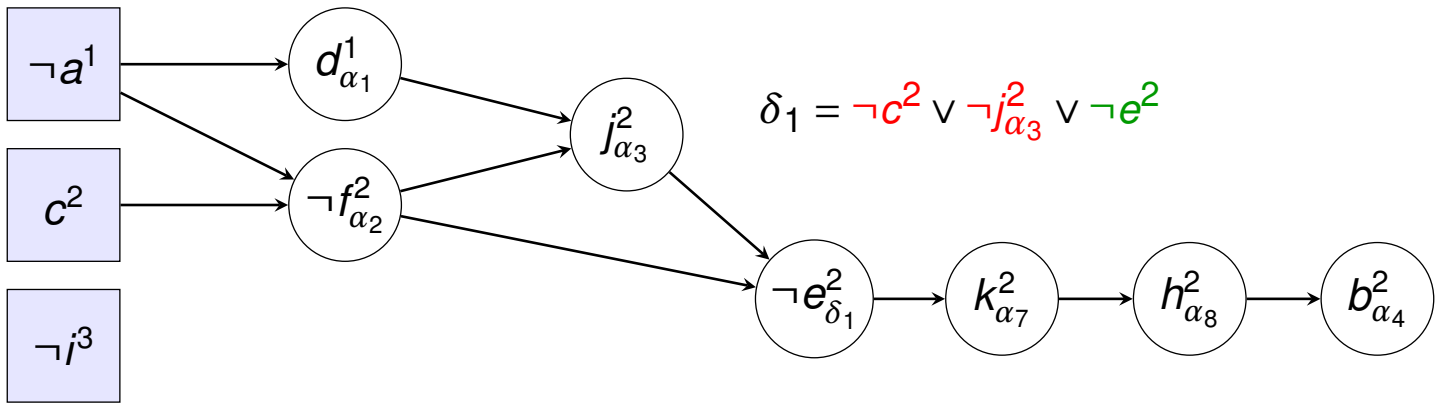
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$



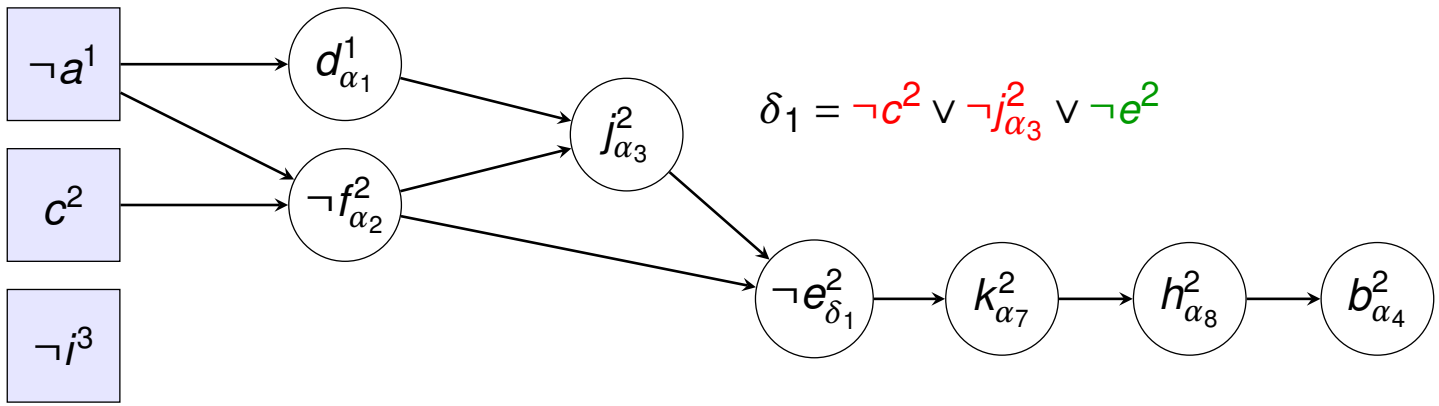
Backjumping

Current Formula State

$\alpha_1 : a \vee d$ $\quad \alpha_2 : a \vee \neg c \vee \neg f$ $\quad \alpha_3 : \neg d \vee j \vee f$

$\alpha_4 : b \vee h$ $\quad \alpha_5 : \neg c \vee \neg e \vee i$ $\quad \alpha_6 : \neg i \vee \neg j \vee \neg g$

$\alpha_7 : e \vee \neg k$ $\quad \alpha_8 : e \vee \neg h \vee k$ $\quad \alpha_9 : \neg c \vee \neg e \vee \neg i \vee g$

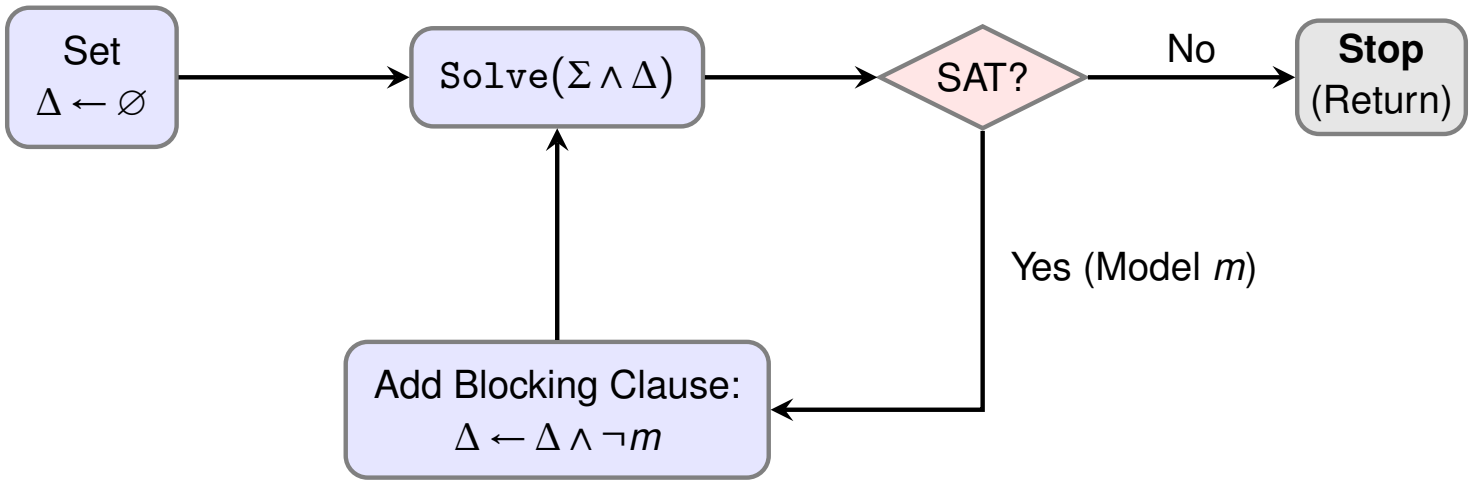


SATISFIABILITY PROVED

Outline

Generating MODS: The All-SAT Loop

- A naive Knowledge Compilation language is MODS: explicitly listing all satisfying assignments.
- We can compute this incrementally using a standard SAT solver (Audemard et al., 2013).



- *Note:* A blocking clause $\neg m$ rules out the model m so the solver is forced to find a new one.

KC Map for MODS: Queries

$\mathscr{L}$	CO	VA	CE	IM	EQ	SE	CT	ME
Circ	○	○	○	○	○	○	○	○
CNF	○	√	○	√	○	○	○	○
DNF	√	○	√	○	○	○	○	√
MODS	√	√	√	√	√	√	√	√

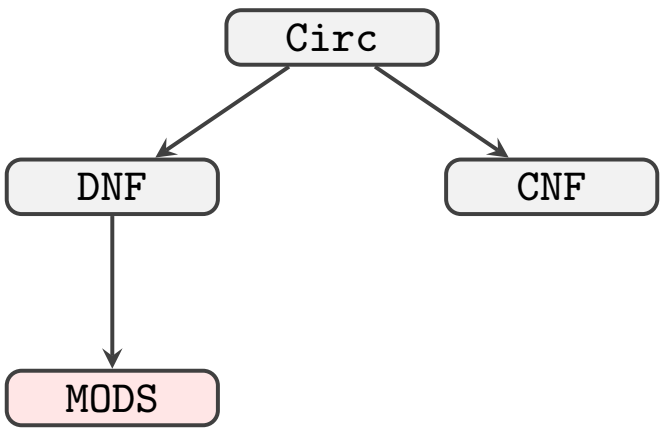
- **Notice:** MODS is the only language here that trivially supports every single query in polynomial time.

KC Map for MODS: Transformations

$\mathscr{L}$	**CD**	**FO**	**SFO**	**$\wedge$C**	**$\wedge$BC**	**$\vee$C**	**$\vee$BC**	**$\neg$C**
Circ	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
CNF	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$
DNF	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$	$\surd$	$\circ$
MODS	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$

- **The Trade-off:** While its queries are trivial, MODS loses significant transformation capabilities compared to general circuits (e.g., no closure under general $\wedge$ or $\vee$).

The Achilles Heel of MODS: Succinctness



Why is MODS strictly less succinct than DNF?

A single term in DNF (e.g., x_1) can implicitly represent 2^{n-1} models. MODS forces us to list them all.

A Disastrous Example:
Can we compile this simple formula efficiently into MODS?

$$\Sigma = \bigvee_{i=1}^{n} x_i$$

- **Size in Circ/ CNF / DNF:** $\mathcal{O}(n)$
- **Size in MODS:** $2^n - 1$ models!

Conclusion: MODS provides ultimate tractability, but zero compression. We need an intermediate language.

Outline

Between MODS and DNF: Orthogonal DNF (ODNF)

- DNF is succinct but fails at Model Counting (**CT**) because terms can overlap.
- MODS makes **CT** easy, but has zero compression (lists every single assignment).

Orthogonal DNF (ODNF)

A formula in DNF ($\bigvee_{i=1}^{m} t_i$) where all terms are **mutually exclusive** (orthogonal):

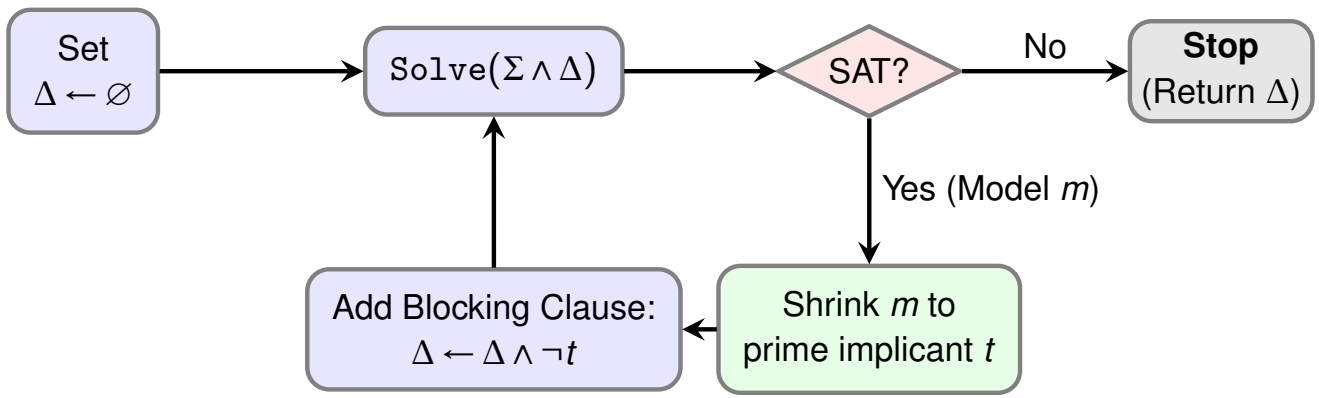
$$t_i \wedge t_j \models \bot \quad \text{for all } i \neq j$$

Why it works for Model Counting:

- Because terms do not overlap, the total number of models is exactly the sum of the models of each term.
- A single term t of length k (over n variables) compactly represents 2^{n-k} models.
- *Example:* If $n = 10$, the term $(x_1 \wedge \neg x_2)$ represents $2^8 = 256$ models perfectly, replacing 256 lines of MODS!

Compiling ODNF from CNF

We can adapt our MODS generator. Instead of blocking a full model m, we shrink m into a **Prime Implicant** t and block that instead! (Audemard et al., 2013; Spallitta et al., 2024)



- **Why is it Orthogonal?** The clause $\neg t$ forces the next implicant to contradict t.
- **Why is shrinking fast?** Because Σ is in CNF! Checking if a reduced term t is still an implicant ($t \models \Sigma$) simply requires checking if t contains at least one literal from every clause in Σ.

The KC Map: Queries for ODNF

$\mathscr{L}$	CO	VA	CE	IM	EQ	SE	CT	ME
Circ	∘	∘	∘	∘	∘	∘	∘	∘
CNF	∘	√	∘	√	∘	∘	∘	∘
DNF	√	∘	√	∘	∘	∘	∘	√
MODS	√	√	√	√	√	√	√	√
ODNF	√	√	√	√	√	√	√	√

- **The Breakthrough:** By simply forcing the terms of a DNF to be mutually exclusive, Model Counting (**CT**) flips from ∘ to √!
- *Why?* Because $\|t_1 \vee t_2\| = \|t_1\| + \|t_2\|$ is only mathematically true if $t_1 \wedge t_2 \models \bot$.

The KC Map: Transformations for ODNF

$\mathscr{L}$	**CD**	**FO**	**SFO**	**∧C**	**∧BC**	**∨C**	**∨BC**	**¬C**
Circ	√	○	√	√	√	√	√	√
CNF	√	○	√	√	√	○	√	○
DNF	√	√	√	○	√	√	√	○
MODS	√	√	√	○	√	○	○	○
ODNF	√	○	○	○	√	○	○	○

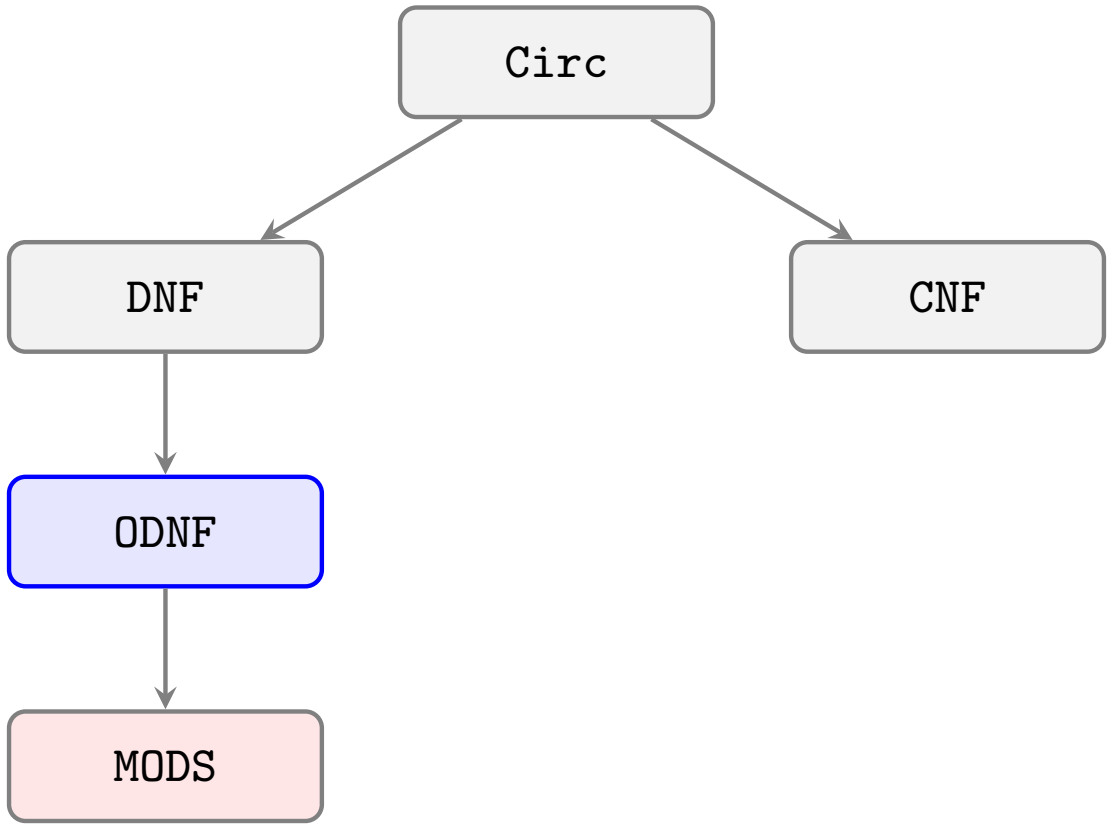
Why is Singleton Forgetting (**SFO**) strictly ○?

It is not just that naive erasure fails; the operation mathematically forces an **exponential blow-up** in the representation size.

- Logically, forgetting a variable x is defined as: $\exists x.\Sigma \equiv (\Sigma \mid x) \vee (\Sigma \mid \neg x)$.
- Conditioning (**CD**) is √, so $(\Sigma \mid x)$ and $(\Sigma \mid \neg x)$ are both valid, compact ODNFs.
- **The Proof:** We must combine them with an OR (∨**C**). Making two arbitrary ODNFs orthogonal to each other requires resolving all their intersections. In the worst case, this shatters the terms and causes an **exponential explosion**. Thus, no poly-time algorithm can exist.

Succinctness: Where does ODNF fit?

- ODNF sits perfectly between the "too relaxed" DNF and the "too strict" MODS.



- $\texttt{DNF} <_s \texttt{ODNF}$: Making terms orthogonal can require an exponential blow-up in the number of terms.
- $\texttt{ODNF} <_s \texttt{MODS}$: A single ODNF term of size k compactly represents 2^{n-k} models, avoiding the need to list them all.

Outline

Taking Advantage of the Solver's Trace

- When a SAT solver evaluates a CNF instance Σ, it systematically explores the search space of interpretations until it finds a model (or proves unsatisfiability).
- **The KC Insight:** We can use this *exact same search space* to compile Σ. We just prevent the solver from stopping when it finds the first model.
- By recording the trace of the solver's decisions, we generate a compiled structural form of the formula.