PROPOSITIONAL LOGIC 2

ARTIFICIAL INTELLIGENCE | COMP 131

TODAY ON AI

- Automated reasoning
- Efficient satisfiability
- Questions?

SECTION 01

Automated reasoning

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	$p, p \rightarrow q$	q
AND elimination	$p \wedge q$	p, q
Double negation	$\neg\neg p$	p
Unit resolution	$p \vee q, \neg q$	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.

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

INFERENCE

KNOWLEDGE BASE

1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	**police_car → policeman**
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry

NEW FACTS

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.

KNOWLEDGE BASE

1 $\text{person\_in\_front\_of\_car} \rightarrow \text{brake}$

2 $(\text{yellow\_light} \lor \text{policeman}) \land \neg\text{slippery}) \rightarrow \text{brake}$

3 **$\text{police\_car} \rightarrow \text{policeman}$**

4 $\text{snow} \rightarrow \text{slippery}$

5 $\text{slippery} \rightarrow \neg\text{dry}$

6 $\text{red\_light} \rightarrow \text{brake}$

7 $\text{winter} \rightarrow \text{snow}$

FACTS

yellow_light	¬red_light	¬snow
$\boxed{\text{police\_car}}$	¬person_in_front_of_car	dry

NEW FACTS

policeman

QUERY

Do we need to $Brake$?

INFERENCE

KNOWN	**police_car**
MP R3	$\text{police\_car} \rightarrow \text{policeman}$
	policeman

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.

KNOWLEDGE BASE

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

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry

NEW FACTS

policeman	¬slippery

QUERY

Do we need to $Brake$?

INFERENCE

KNOWN	**police_car**
MP R3	police_car → policeman
	policeman
KNOWN	**dry**
MT R5	slippery → ¬dry
DN	¬¬dry → ¬slippery
MP	dry → ¬slippery
	¬slippery

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.

KNOWLEDGE BASE

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

$\boxed{yellow\_light}$	$\neg red\_light$	$\neg snow$
$police\_car$	$\neg person\_in\_front\_of\_car$	dry

NEW FACTS

$\boxed{policeman}$ $\boxed{\neg slippery}$

QUERY

Do we need to $Brake$?

INFERENCE

KNOWN	$police\_car$
MP R3	$police\_car \rightarrow policeman$
	$policeman$
KNOWN	dry
MT R5	$slippery \rightarrow \neg dry$
DN	$\neg\neg dry \rightarrow \neg slippery$
MP	$dry \rightarrow \neg slippery$
	$\neg slippery$
KNOWN	$yellow\_light$
KNOWN	$policeman$
KNOWN	$\neg slippery$
MP R2	$((yellow\_light \lor policeman) \land \neg slippery) \rightarrow brake$
	$brake$

CONCLUSION $\boxed{brake}$

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	$person\_in\_front\_of\_car \rightarrow brake$
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

$yellow\_light$	$\neg red\_light$	$\neg snow$
$police\_car$	$\neg person\_in\_front\_of\_car$	dry

brake

KNOWLEDGE BASE

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

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry

brake

R1

person_in_front_of_car → brake

KNOWLEDGE BASE

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

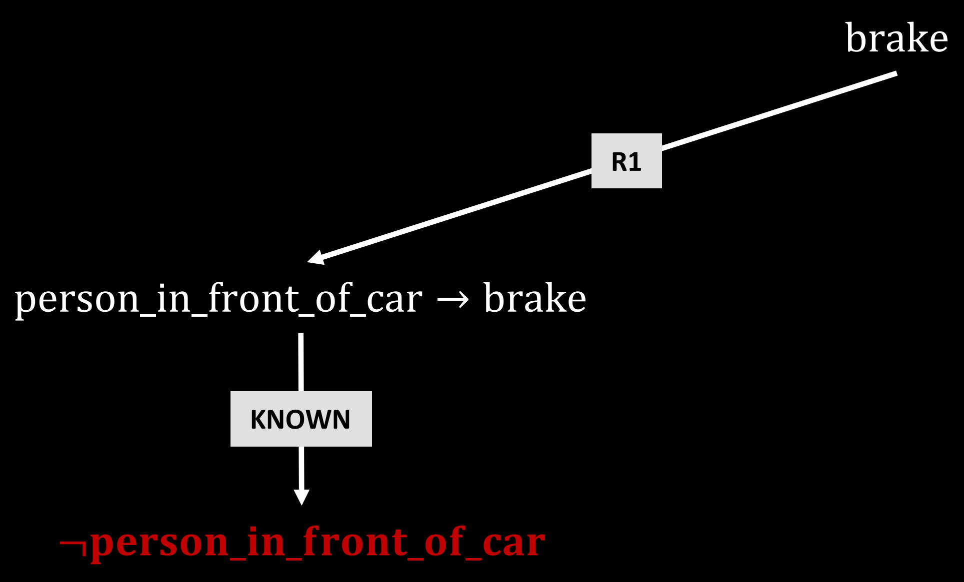
QUERY

Do we need to *Brake*?

FACTS

yellow_light ¬red_light ¬snow

police_car ¬person_in_front_of_car dry



KNOWLEDGE BASE

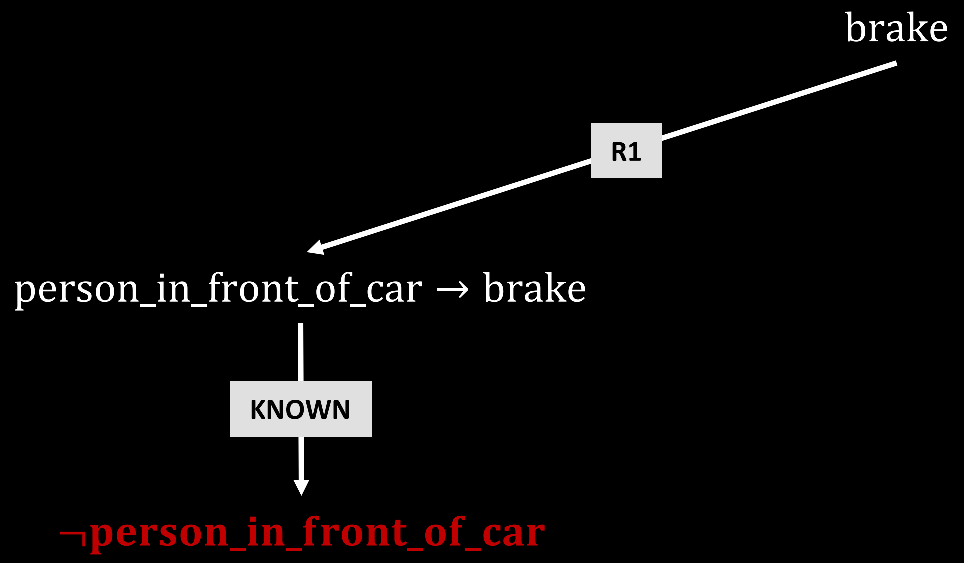
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	$\boxed{\text{¬person\_in\_front\_of\_car}}$	dry



KNOWLEDGE BASE

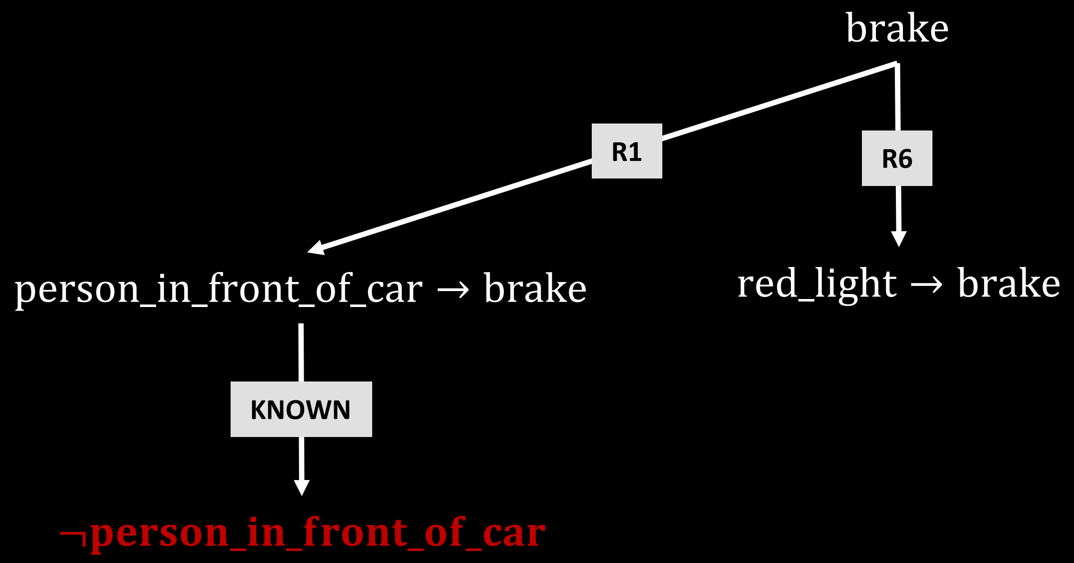
1 ~~person_in_front_of_car → brake~~
2 (yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

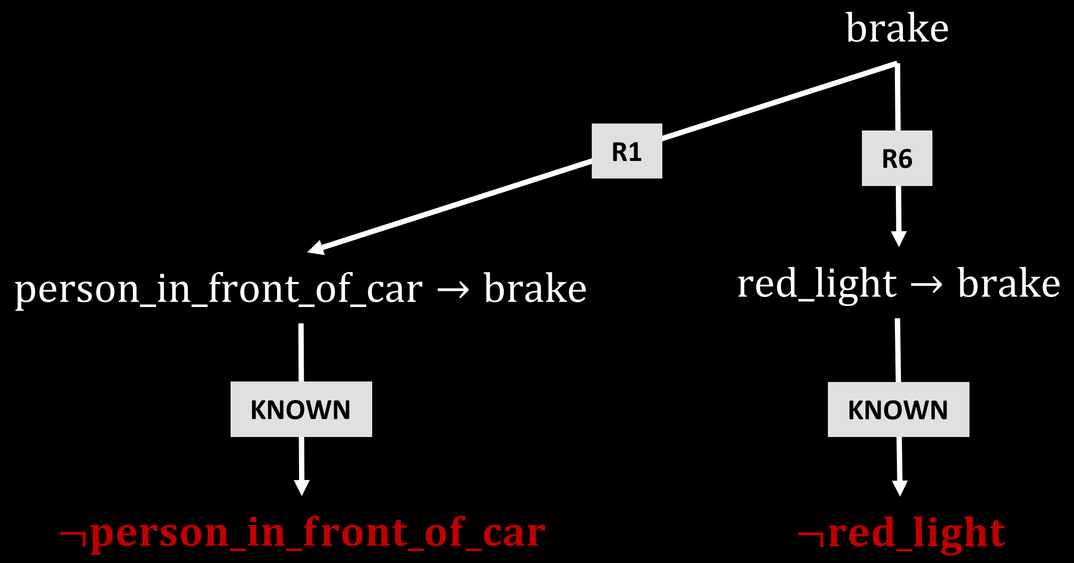
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

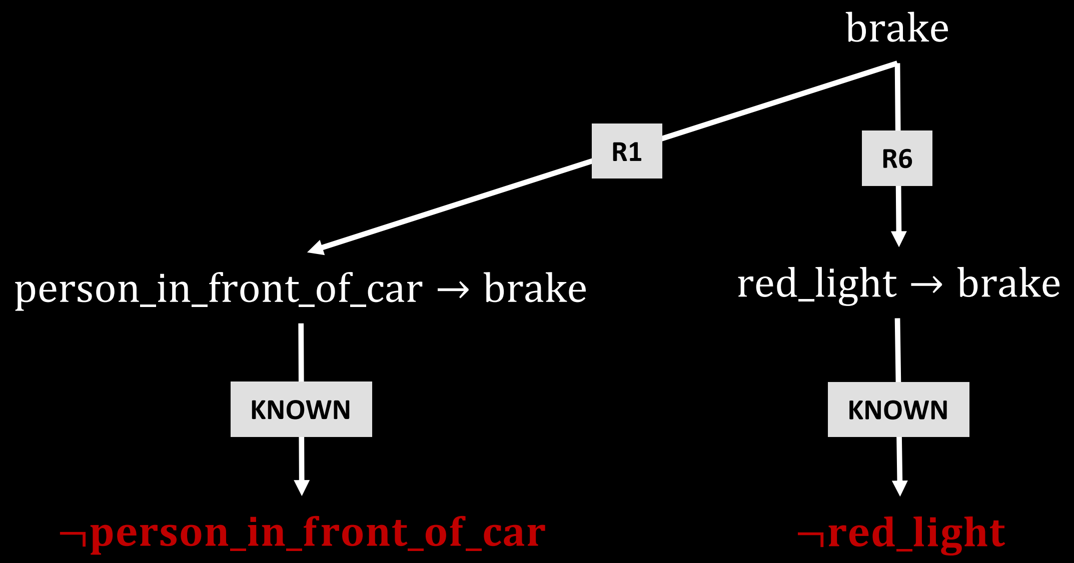
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	**red_light → brake**
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

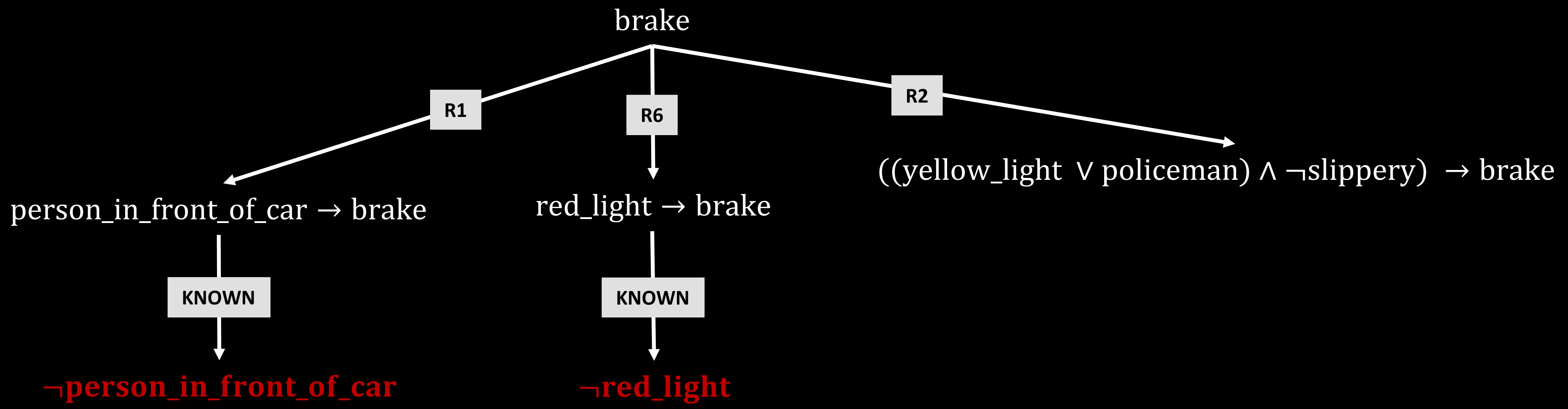
1 ~~person_in_front_of_car → brake~~
2 (yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 ~~red_light → brake~~
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

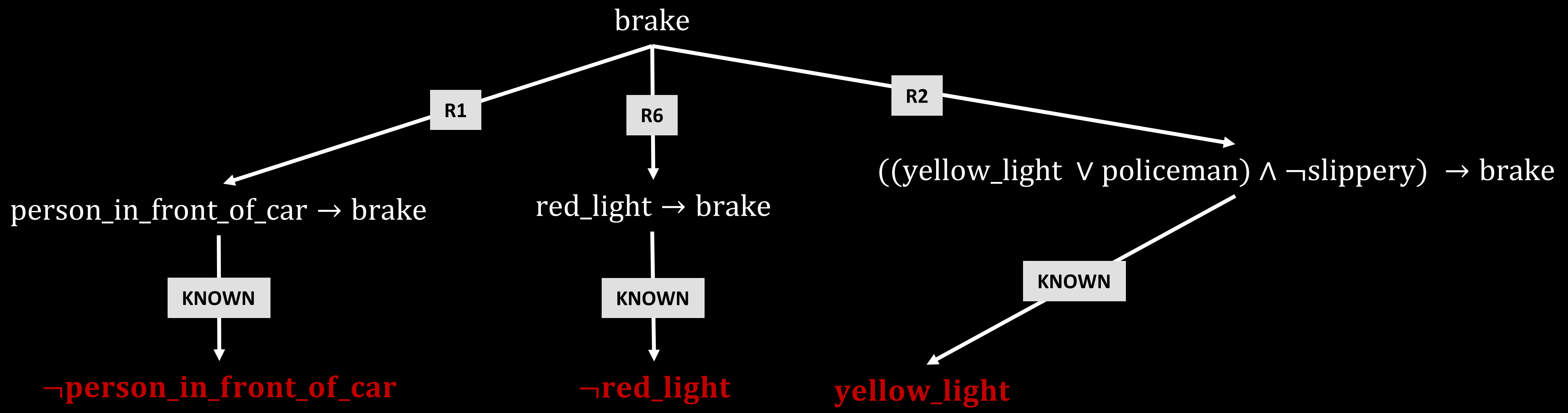
1. $\text{person\_in\_front\_of\_car} \rightarrow \text{brake}$
2. $(\text{yellow\_light} \lor \text{policeman}) \land \neg\text{slippery}) \rightarrow \text{brake}$
3. $\text{police\_car} \rightarrow \text{policeman}$
4. $\text{snow} \rightarrow \text{slippery}$
5. $\text{slippery} \rightarrow \neg\text{dry}$
6. $\text{red\_light} \rightarrow \text{brake}$
7. $\text{winter} \rightarrow \text{snow}$

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

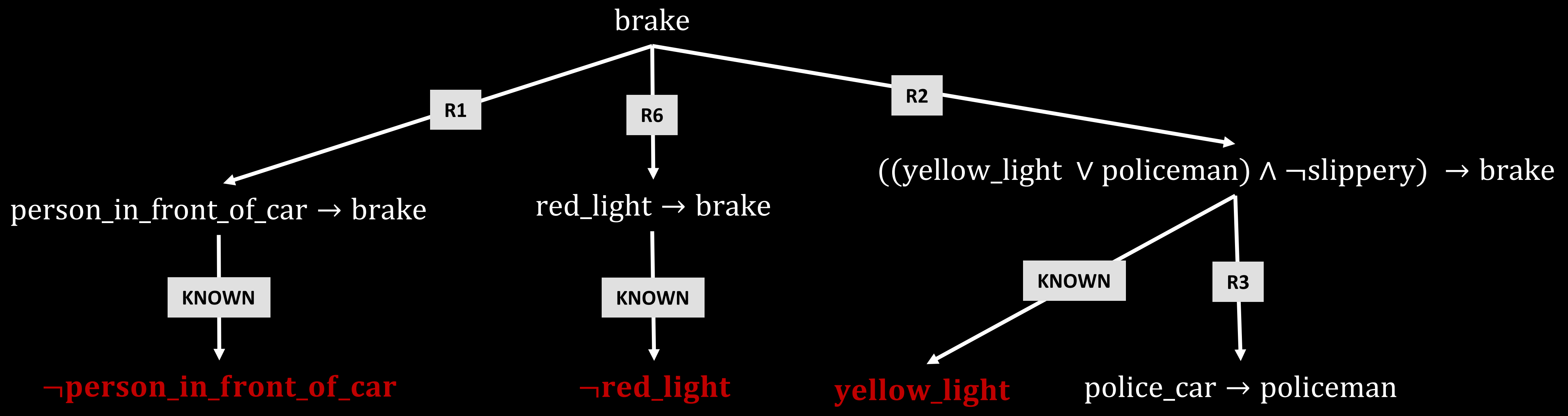
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

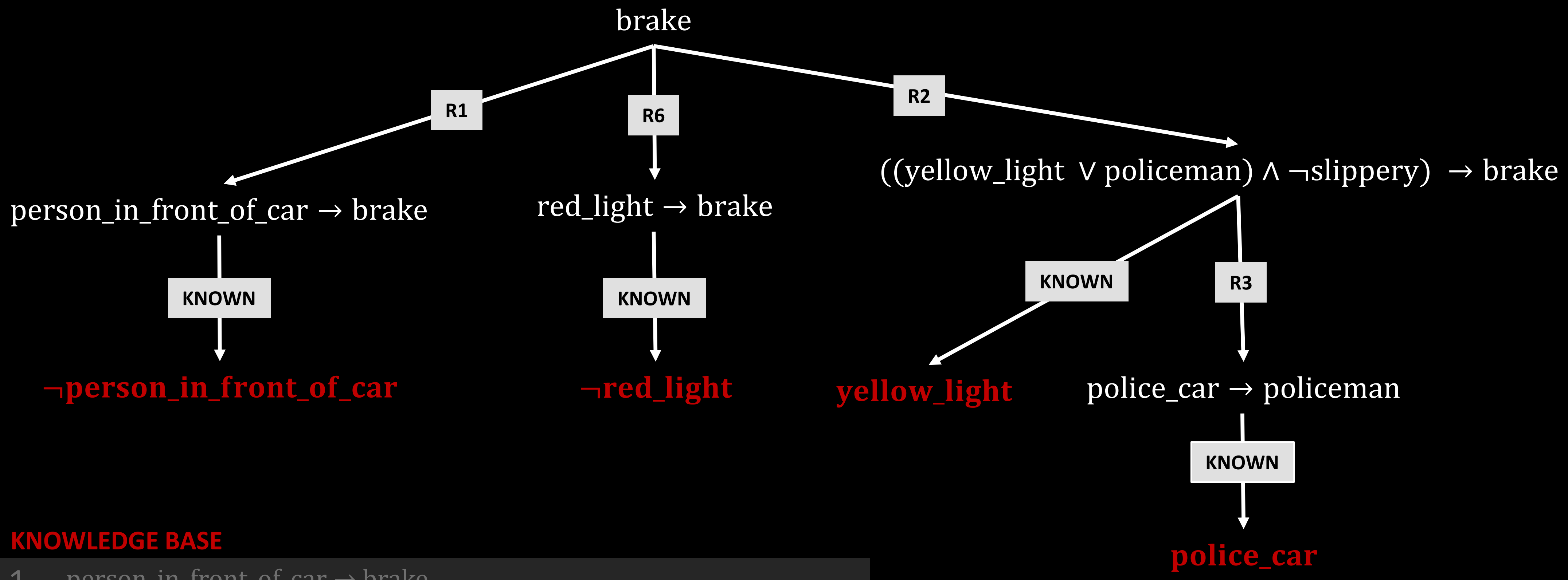
1 person_in_front_of_car → brake
2 (yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light ¬red_light ¬snow
police_car ¬person_in_front_of_car dry



KNOWLEDGE BASE

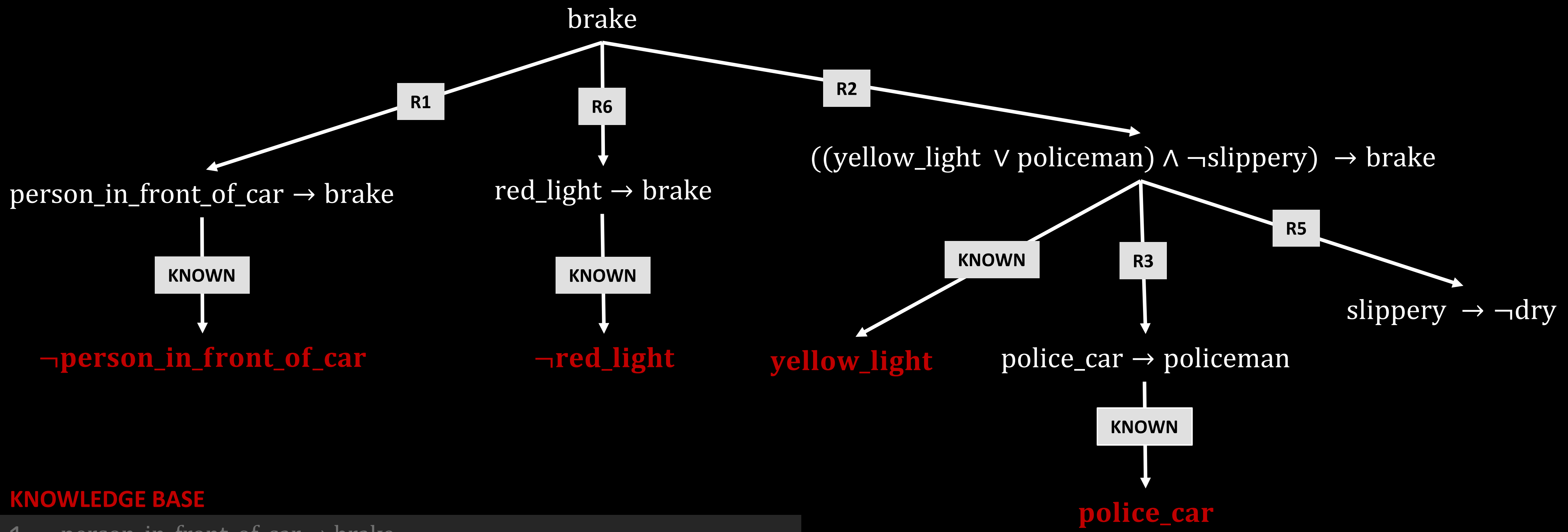
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	**police_car → policeman**
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

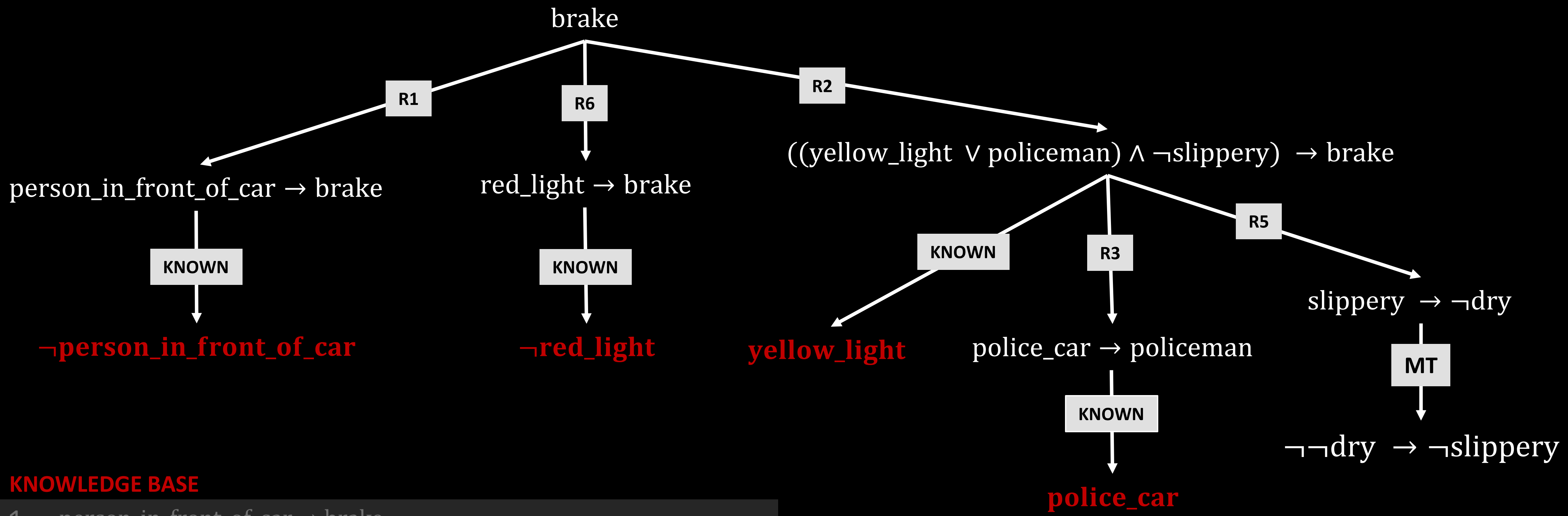
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	**police_car → policeman**
4	snow → slippery
5	**slippery → ¬dry**
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

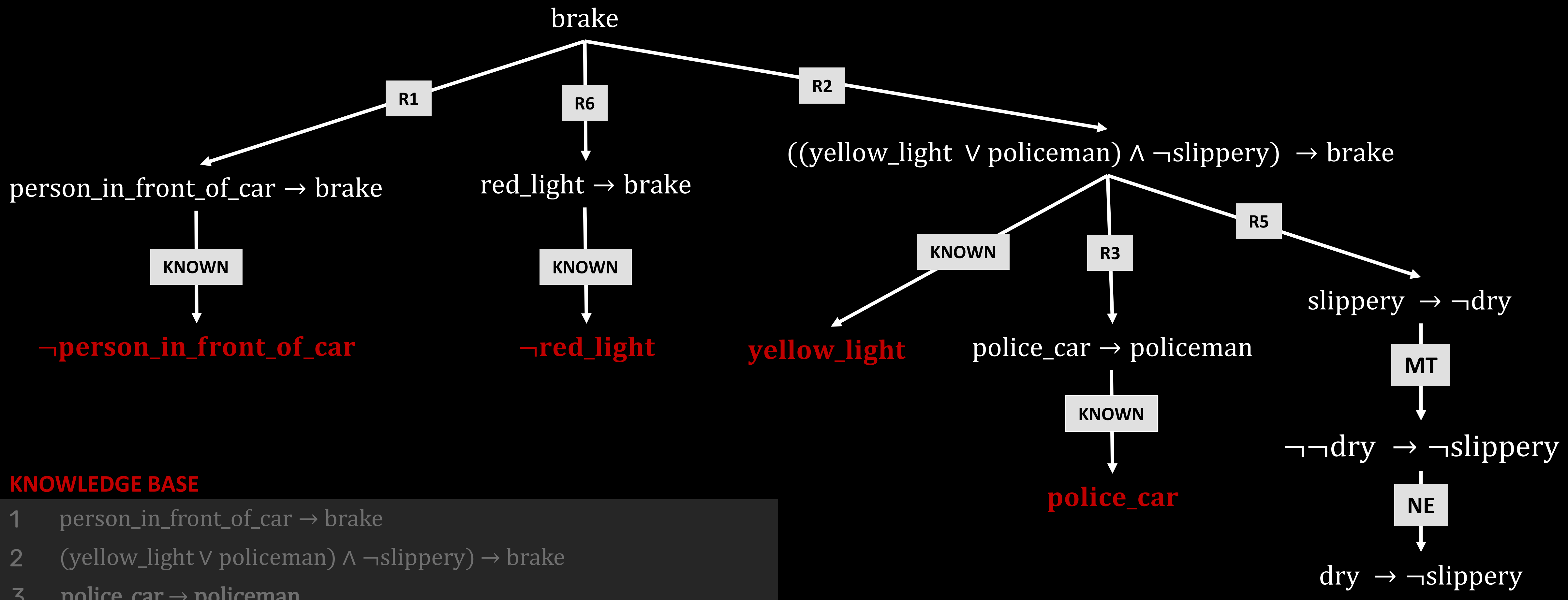
1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

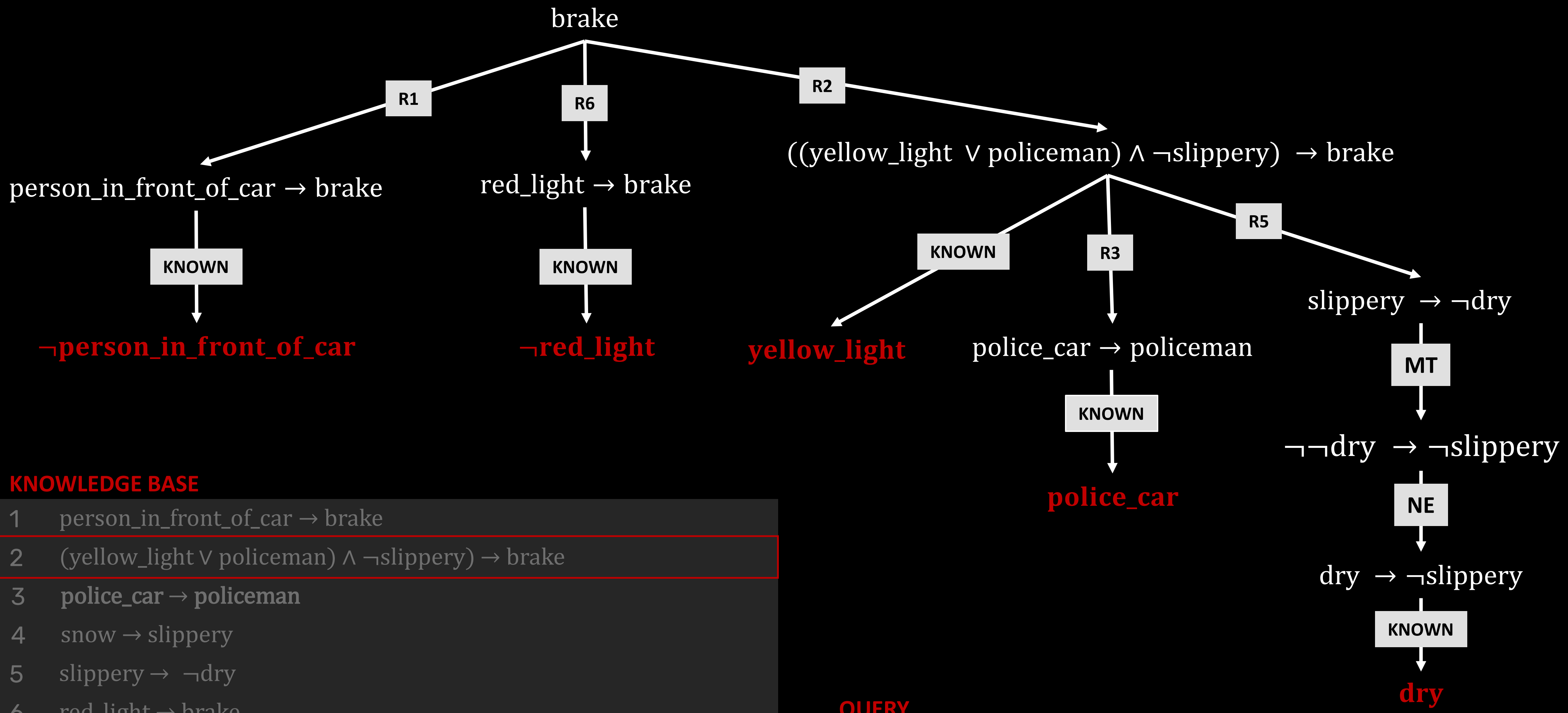
1 person_in_front_of_car → brake
2 (yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry



KNOWLEDGE BASE

1	person_in_front_of_car → brake
2	(yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	**police_car → policeman**
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry

Automated reasoning

BACKWARD CHAINING

brake

R1: person_in_front_of_car → brake — KNOWN — ¬person_in_front_of_car

R6: red_light → brake — KNOWN — ¬red_light

R2: ((yellow_light ∨ policeman) ∧ ¬slippery) → brake

KNOWN — yellow_light

R3: police_car → policeman — KNOWN — police_car

R5: slippery → ¬dry

MT

¬¬dry → ¬slippery

NE

dry → ¬slippery

KNOWN

dry

KNOWLEDGE BASE

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

CONCLUSION **brake**

QUERY

Do we need to *Brake*?

FACTS

yellow_light	¬red_light	¬snow
police_car	¬person_in_front_of_car	dry

SECTION 02

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**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   ∨
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   ∨
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   ∨
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \lor q \lor r \lor s$ ∧

$\neg p \lor q \lor \neg r$ ∧

$\neg q \lor \neg r \lor s$ ∧

$p \lor \neg q \lor r \lor s$ ∧

$q \lor \neg r \lor \neg s$ ∧

$\neg p \lor \neg s$ ∧

$p \lor \neg q$ ∧

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   ∨
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

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

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

CLAUSES

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

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

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

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

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

$\neg p \vee \neg s \quad \wedge$

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

{ }

0

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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

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

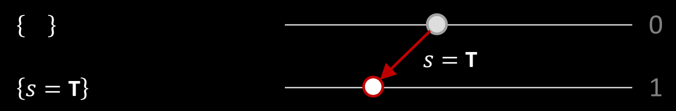
$\neg q \vee \neg r \vee s \quad \wedge$

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

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

$\neg p \vee \neg s \quad \wedge$

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   ∨
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \vee q \vee r \vee \mathbf{s}$ ∧

$\neg p \vee q \vee \neg r$ ∧

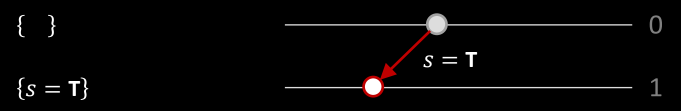
$\neg q \vee \neg r \vee \mathbf{s}$ ∧

$p \vee \neg q \vee r \vee \mathbf{s}$ ∧

$q \vee \neg r \vee \neg \mathbf{s}$ ∧

$\neg p \vee \neg \mathbf{s}$ ∧

$p \vee \neg q$ ∧



DPLL(C, S, M):

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

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

CLAUSES

$p \vee q \vee r \vee s \quad \wedge \quad = \mathrm{T}$

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

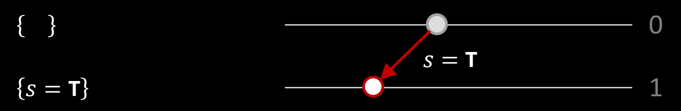
$\neg q \vee \neg r \vee s \quad \wedge \quad = \mathrm{T}$

$p \vee \neg q \vee r \vee s \quad \wedge \quad = \mathrm{T}$

$q \vee \neg r \vee \boldsymbol{\neg s} \quad \wedge$

$\neg p \vee \boldsymbol{\neg s} \quad \wedge$

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



Automated reasoning

DAVIS-PUTMAN-LOGEMANN-LOVELAND ALGORITHM

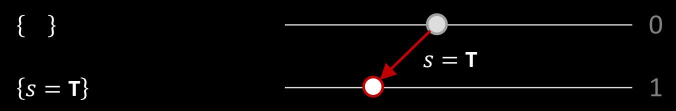
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   ∨
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \vee q \vee r \vee s \quad \wedge$ = T

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

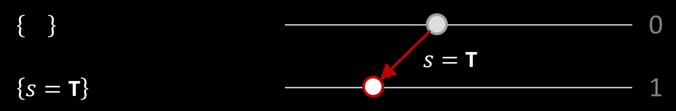
$\neg q \vee \neg r \vee s \quad \wedge$ = T

$p \vee \neg q \vee r \vee s \quad \wedge$ = T

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

$\neg p \quad \wedge$

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



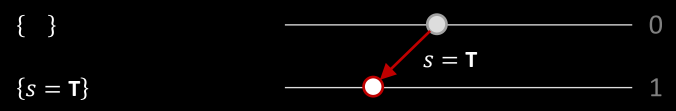
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



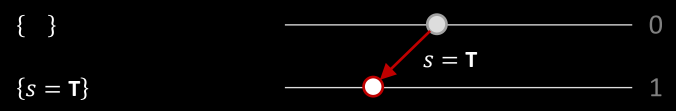
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



DPLL(C, S, M):

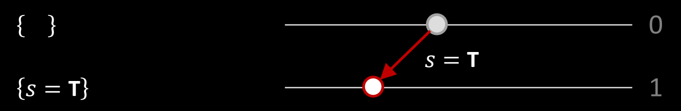
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(r, \mathbf{F})$

CLAUSES

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



42

Automated reasoning

DAVIS-PUTMAN-LOGEMANN-LOVELAND ALGORITHM

DPLL(C, S, M):

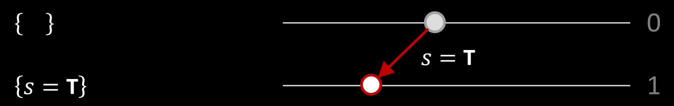
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(r, \mathbf{F})$

CLAUSES

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \vee q \vee r \vee s \quad \wedge$ = T

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

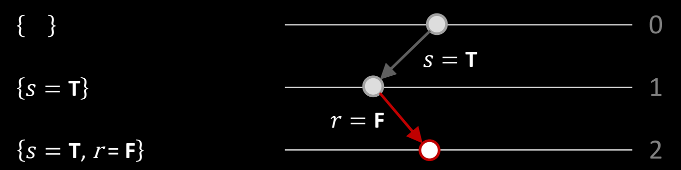
$\neg q \vee \neg r \vee s \quad \wedge$ = T

$p \vee \neg q \vee r \vee s \quad \wedge$ = T

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

$\neg p \quad \wedge$

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   ∨
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \lor q \lor r \lor s \quad \land \quad = \text{T}$

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

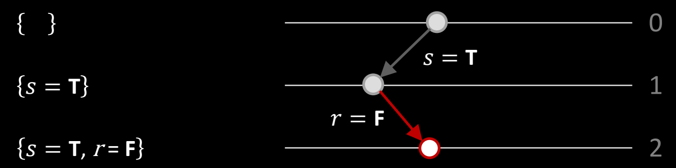
$\neg q \lor \neg r \lor s \quad \land \quad = \text{T}$

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

$q \lor \neg r \quad \land$

$\neg p \quad \land$

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



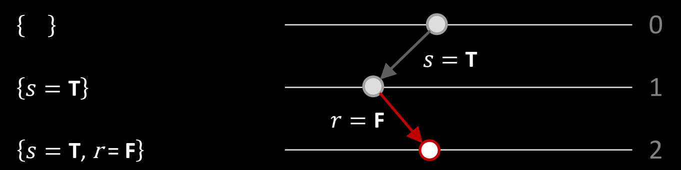
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



46

DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \textbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \textbf{F}\}$)

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

CLAUSES

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

$\neg p \vee q \vee \neg r$ $\wedge$ = T

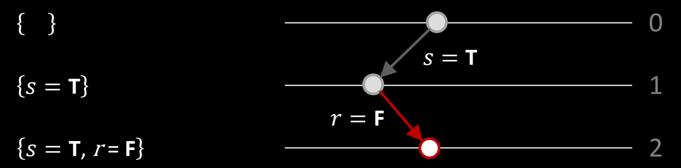
$\neg q \vee \neg r \vee s$ $\wedge$ = T

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

$q \vee \neg r$ $\wedge$ = T

$\neg p$ $\wedge$

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

$p \vee q \vee r \vee s \quad \wedge \quad = \mathrm{T}$

$\neg p \vee q \vee \neg r \quad \wedge \quad = \mathrm{T}$

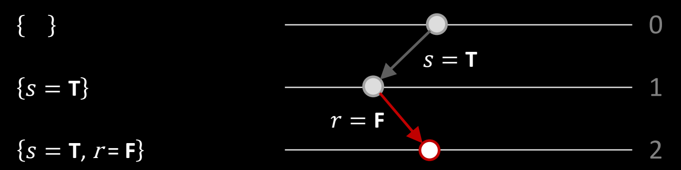
$\neg q \vee \neg r \vee s \quad \wedge \quad = \mathrm{T}$

$p \vee \neg q \vee r \vee s \quad \wedge \quad = \mathrm{T}$

$q \vee \neg r \quad \wedge \quad = \mathrm{T}$

$\neg p \quad \wedge$

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



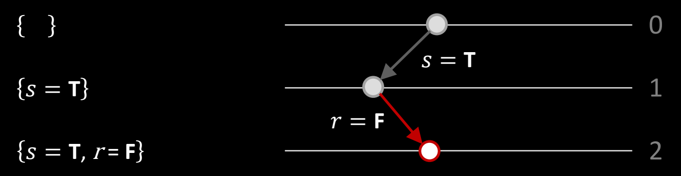
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



DPLL(C, S, M):

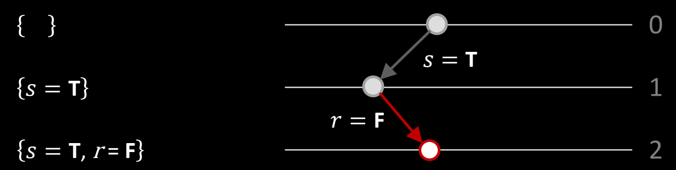
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(q, \mathbf{F})$

CLAUSES

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



DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

(q, **F**)

CLAUSES

$p \vee q \vee r \vee s \quad \wedge \quad$ = T

$\neg p \vee q \vee \neg r \quad \wedge \quad$ = T

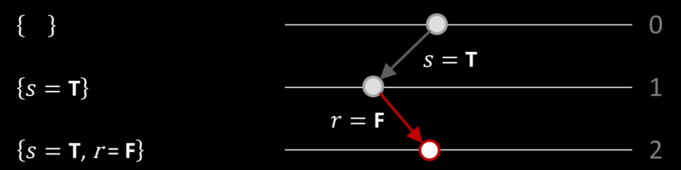
$\neg q \vee \neg r \vee s \quad \wedge \quad$ = T

$p \vee \neg q \vee r \vee s \quad \wedge \quad$ = T

$q \vee \neg r \quad \wedge \quad$ = T

$\neg p \quad \wedge$

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



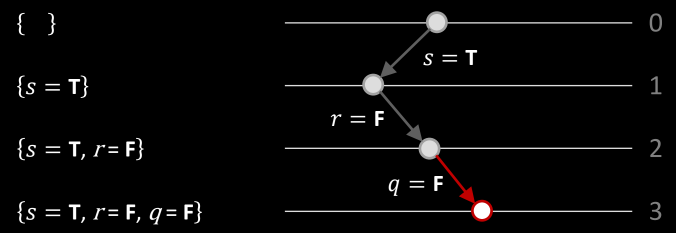
DPLL(C, S, M):

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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\lor$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



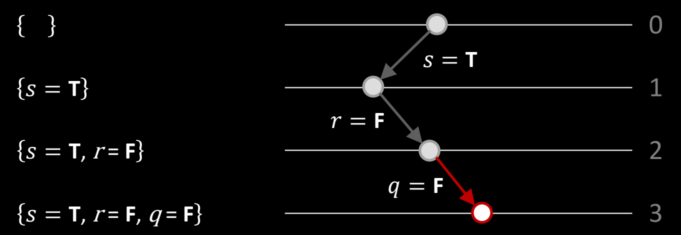
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



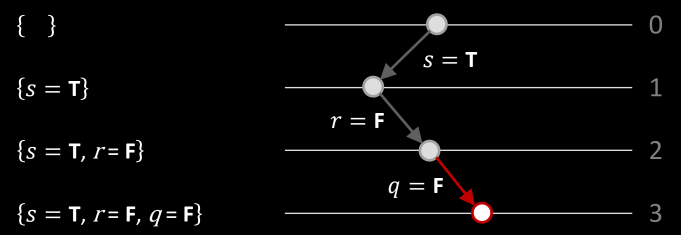
DPLL(C, S, M):

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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\lor$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



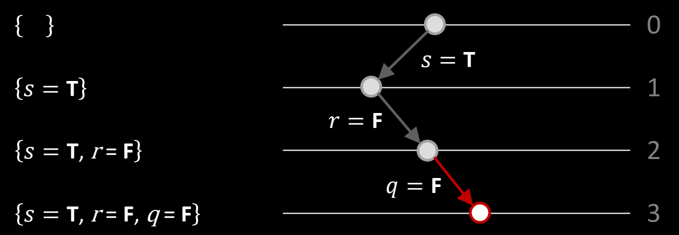
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



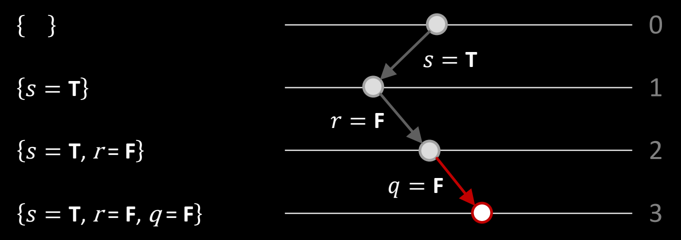
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

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



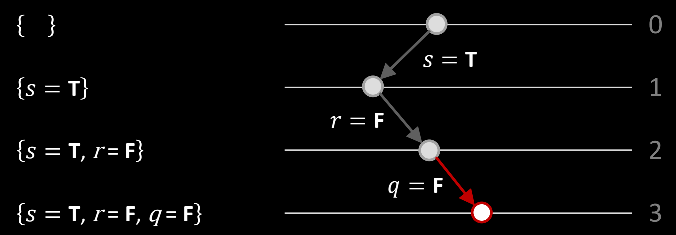
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



DPLL(C, S, M):

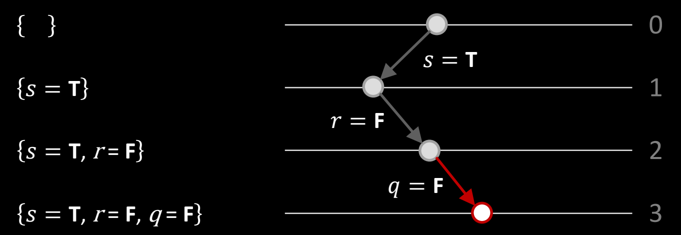
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(p, \mathbf{F})$

CLAUSES

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



DPLL(C, S, M):

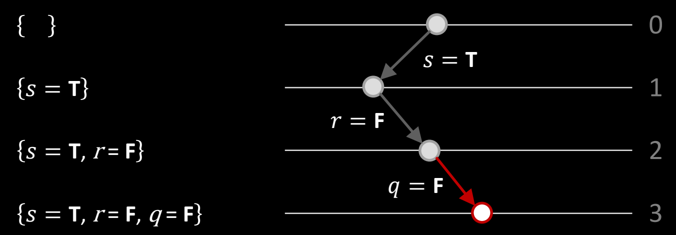
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL$(C, S - t, M \cup \{t = v\})$
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL$(C, S - u,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

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

$(p, \mathbf{F})$

CLAUSES

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



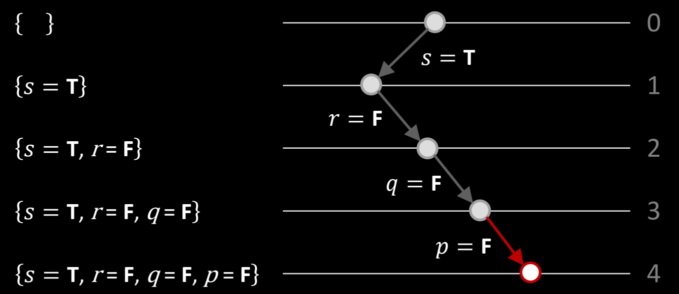
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   ∨
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

$p \lor q \lor r \lor s$	∧	= T
$\neg p \lor q \lor \neg r$	∧	= T
$\neg q \lor \neg r \lor s$	∧	= T
$p \lor \neg q \lor r \lor s$	∧	= T
$q \lor \neg r$	∧	= T
$\neg p$	∧	
$p \lor \neg q$	∧	= T



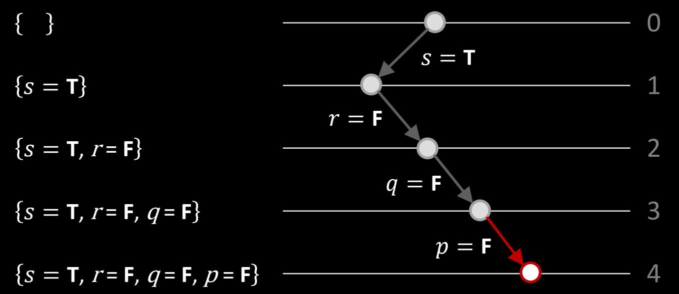
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



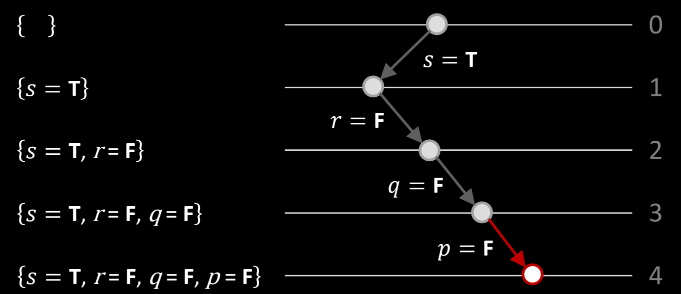
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

CLAUSES

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



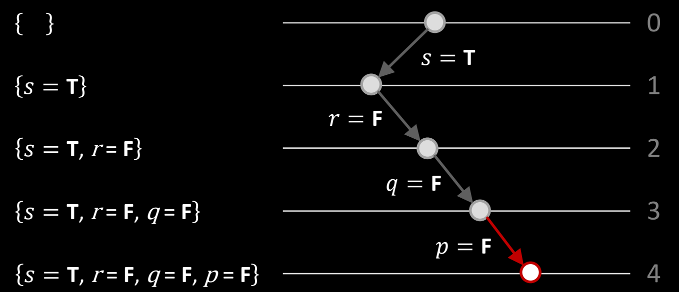
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



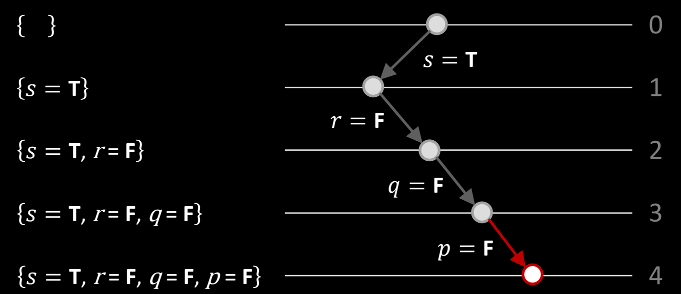
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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);
6. Return
   DPLL($C, R, M \cup \{P = \mathbf{T}\}$)
   ∨
   DPLL(C, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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



PRACTICE

Exercises from the textbook (chapter 7):

7.1, 7.4, 7.5, 7.7, 7.10

READINGS

Chapters 7

QUESTIONS ?

ARTIFICIAL INTELLIGENCE

COMP 131

FABRIZIO SANTINI

VERSION 4.1