

PROPOSITIONAL LOGIC 2

ARTIFICIAL INTELLIGENCE | COMP 131

- Automated reasoning
- Efficient satisfiability
- Questions?

Logical inference is used to create new sentences that logically follow from a given knowledge base.

The most used inference rules:

RULE	PREMISE	CONCLUSION
Modus Ponens	p, p o q	q
AND elimination	$p \wedge q$	p, q
Double negation	$\neg \neg p$	p
Unit resolution	$p \lor q, \neg q$	p
AND introduction	p,q	$p \wedge q$
Modus Tollens	$\neg q, p \rightarrow q$	$\neg p$

- There are two directions of search: forward and backward chaining.
- There is also the DPLL, a complete algorithm for deciding if a sentence is satisfiable.

- 1 person_in_front_of_car \rightarrow brake
- 2 (yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Forward chaining: answer queries using a knowledge base to determine new facts until we find that our query is **true**, or until we've run out of new facts to generate.

QUERY

Do we need to *Brake*?

INFERENCE

Automated reasoning

FORWARD CHAINING

- 1 person_in_front_of_car → brake
- 2 (yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light → brake
- 7 winter \rightarrow snow

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Forward chaining: answer queries using a knowledge base to determine new facts until we find that our query is **true**, or until we've run out of new facts to generate.

QUERY

Do we need to *Brake*?

INFERENCE

KNOWN police_car

MP R3 police_car → policeman

policeman

FORWARD CHAINING

- 1 person_in_front_of_car → brake
- 2 (yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ **dry** ∧ police_car ∧ ¬person_in_front_of_car

Forward chaining: answer queries using a knowledge base to determine new facts until we find that our query is **true**, or until we've run out of new facts to generate.

Automated reasoning

FORWARD CHAINING

QUERY

Do we need to *Brake*?

INFERENCE

KNOWN police_car

MP R3 police_car → policeman

policeman

KNOWN dry

MT R5 slippery $\rightarrow \neg dry$

DN $\neg\neg dry \rightarrow \neg slippery$

MP $dry \rightarrow \neg slippery$

¬slippery

- 1 person_in_front_of_car → brake
- 2 (yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car → policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

FACTS

yellow_light $\land \neg red_light \land \neg snow \land dry \land police_car \land \neg person_in_front_of_car$

Forward chaining: answer queries using a knowledge base to determine new facts until we find that our query is **true**, or until we've run out of new facts to generate.

QUERY

Do we need to *Brake*?

INFERENCE

KNOWN police_car

MP R3 police_car → policeman

policeman

KNOWN dry

MT R5 slippery $\rightarrow \neg dry$

DN $\neg\neg dry \rightarrow \neg slippery$

MP $dry \rightarrow \neg slippery$

¬slippery

KNOWN yellow_light

KNOWN policeman

KNOWN ¬slippery

MP R2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake

- 1 person_in_front_of_car → brake
- 2 (yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Forward chaining: answer queries using a knowledge base to determine new facts until we find that our query is **true**, or until we've run out of new facts to generate.

QUERY

Do we need to *Brake*?

INFERENCE

KNOWN police_car

MP R3 police_car → policeman

policeman

KNOWN dry

MT R5 slippery $\rightarrow \neg dry$

DN $\neg\neg dry \rightarrow \neg slippery$

MP $dry \rightarrow \neg slippery$

¬slippery

KNOWN yellow_light

KNOWN policeman
KNOWN ¬slippery

MP R2 ((yellow_light V policeman) $\land \neg$ slippery) \rightarrow brake

CONCLUSION brake

Backward chaining: an approach alternative to forward chaining in which the query is **explicitly proven** with the given knowledge and work **backward** until all the facts are known.

KNOWLEDGE BASE

- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning

brake

KNOWLEDGE BASE

- 1 person_in_front_of_car \rightarrow brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car → policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning

brake

KNOWLEDGE BASE

- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
 - 3 police_car → policeman
 - 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
 - 7 winter \rightarrow snow

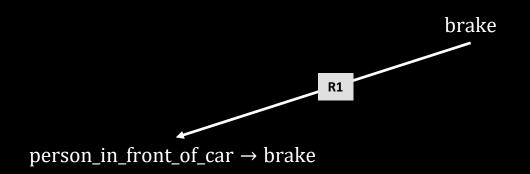
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car





- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
 - 3 police_car → policeman
 - 4 snow \rightarrow slippery
 - 5 slippery $\rightarrow \neg dry$
- 6 red_light → brake
 - 7 winter \rightarrow snow

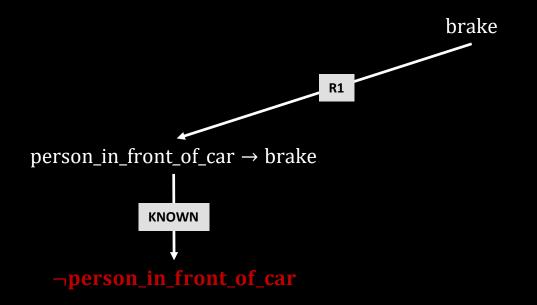
Automated reasoning

QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car



- 1 person_in_front_of_car → brake
 - 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
 - 3 police_car \rightarrow policeman
 - 4 snow \rightarrow slippery
 - 5 slippery $\rightarrow \neg dry$
- 6 red_light → brake
 - 7 winter \rightarrow snow

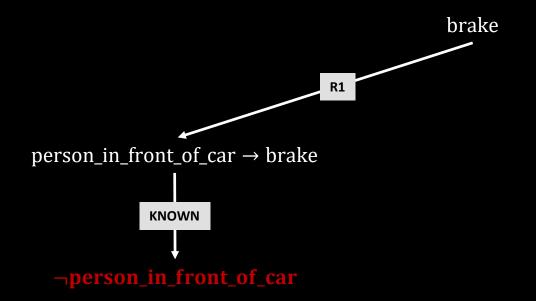
QUERY

Do we need to *Brake*?

FACTS

yellow_light $\land \neg red_light \land \neg snow \land dry \land police_car \land \neg person_in_front_of_car$

Automated reasoning



- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
 - 7 winter \rightarrow snow

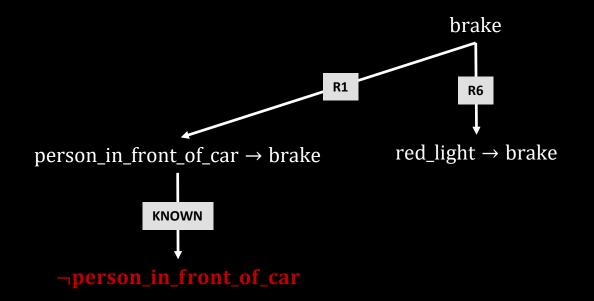
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



- 1 person_in_front_of_car \rightarrow brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
 - 7 winter \rightarrow snow

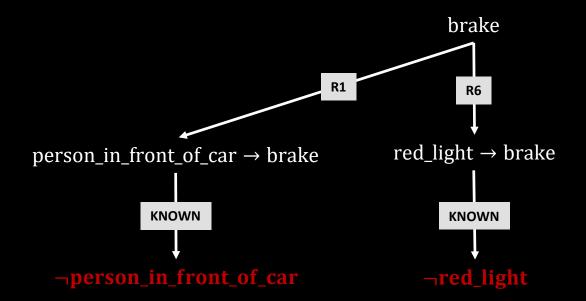
FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

QUERY

Do we need to *Brake*?

Automated reasoning
BACKWARD CHAINING



- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light → brake
- 7 winter \rightarrow snow

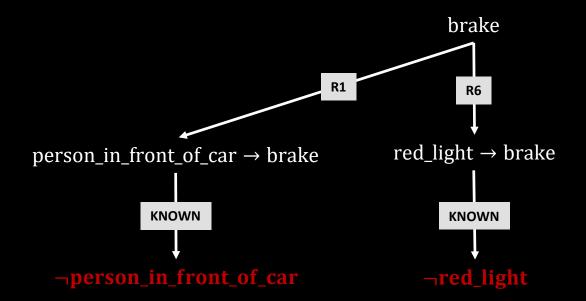
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light → brake
- 7 winter \rightarrow snow

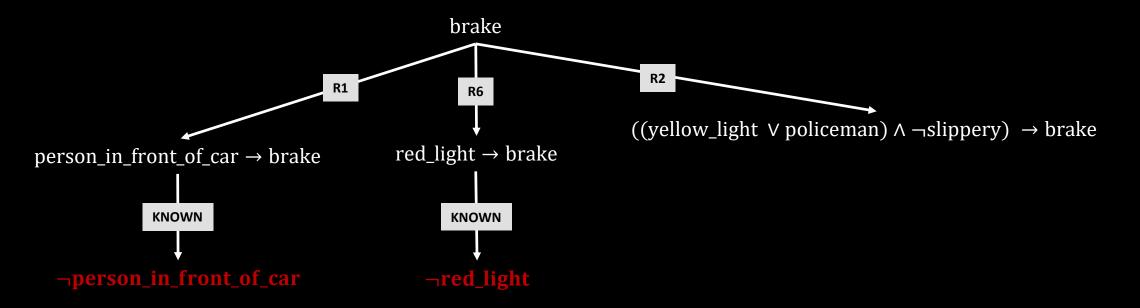
QUERY

Do we need to *Brake*?

FACTS

yellow_light $\land \neg red_light \land \neg snow \land dry \land police_car \land \neg person_in_front_of_car$

Automated reasoning BACKWARD CHAINING



- 1 person_in_front_of_car \rightarrow brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

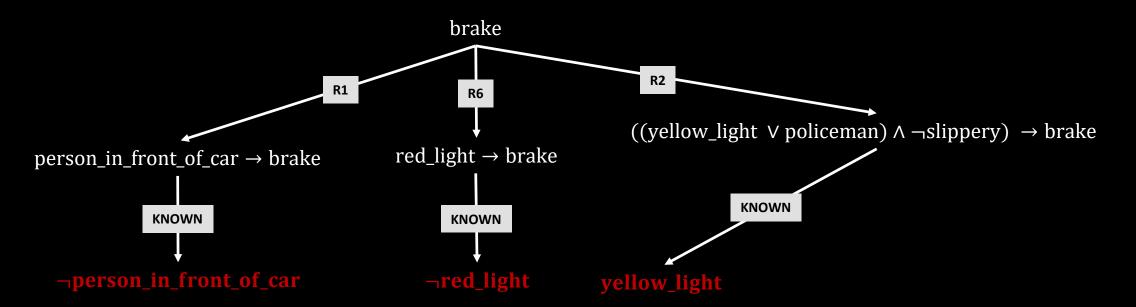
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



- 1 person_in_front_of_car → brake
- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

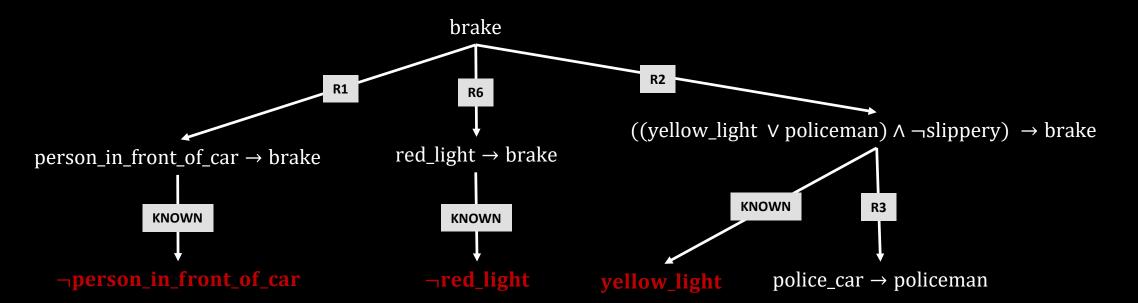
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



- 1 person_in_front_of_car → brake
- 2 ((yellow_light ∨ policeman) ∧ ¬slippery) → brake
- 3 police_car → policeman
 - 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

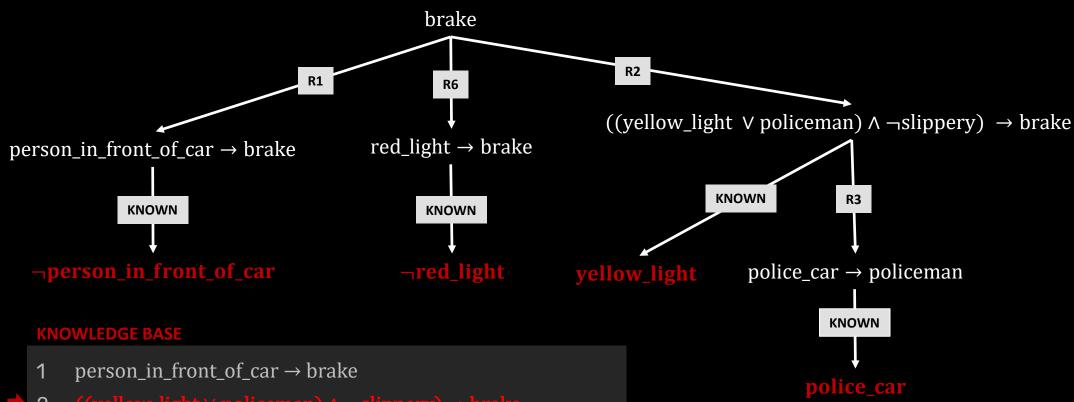
QUERY

Do we need to *Brake*?

FACTS

yellow_light $\land \neg red_light \land \neg snow \land dry \land police_car \land \neg person_in_front_of_car$

Automated reasoning



- 2 ((yellow_light ∨ policeman) ∧ ¬slippery) → brake
- 3 police_car → policeman
 - 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
- 6 red_light \rightarrow brake
- 7 winter \rightarrow snow

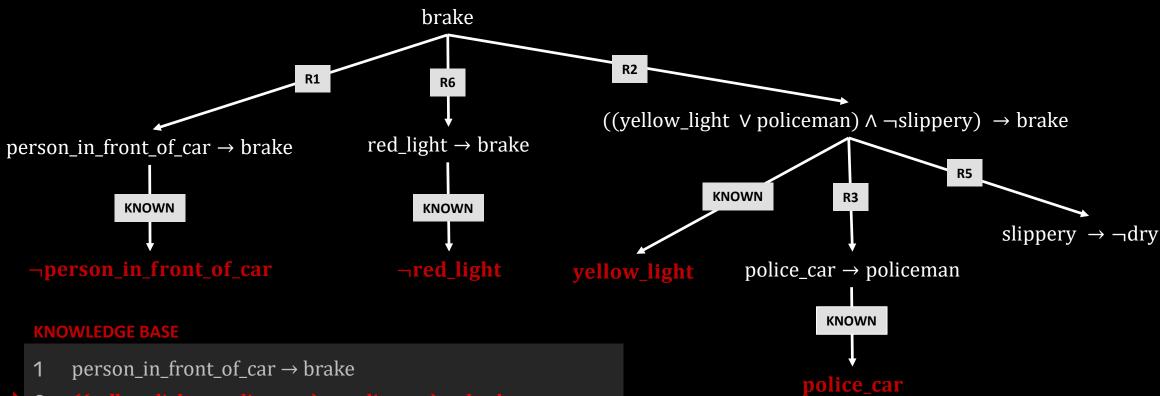
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



- 2 ((yellow_light \lor policeman) $\land \neg$ slippery) \rightarrow brake
 - 3 police_car \rightarrow policeman
 - 4 snow \rightarrow slippery
- 5 slippery $\rightarrow \neg dry$
 - 6 red_light \rightarrow brake
 - 7 winter \rightarrow snow

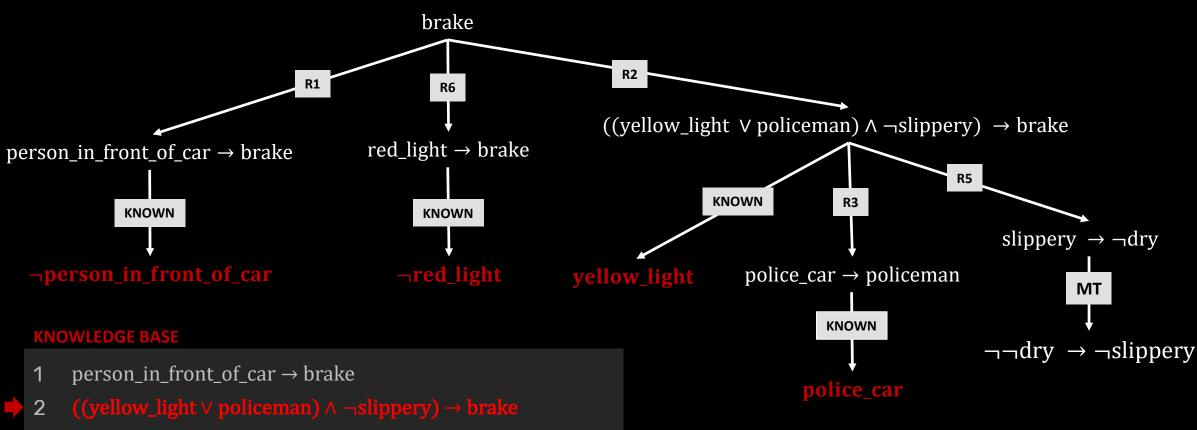
QUERY

Do we need to *Brake*?

FACTS

yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

Automated reasoning



QUERY

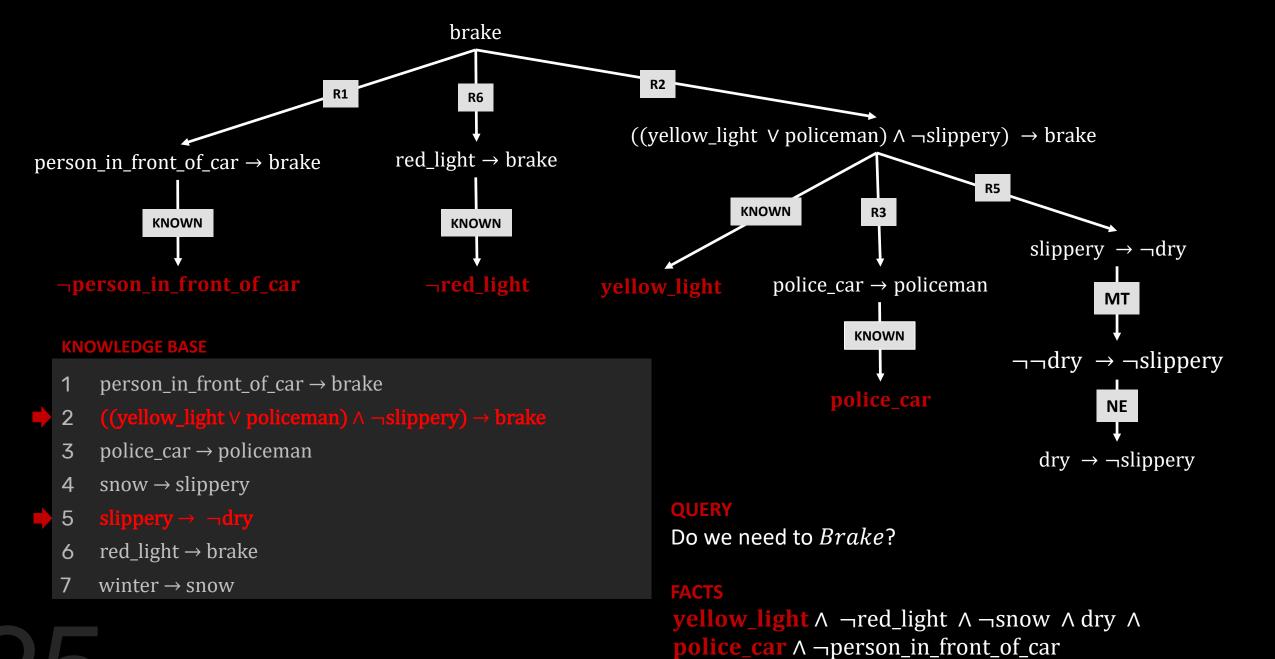
Do we need to *Brake*?

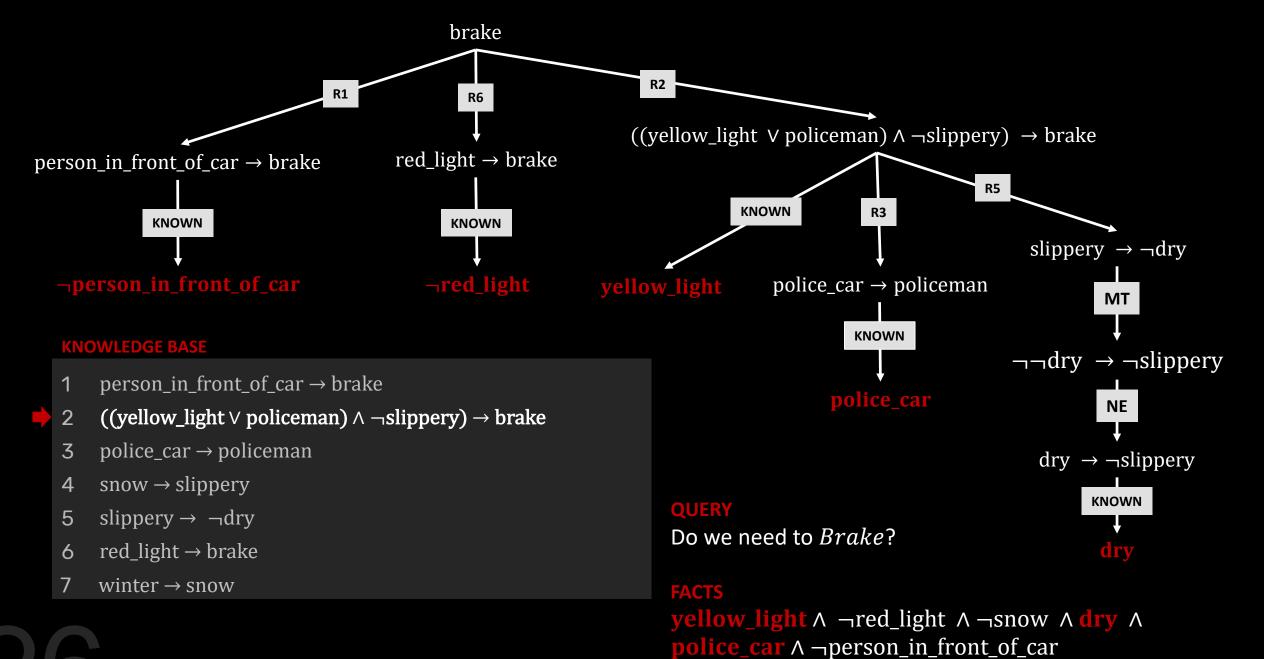
FACTS

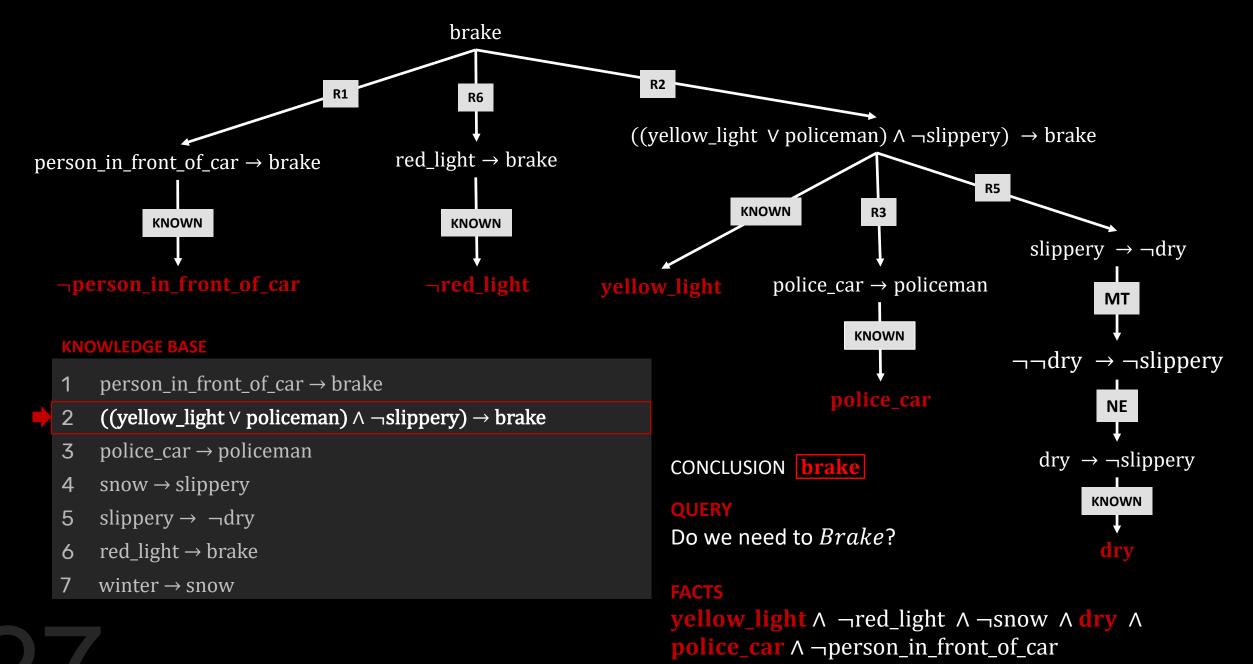
yellow_light ∧ ¬red_light ∧ ¬snow ∧ dry ∧
police_car ∧ ¬person_in_front_of_car

- 3 police_car \rightarrow policeman
- 4 snow \rightarrow slippery
- \bullet 5 slippery $\rightarrow \neg dry$
 - 6 red_light \rightarrow brake
 - 7 winter \rightarrow snow

Automated reasoning







Efficient satisfiability

The Davis-Putman-Logemann-Loveland is a complete search algorithm for deciding if sentences are satisfiable:

- It uses Depth-First Search for backtracking
- DPLL requires that the knowledge base is represented in a CNF form
- It uses improvements to shorten the search:
 - Early possible termination
 - Pure symbol heuristic: the symbol appears only with one polarity (T or F)
 - Unit clause heuristic: the symbol appears alone in a sentence

DPLL(*C* , *S* , *M*):

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return F
- 3. If there is a (t, polarity v) = pure symbol(C), return $\text{DPLL}(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;
- 6. Return $DPLL(C, R, M \cup \{P = \mathbf{T}\})$ V $DPLL(C, R, M \cup \{P = \mathbf{F}\})$

$$S = \{s, r, q, p\} M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

 $p \vee \neg q$

$$p \lor q \lor r \lor s \land \land \neg p \lor q \lor \neg r \land \land \land \land p \lor \neg q \lor r \lor s \land \land q \lor \neg r \lor \neg s \land \land \land \neg p \lor \neg s \land \land \land \land$$

{ }

Λ

$$S = \{s, r, q, p\} M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

Λ

{ }

 $p \vee \neg q$

$$S = \{s, r, q, p\} \ M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return $\text{DPLL}(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

$$p \lor q \lor r \lor s \land$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \vee \neg q \vee r \vee s \wedge$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

$$S = \{s, r, q, p\} \ M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

$$p \lor q \lor r \lor s \land$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \lor \neg q \lor r \lor s \land$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

$$S = \{s, r, q, p\} \ M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

$$p \lor q \lor r \lor s \land \land \neg p \lor q \lor \neg r \land \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \lor \neg q \lor r \lor s \land$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

$$S = \{s, r, q, p\} \ M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$$P = \{s\} R = \{r, q, p\}$$

6. Return

DPLL(
$$C, R, M \cup \{P = \mathbf{T}\}$$
) V DPLL($C, R, M \cup \{P = \mathbf{F}\}$)

CLAUSES

$$p \lor q \lor r \lor s \land$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \lor \neg q \lor r \lor s \land$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

$$S = \{s, r, q, p\} M = \{\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

$$P = \{s\} R = \{r, q, p\}$$

6. Return

$$DPLL(C, R, M \cup \{P = \mathbf{T}\})$$

$$M \cup \{s = \mathbf{T}\}$$

 $\mathsf{DPLL}(\mathsf{C}, R, M \cup \{P = \mathbf{F}\})$

CLAUSES

$$p \vee q \vee r \vee s \wedge$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \lor \neg q \lor r \lor s \land$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \qquad \wedge$$

$$S = \{r, q, p\} \ M = \{s = \mathsf{T}\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return $\text{DPLL}(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

$$p \lor q \lor r \lor s \land$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \vee \neg q \vee r \vee s \wedge$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

{ }

 $\{s = T\}$



Automated reasoning

$$S = \{r, q, p\} \ M = \{s = \mathbf{T}\}\$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

$$p \vee q \vee r \vee s \wedge$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land$$

$$p \vee \neg q \vee r \vee s \wedge$$

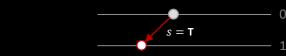
$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s \land$$

$$p \vee \neg q \wedge$$

{ }

 $\{s = T\}$



$$S = \{r, q, p\} \ M = \{s = \mathbf{T}\}\$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

CLAUSES

$p \lor q \lor r \lor s \land = T$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land = \mathsf{T}$$

$$p \lor \neg q \lor r \lor s \land = \mathsf{T}$$

$$q \vee \neg r \vee \neg s \wedge$$

$$\neg p \lor \neg s$$
 \land

$$p \lor \neg q \land$$

{ }

 $\{s = T\}$



Automated reasoning

$$S = \{r, q, p\} \ M = \{s = T\}$$

- If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return **F**
- If there is a (t, polarity v) = pure symbol(C), 3. return DPLL(C, S - t, $M \cup \{t = v\}$)
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL(C, S - u, $M \cup \{u = v\}$)
- P = first(S); R = rest(S);5.
- 6. Return $DPLL(C, R, M \cup \{P = T\})$ $DPLL(C, R, M \cup \{P = \mathbf{F}\})$

٨	= T
Λ	
٨	= T
٨	= T
٨	
	Λ Λ

Λ

Λ

{ }

 $\neg p$

 $p \lor \neg q$

 $\{s = T\}$



$$S = \{r, q, p\} \ M = \{s = T\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	Λ	
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	Λ	
$\neg p$	Λ	
$p \vee \neg q$	Λ	

 $\{s = T\}$



Automated reasoning

$$S = \{r, q, p\} \ M = \{s = T\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

LAUSES		
$p \lor q \lor r \lor s$	Λ	= T
$\neg p \lor q \lor \neg r$	Λ	
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	Λ	
$\neg p$	Λ	
$p \vee \neg q$	Λ	

{ }

 $\{s = T\}$



$$S = \{r, q, p\} \ M = \{s = \mathbf{T}\}\$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

p	V	q	V	$r \vee s$	Λ	= T
---	---	---	---	------------	---	-----

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land = \mathsf{T}$$

$$p \lor \neg q \lor r \lor s \land = \mathsf{T}$$

$$q \vee \neg r$$

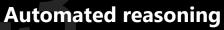
Λ

$$\neg p$$

 $p \vee \neg q$

$$\{s = T\}$$





$$S = \{r, q, p\} \ M = \{s = \mathbf{T}\}\$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a $(t, \text{ polarity } v) = \text{pure symbol}(C), \quad (r, \mathbf{F})$ return DPLL $(C, S - t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

$$p \lor q \lor r \lor s \qquad \land \qquad = \mathsf{T}$$
$$\neg p \lor q \lor \neg r \qquad \land$$

$$\neg q \lor \neg r \lor s \land = \mathsf{T}$$

$$p \lor \neg q \lor r \lor s \land = \mathsf{T}$$

$$q \vee \neg r \wedge$$

$$\neg p$$
 \wedge

$$p \vee \neg q \qquad \wedge$$

$$\{ \}$$
 $s = T$

$$S = \{r, q, p\} \ M = \{s = \mathbf{T}\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a $(t, \text{ polarity } v) = \text{pure symbol}(C), \quad (r, \mathbf{F})$ return DPLL $(C, S - t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$$p \lor q \lor r \lor s \land = T$$

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land = \mathsf{T}$$

$$p \lor \neg q \lor r \lor s \land = \mathsf{T}$$

$$q \vee \neg r \wedge$$

$$\neg p$$
 \wedge

$$p \vee \neg q \wedge$$

$$\{ \}$$
 $s = T$

$$\{s = T\}$$

$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

$p \lor q \lor r \lor s \land = 7$	Γ
------------------------------------	---

$$\neg p \lor q \lor \neg r \land$$

$$\neg q \lor \neg r \lor s \land = \mathsf{T}$$

$$p \lor \neg q \lor r \lor s \land = \mathsf{T}$$

Λ

$$q \vee \neg r$$

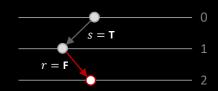
$$\neg p$$
 \wedge

$$p \vee \neg q \wedge$$

{ }

 $\{s = T\}$

 $\{s = T, r = F\}$



Automated reasoning

$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

 $p \lor q \lor r \lor s \land = T$

 $\neg p \lor q \lor \neg r$ \land

 $\neg q \lor \neg r \lor s \land = \mathsf{T}$

 $p \lor \neg q \lor r \lor s \land = \mathsf{T}$

Λ

q ∨ ¬**r**

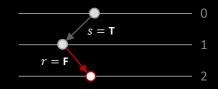
 $\neg p$ \wedge

 $p \vee \neg q \qquad \wedge$

{ }

 $\{s = T\}$

 $\{s = T, r = F\}$



$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return F
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

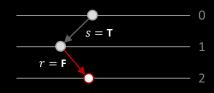
CLAUSES

CEAOSES		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$n \vee \neg a$	٨	

{ }

 $\{s = T\}$

 $\{s = T, r = F\}$



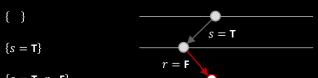
$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$p \lor q \lor r \lor s \qquad \land \qquad = \mathsf{T}$ $\neg p \lor q \lor \neg r \qquad \land \qquad = \mathsf{T}$ $\neg q \lor \neg r \lor s \qquad \land \qquad = \mathsf{T}$ $p \lor \neg q \lor r \lor s \qquad \land \qquad = \mathsf{T}$ $q \lor \neg r \qquad \qquad \land \qquad = \mathsf{T}$

Λ

Λ



 $p \vee \neg q$

 $\neg p$

$$S = \{q, p\} \ M = \{s = T, r = F\}$$

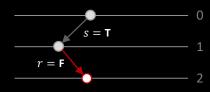
- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. P = first(S); R = rest(S);

CLAUSES

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	Λ	

 $\{s = T\}$

 $\{s = T, r = F\}$



$$S = \{q, p\} \ M = \{s = T, r = F\}$$

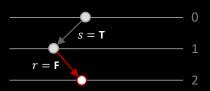
- If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL(C, S - t, $M \cup \{t = v\}$)
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL(C, S - u, $M \cup \{u = v\}$)
- P = first(S); R = rest(S);5.
- 6. Return $DPLL(C, R, M \cup \{P = T\})$ $DPLL(C, R, M \cup \{P = \mathbf{F}\})$

CLAUSES		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	٨	

{ }

 $\{s = T\}$

 $\{s = T, r = F\}$

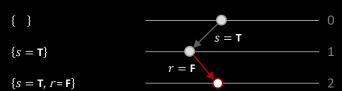


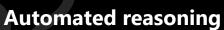
$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a $(t, \text{ polarity } v) = \text{pure symbol}(C), \quad (q, \mathbf{F})$ return DPLL $(C, S - t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUJLJ		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	٨	

$$p \lor \neg q$$
 \land





$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return $\text{DPLL}(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

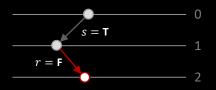
CLAUSES

 $p \vee \neg q$

CLAUSLS		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	

$$\{s = T\}$$

$$\{s = T, r = F\}$$

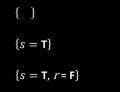


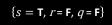
Λ

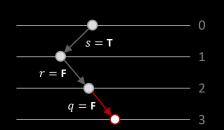
$$S = \{p\} \ M = \{s = T, r = F, q = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

LAUSLS		
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$n \vee \neg a$	۸	







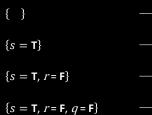
$$S = \{p\} \ M = \{s = T, r = F, q = F\}$$

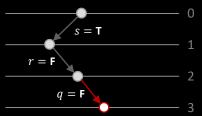
- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

CLAUSES

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	

$$p \vee \neg q$$





Λ

$$S = \{q, p\} \ M = \{s = T, r = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

CLAUSES

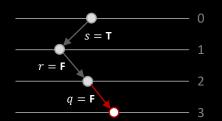
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	٨	= T

{ }

 $\{s = T\}$

 $\{s = T, r = F\}$

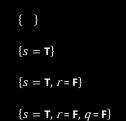
 $\{s = T, r = F, q = F\}$

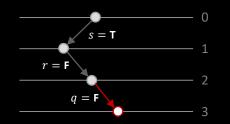


- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$S = \{p\} \ M = \{s = T, r = F, q = F\}$

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg a$	Λ	= T





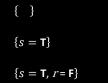
- If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return F
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL(C, S - t, $M \cup \{t = v\}$)
- If there is a (u, polarity v) = unit clause(C), 4. return DPLL(C, S - u, $M \cup \{u = v\}$)
- P = first(S); R = rest(S);5.
- 6. Return $DPLL(C, R, M \cup \{P = T\})$ $DPLL(C, R, M \cup \{P = \mathbf{F}\})$

$S = \{p\} \ M = \{s = T, r = F, q = F\}$

CLAUSES

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	٨	
$n \vee \neg a$	Λ	= T

s = T

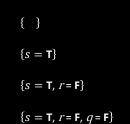


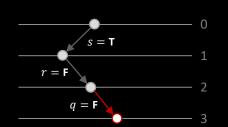


$$S = \{p\} \ M = \{s = T, r = F, q = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If C contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
p ∨ ¬a	Λ	= T



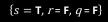


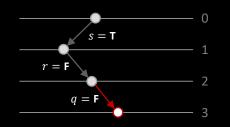
$$S = \{p\} \ M = \{s = T, r = F, q = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), (p, F)return DPLL $(C, S - t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return $\text{DPLL}(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	Λ	= T

$$\{s = T\}$$
 $\{s = T, r = F\}$





$$S = \{p\} \ M = \{s = T, r = F, q = F\}$$

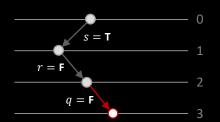
- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a $(t, \text{ polarity } v) = \text{pure symbol}(C), \quad (p, \mathbf{F})$ return $\text{DPLL}(C, S - t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL $(C, S u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S)$; $R = \mathbf{rest}(S)$;

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	Λ	= T

$$\{s = T\}$$

$$\{s = T, r = F\}$$

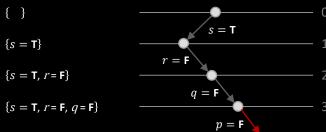
$$\{s = T, r = F, q = F\}$$



$$S = \{\} M = \{s = T, r = F, q = F, p = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

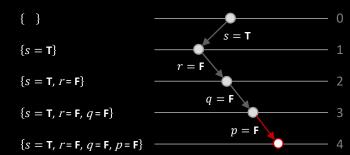
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \vee \neg q \vee r \vee s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \vee \neg q$	٨	= T



$$S = \{\} M = \{s = T, r = F, q = F, p = F\}$$

- If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL(C, S - t, $M \cup \{t = v\}$)
- 4. If there is a (u, polarity v) = unit clause(C), return DPLL($C, S - u, M \cup \{u = v\}$)
- $P = \mathbf{first}(S); R = \mathbf{rest}(S);$ 5.
- 6. Return $DPLL(C, R, M \cup \{P = T\})$ $DPLL(C, R, M \cup \{P = \mathbf{F}\})$

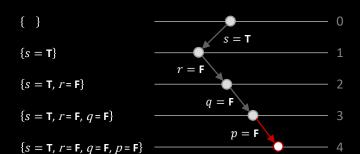
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	Λ	
$p \lor \neg q$	Λ	= T



$$S = \{\} M = \{s = T, r = F, q = F, p = F\}$$

- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

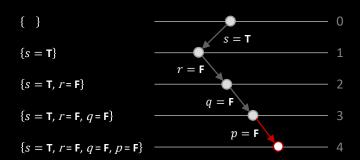
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	٨	= T
$p \vee \neg q$	٨	= T



- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- 2. If *C* contains an empty clause, return **F**
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

$$S = \{\} M = \{s = T, r = F, q = F, p = F\}$$

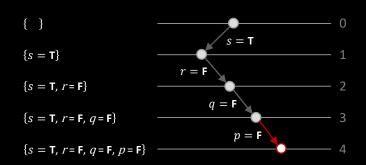
$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	٨	= T
$p \vee \neg q$	٨	= T



- 1. If (every $c \in C$ is **T**) \lor (C is empty), return **T**
- If C contains an empty clause, return F
- 3. If there is a (t, polarity v) = pure symbol(C), return DPLL $(C, S t, M \cup \{t = v\})$
- 4. If there is a (u, polarity v) = unit clause(C),return DPLL $(C, S - u, M \cup \{u = v\})$
- 5. $P = \mathbf{first}(S); R = \mathbf{rest}(S);$

$S = \{\} M = \{s = T, r = F, q = F, p = F\}$

$p \lor q \lor r \lor s$	٨	= T
$\neg p \lor q \lor \neg r$	٨	= T
$\neg q \lor \neg r \lor s$	٨	= T
$p \lor \neg q \lor r \lor s$	٨	= T
$q \vee \neg r$	٨	= T
$\neg p$	٨	= T
$p \vee \neg q$	٨	= T





Exercises from the textbook (chapter 7):

7.1, 7.4, 7.5, 7.7, 7.10



QUESTIONS?



ARTIFICIAL INTELLIGENCE COMP 131

FABRIZIO SANTINI