Resolution Algorithm

function PL-RESOLUTION(KB, α) **returns** $true$ or $false$

inputs: KB, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
$new \leftarrow \{\,\}$
loop do
  for each pair of clauses C_i, C_j **in** $clauses$ **do**
    $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
    if $resolvents$ contains the empty clause **then return** $true$
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clauses$ **then return** $false$
  $clauses \leftarrow clauses \cup new$

Resolution Algorithm

Factoring :

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

$2n$ $\qquad$ 2^{2n} $\qquad$ n $\qquad$ $(a \vee \neg a \vee b \vee \ldots)$

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

2^{2n}

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

2^{2n}

Can we ignore clauses that are tautology?

$C_1 \wedge C_2 \wedge \ldots \wedge (a \vee \neg a \vee b)$

iff $C_1 \wedge C_2 \wedge \ldots$

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

$$2^{2n}$$

Can we ignore clauses that are tautology?

What happens when we can resolve two clauses in more than one way?

$$\frac{\neg a \vee b \vee c, \qquad a \vee \neg b \vee d}{}$$

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

2^{2n}

Can we ignore clauses that are tautology?

What happens when we can resolve two clauses in more than one way?

$$\frac{\neg a \vee b \vee c, \qquad a \vee \neg b \vee d}{b \vee c \vee \neg b \vee d}$$

A more efficient algorithm

- SAT is NP-complete.
- Can we come up with a more efficient algorithm by making some assumptions?
- Definite clause
- Horn clause

$\neg a \vee \neg b$ $\neg a \vee b$ $a \vee b$

Forward Chaining Example

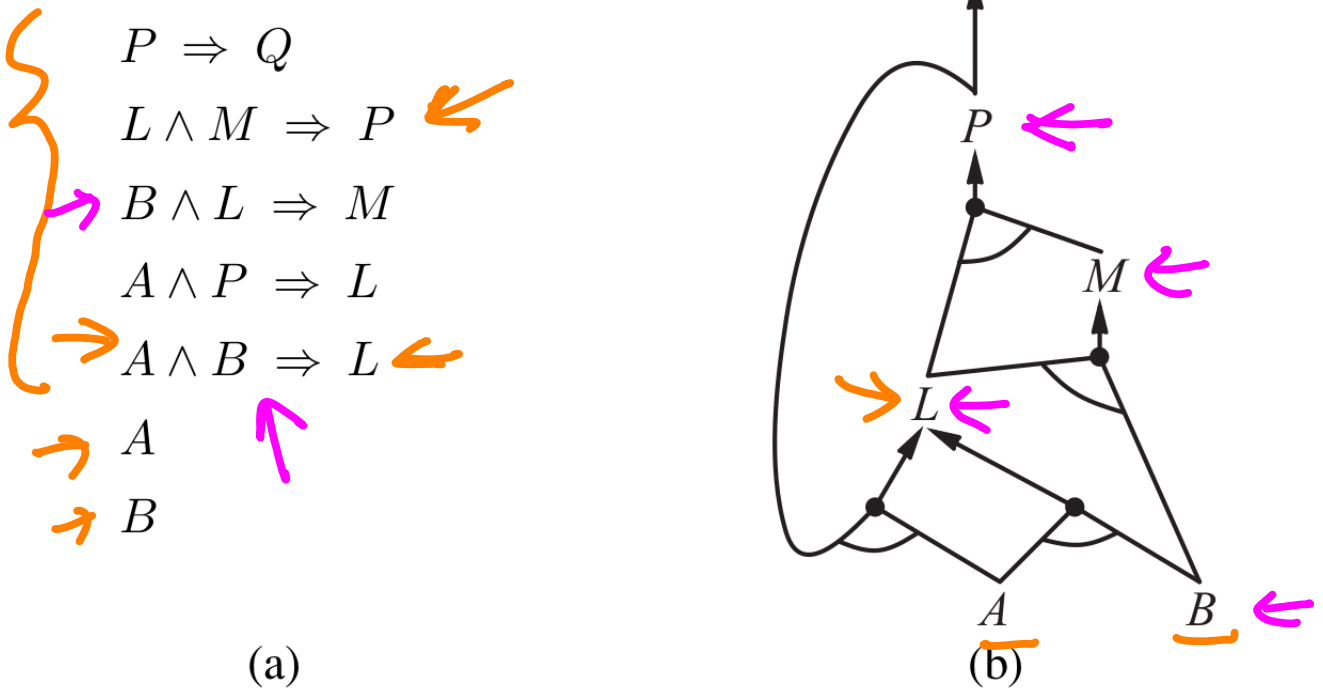
$a_1 \wedge a_2 \wedge \ldots a_n \Rightarrow b$

$T_1 \wedge T_2 \wedge S \Rightarrow D$ ⟸

⇓

$(\neg T_1 \vee \neg T_2 \vee \neg S \vee D)$

Forward Chaining Example



(a) A set of Horn clauses. (b) The corresponding AND–OR graph.

Forward Chaining Algorithm

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*

inputs: KB, the knowledge base, a set of propositional definite clauses
q, the query, a proposition symbol

$count \leftarrow$ a table, where $count[c]$ is the number of symbols in c's premise

$inferred \leftarrow$ a table, where $inferred[s]$ is initially *false* for all symbols

$agenda \leftarrow$ a queue of symbols, initially symbols known to be true in KB

while $agenda$ is not empty **do**
  $p \leftarrow$ POP($agenda$)
  if $p = q$ **then return** *true*
  if $inferred[p] = false$ **then**
    $inferred[p] \leftarrow true$
    for each clause c in KB where p is in c.PREMISE **do**
      decrement $count[c]$
      if $count[c] = 0$ **then** add c.CONCLUSION to $agenda$
return *false*

Forward Chaining Algorithm

- Time complexity? $O(ns)$

Forward Chaining Algorithm

KB

- Time complexity?
- Is it sound?

q

Forward Chaining Algorithm

$KB \models a$

- Time complexity?
- Is it sound?
- Is it complete?

Backward Chaining Algorithm

$KB \vdash q$

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

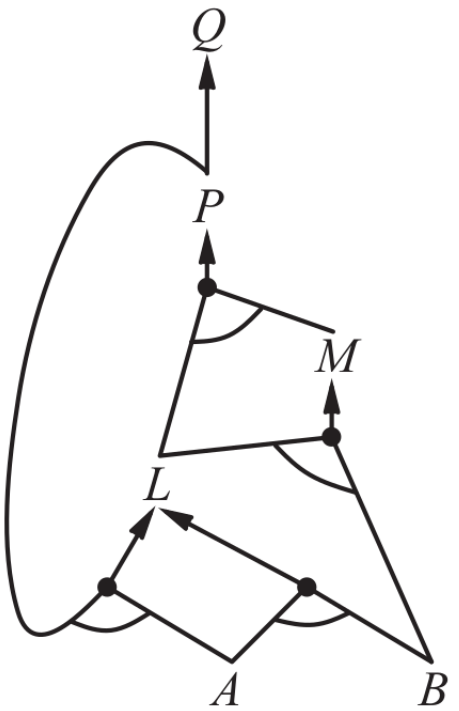
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

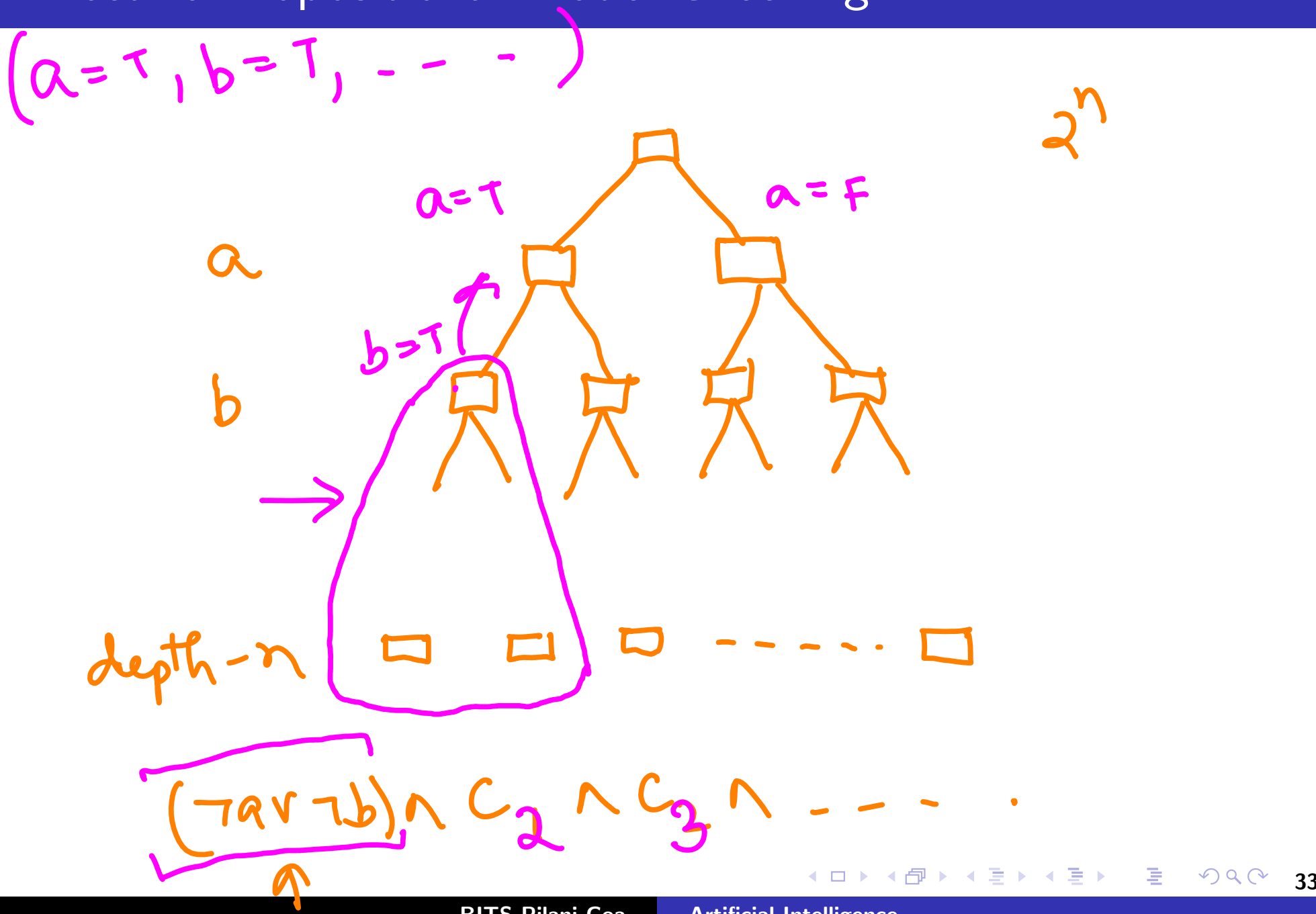
(a)



(b)

(a) A set of Horn clauses. (b) The corresponding AND–OR graph.

Effective Propositional Model Checking



Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF

Output : Is the sentence satisfiable?

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF

Output : Is the sentence satisfiable?

- Early termination

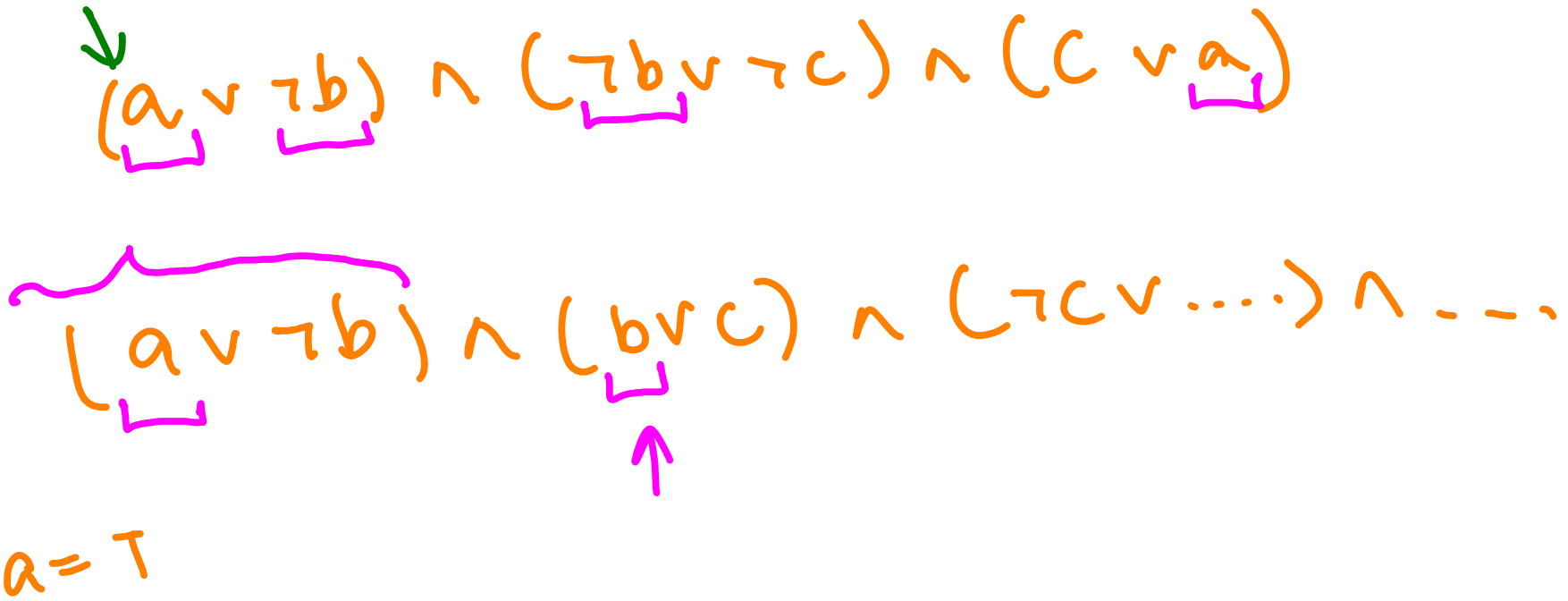
$a = T, b = T$

$(a \vee b) \vee (a \vee c)$

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic



Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic
- Unit clause heuristic