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.

KNOWLEDGE BASE

1	$\text{person\_in\_front\_of\_car} \rightarrow \text{brake}$
2	$(\text{yellow\_light} \vee \text{policeman}) \wedge \neg\text{slippery}) \rightarrow \text{brake}$
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 $Brake$?

INFERENCE

FACTS

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

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.

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*?

INFERENCE

KNOWN	**police_car**
MP R3	police_car → policeman
	policeman

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.

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 ∧ **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 → ¬dry
DN	¬¬dry → ¬slippery
MP	dry → ¬slippery
	¬slippery

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 ∧ 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 → ¬dry
DN	¬¬dry → ¬slippery
MP	dry → ¬slippery
	¬slippery
KNOWN	**yellow_light**
KNOWN	**policeman**
KNOWN	**¬slippery**
MP R2	((yellow_light ∨ policeman) ∧ ¬slippery) → 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

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 → ¬dry
DN	¬¬dry → ¬slippery
MP	dry → ¬slippery
	¬slippery
KNOWN	**yellow_light**
KNOWN	**policeman**
KNOWN	**¬slippery**
MP R2	((yellow_light ∨ policeman) ∧ ¬slippery) → brake

CONCLUSION $\boxed{\textbf{brake}}$

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	$\text{person\_in\_front\_of\_car} \rightarrow \text{brake}$
2	$((\text{yellow\_light} \vee \text{policeman}) \wedge \neg\text{slippery}) \rightarrow \text{brake}$
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

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

brake

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

brake

KNOWLEDGE BASE

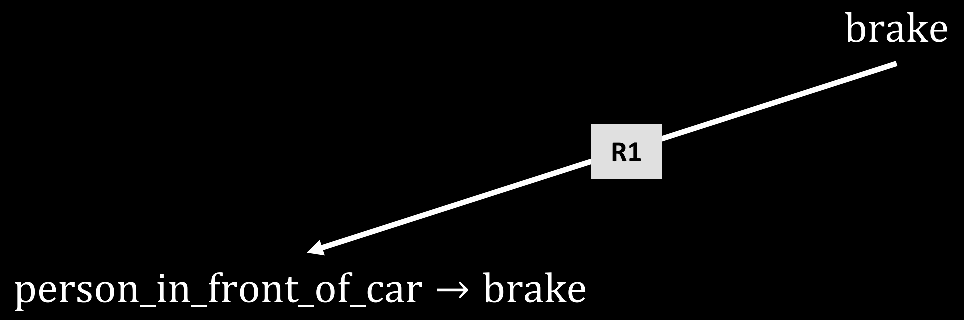
1 person_in_front_of_car → brake
2 ((yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



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 ∧ dry ∧
police_car ∧ ¬person_in_front_of_car

brake

R1

person_in_front_of_car → brake

KNOWN

¬person_in_front_of_car

KNOWLEDGE BASE

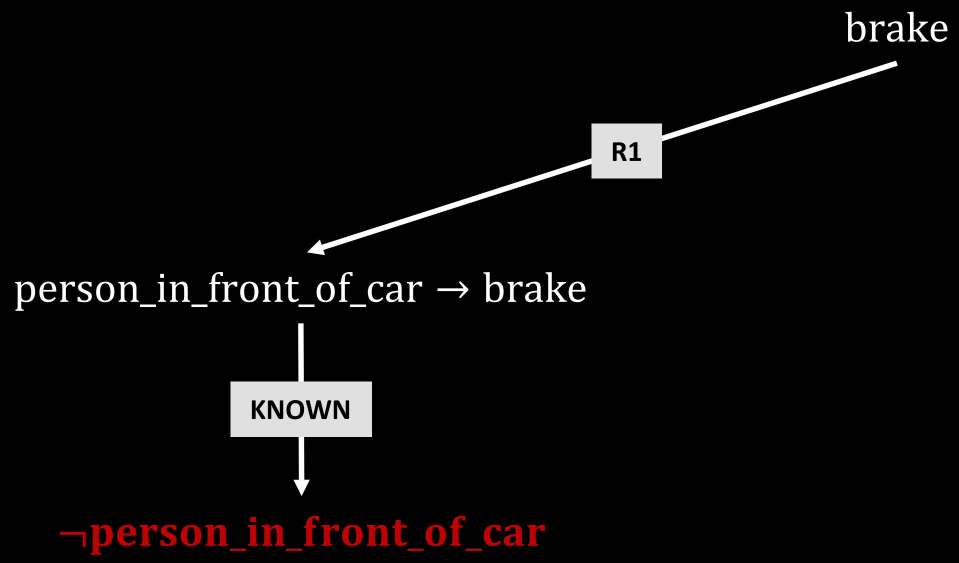
1	person_in_front_of_car → brake
2	((yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

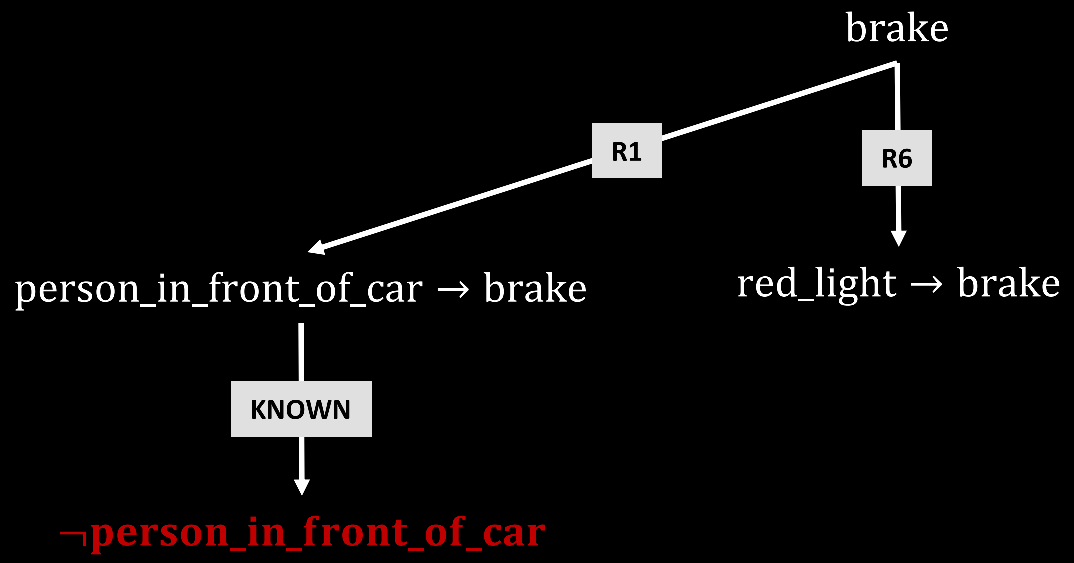
1. ~~person_in_front_of_car → brake~~
2. **((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3. police_car → policeman
4. snow → slippery
5. slippery → ¬dry
6. **red_light → brake**
7. winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

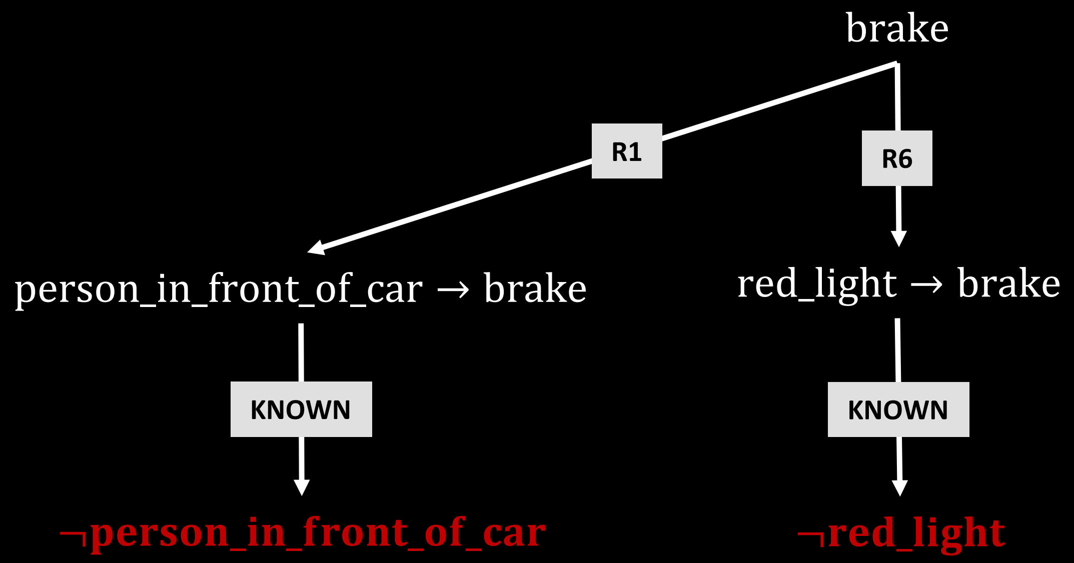
	1	person_in_front_of_car → brake
➡	2	**((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
	3	police_car → policeman
	4	snow → slippery
	5	slippery → ¬dry
➡	6	red_light → brake
	7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

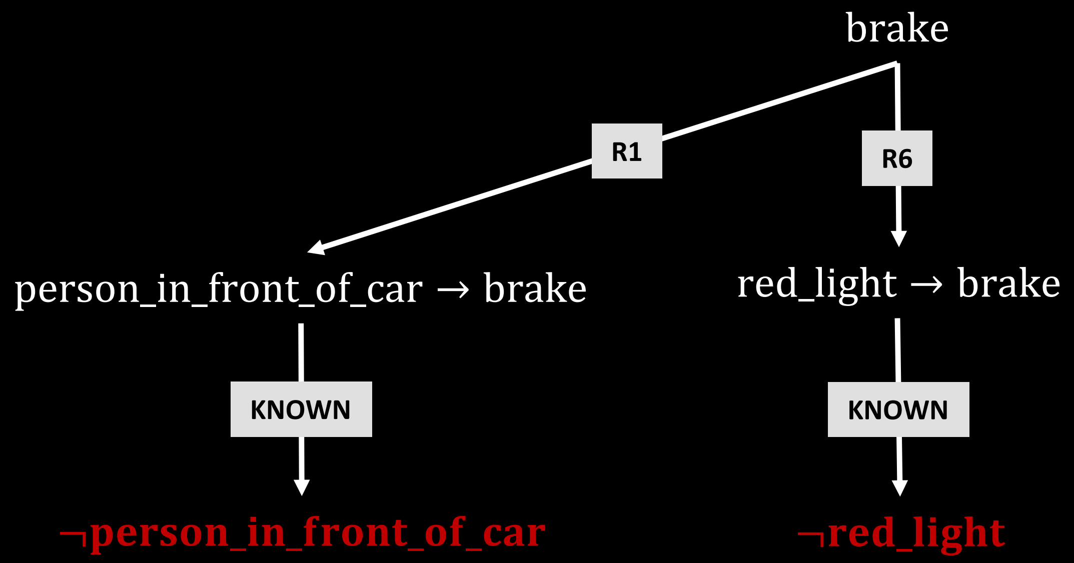
1	person_in_front_of_car → brake
2	**((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	**red_light → brake**
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

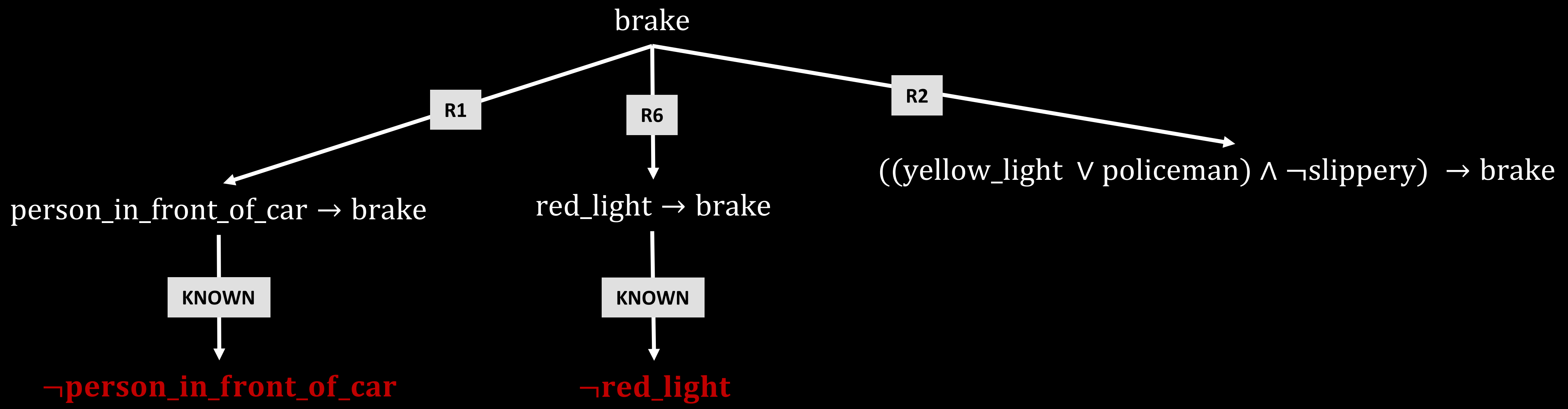
1 person_in_front_of_car → brake
2 **((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
~~6 red_light → brake~~
7 winter → snow

QUERY

Do we need to $Brake$?

FACTS

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



KNOWLEDGE BASE

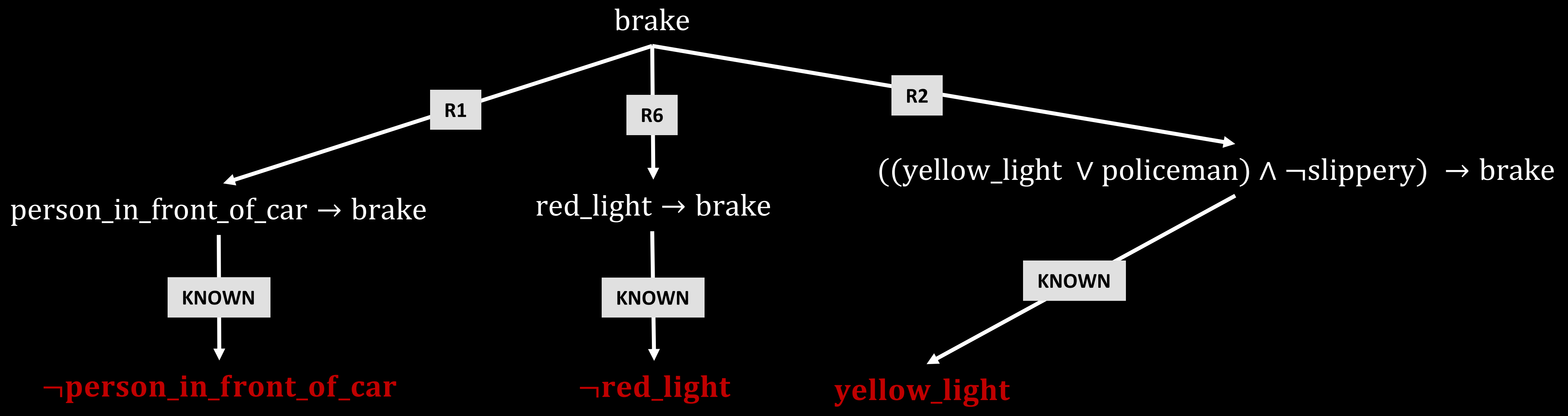
1. person_in_front_of_car → brake
2. **((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3. police_car → policeman
4. snow → slippery
5. slippery → ¬dry
6. red_light → brake
7. winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

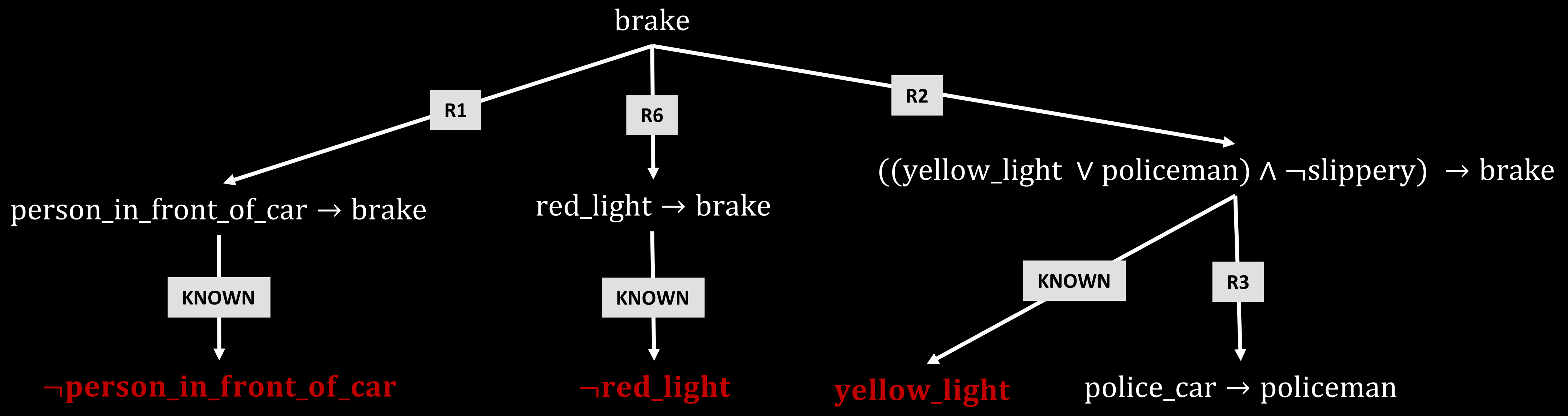
1	person_in_front_of_car → brake
2	**((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

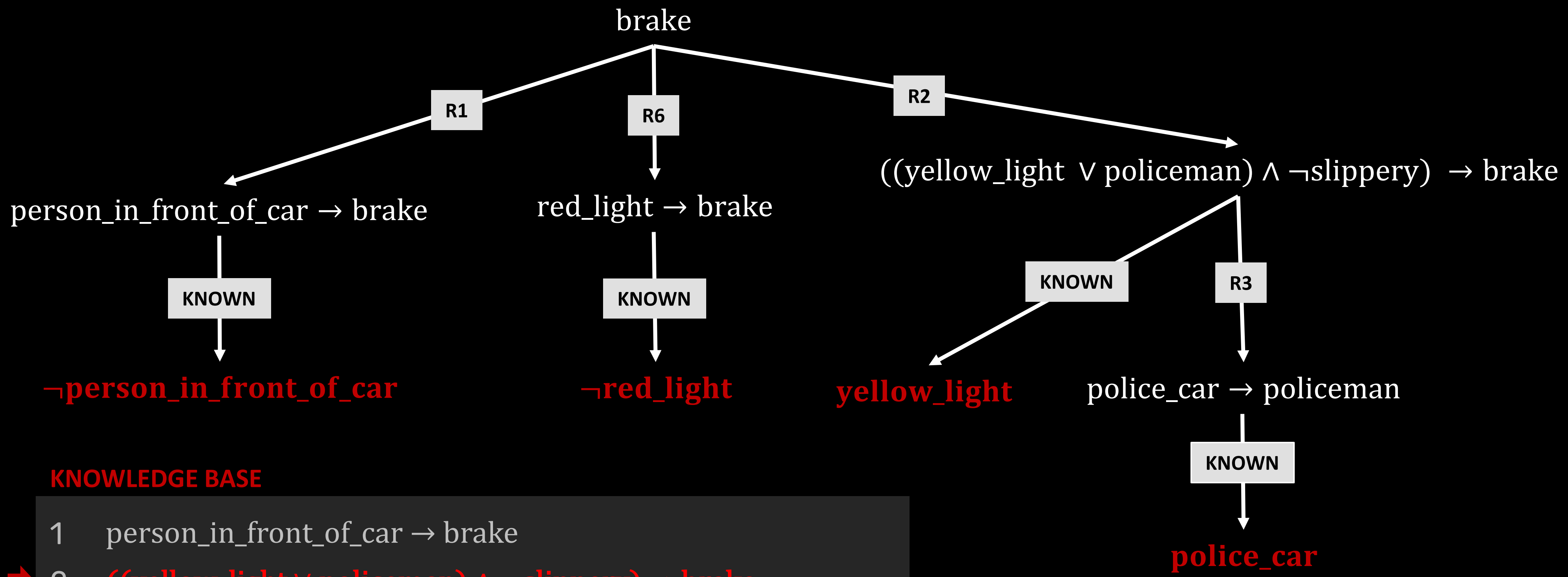
1 person_in_front_of_car → brake
2 **((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3 **police_car → policeman**
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

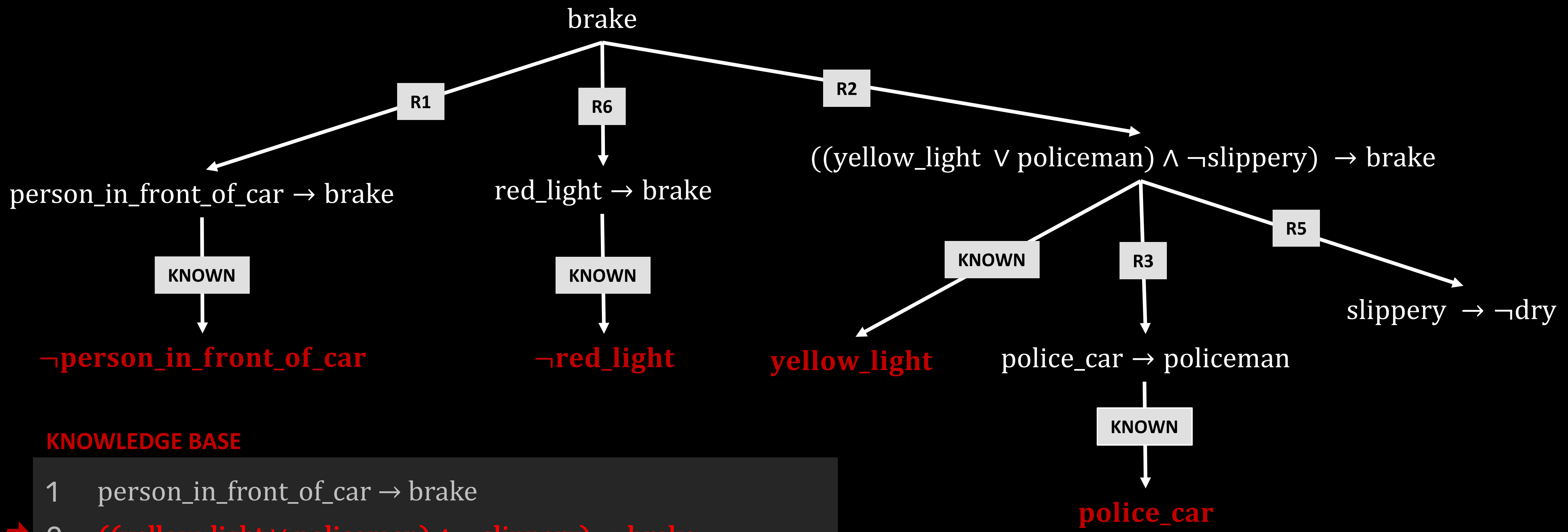
1	person_in_front_of_car → brake
2	((yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



KNOWLEDGE BASE

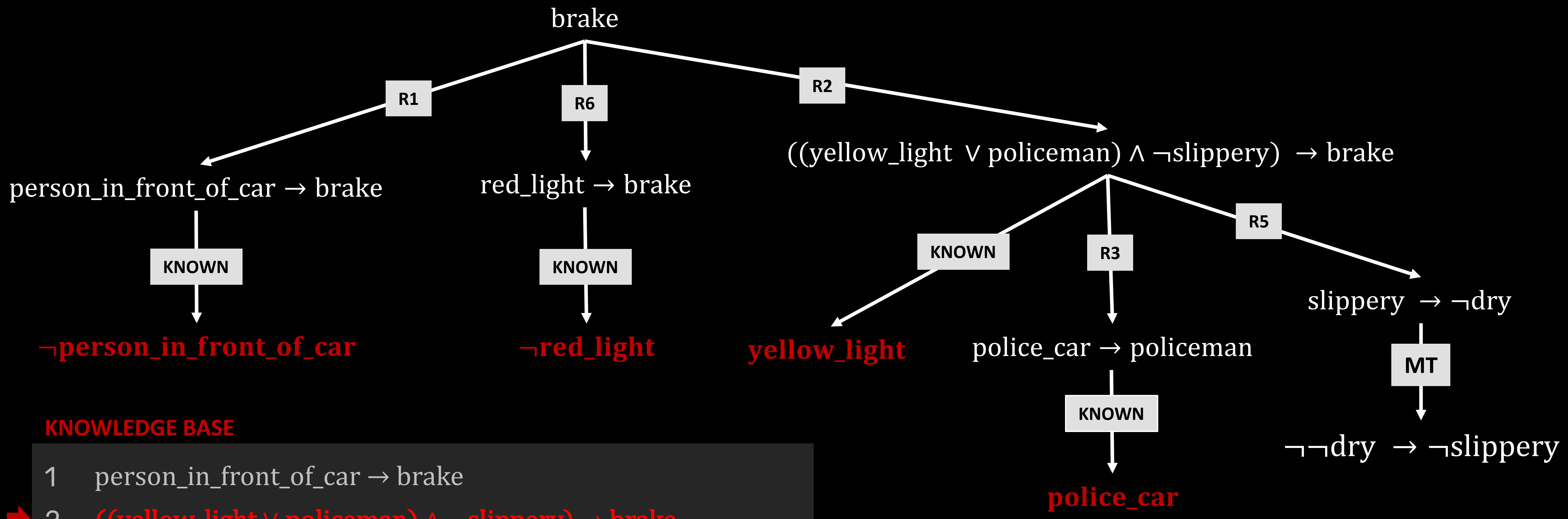
1 person_in_front_of_car → brake
2 ((yellow_light ∨ policeman) ∧ ¬slippery) → brake
3 police_car → policeman
4 snow → slippery
5 slippery → ¬dry
6 red_light → brake
7 winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



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 ∧ dry ∧ police_car ∧ ¬person_in_front_of_car

brake

R1 · R6 · R2

person_in_front_of_car → brake

KNOWN

¬person_in_front_of_car

red_light → brake

KNOWN

¬red_light

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

KNOWN · R3 · R5

yellow_light

police_car → policeman

KNOWN

police_car

slippery → ¬dry

MT

¬¬dry → ¬slippery

NE

dry → ¬slippery

KNOWLEDGE BASE

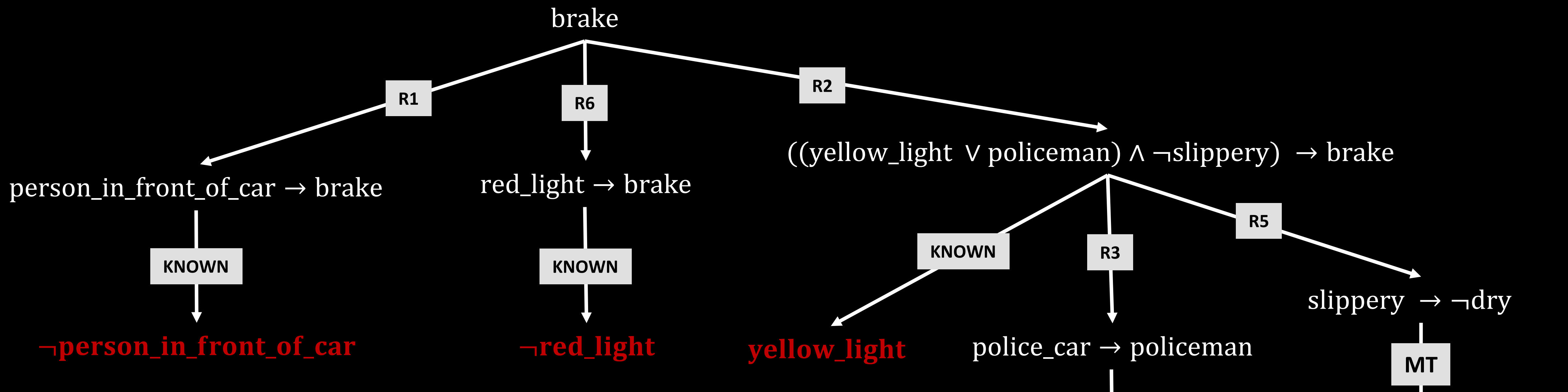
1	person_in_front_of_car → brake
2	((yellow_light ∨ policeman) ∧ ¬slippery) → brake
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

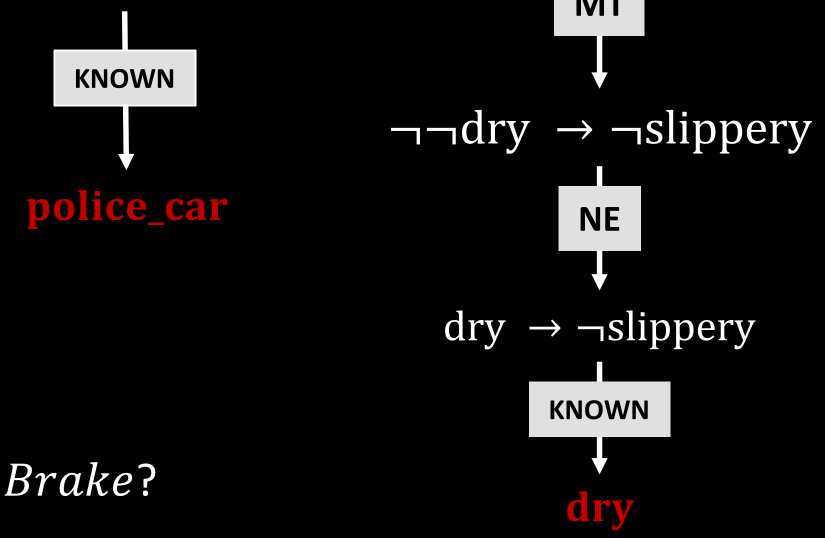
QUERY

Do we need to *Brake*?

FACTS

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





KNOWLEDGE BASE

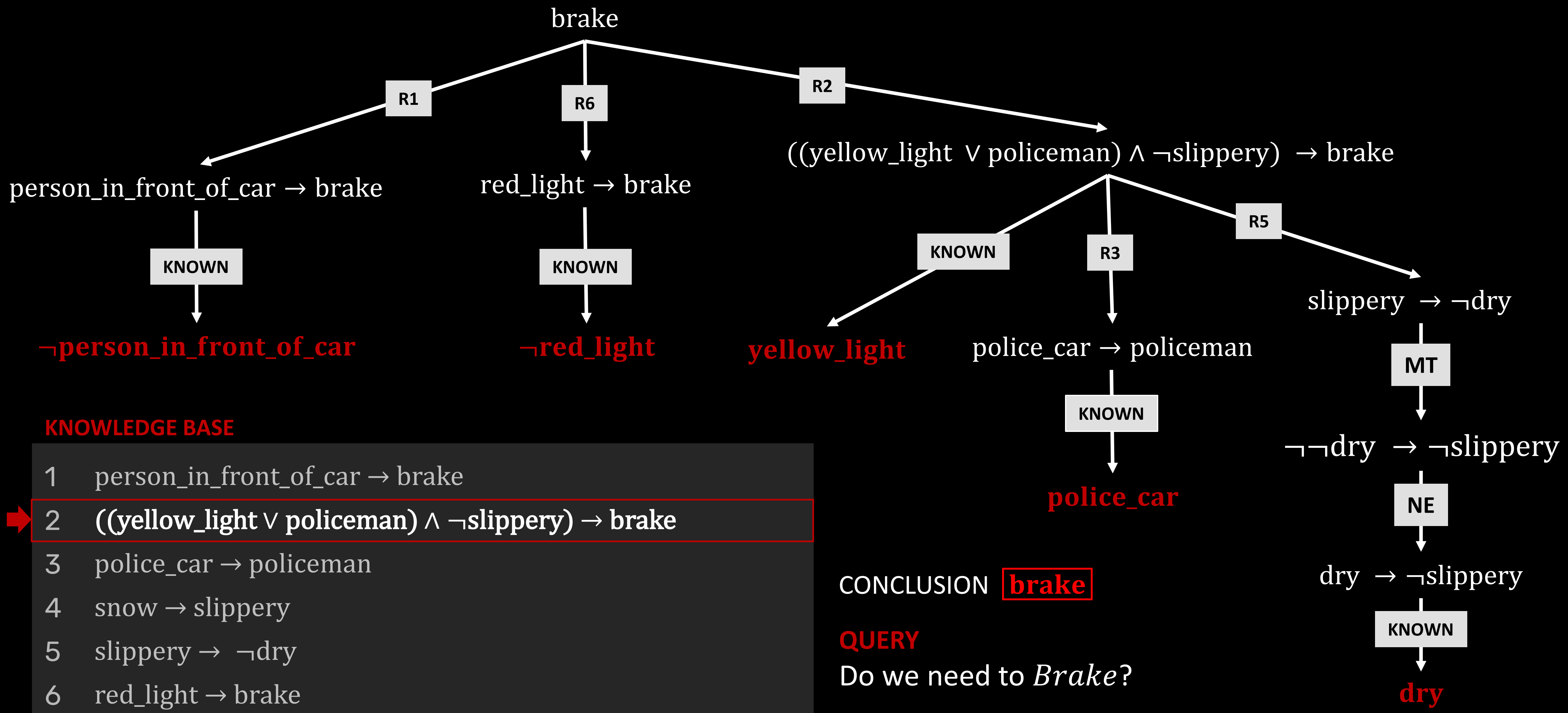
1	person_in_front_of_car → brake
2	**((yellow_light ∨ policeman) ∧ ¬slippery) → brake**
3	police_car → policeman
4	snow → slippery
5	slippery → ¬dry
6	red_light → brake
7	winter → snow

QUERY

Do we need to *Brake*?

FACTS

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



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 ∧ **dry** ∧
police_car ∧ ¬person_in_front_of_car

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**) ∨ (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

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

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

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

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

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

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

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

{ }

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 = $ **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**) $\lor$ (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, $R, M \cup \{P = \mathbf{F}\}$)

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

CLAUSES

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

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

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

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

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

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

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

{ }

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

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

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

CLAUSES

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

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

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

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

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

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

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

{ }

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**$\}$)

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

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

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

CLAUSES

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

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

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

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

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

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

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

{ }

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

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

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

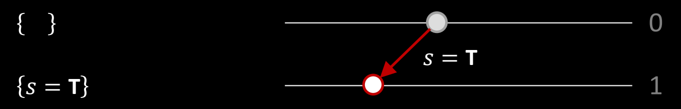
$\neg q \vee \neg r \vee s \quad \wedge$

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

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

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

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



DPLL(C, S, M):

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

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

CLAUSES

$p \lor q \lor r \lor \mathbf{s}$ $\land$

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

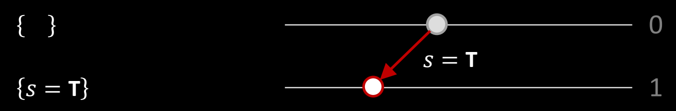
$\neg q \lor \neg r \lor \mathbf{s}$ $\land$

$p \lor \neg q \lor r \lor \mathbf{s}$ $\land$

$q \lor \neg r \lor \neg \mathbf{s}$ $\land$

$\neg p \lor \neg \mathbf{s}$ $\land$

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



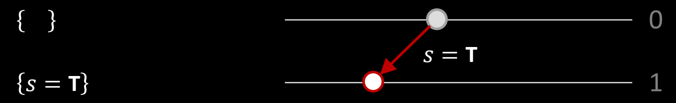
DPLL(C, S, M):

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



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

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

CLAUSES

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

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

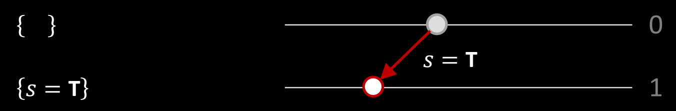
$\neg q \vee \neg r \vee s \quad \wedge \quad = \text{T}$

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

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

$\neg p \quad \wedge$

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



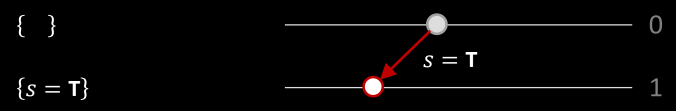
DPLL(C, S, M):

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

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

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

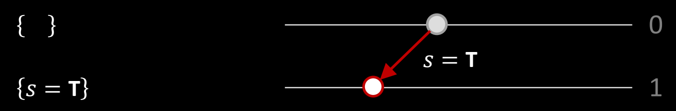
$\neg q \vee \neg r \vee s \quad \wedge \quad = \mathrm{T}$

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

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

$\neg p \quad \wedge$

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



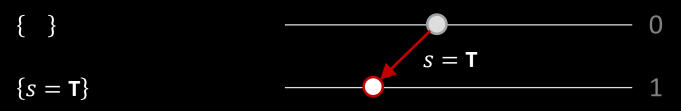
DPLL(C, S, M):

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, $R, M \cup \{P = \mathbf{F}\}$)

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

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):

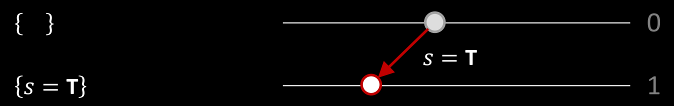
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(r, \mathbf{F})$

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):

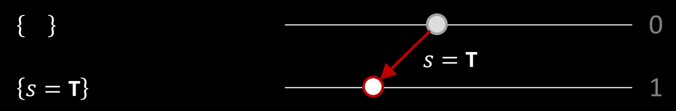
1. If (every c $\in C$ is **T**) ∨ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u, M \cup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

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

$(r, \mathbf{F})$

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



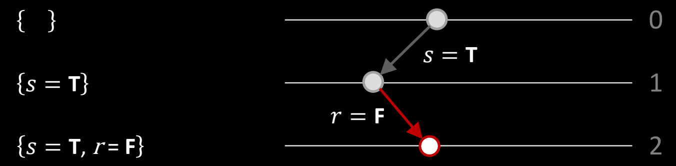
DPLL(C, S, M):

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

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



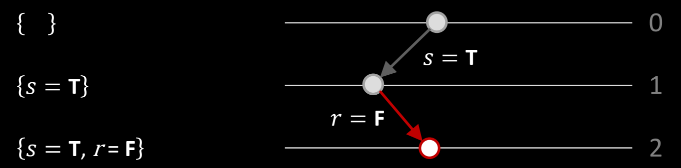
DPLL(C, S, M):

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

$p \vee q \vee r \vee s$	$\wedge$	= 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):

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

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

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

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

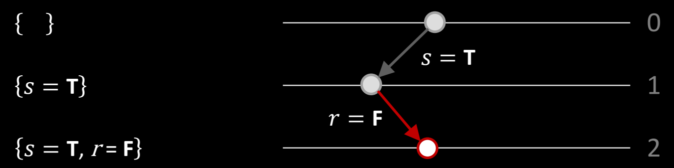
$\neg q \vee \neg r \vee s \quad \wedge \quad = \mathrm{T}$

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

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

$\neg p \quad \wedge$

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



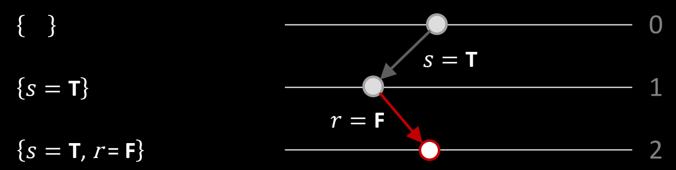
DPLL(C, S, M):

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($C, R, M \cup \{P = \textbf{T}\}$)
   $\vee$
   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

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

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

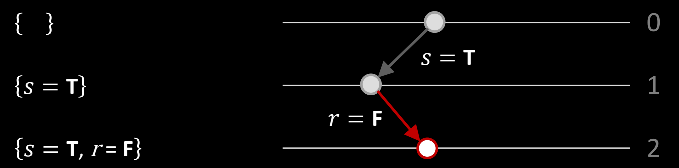
$\neg q \vee \neg r \vee s \quad \wedge \quad = \text{T}$

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

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

$\neg p \quad \wedge$

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



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

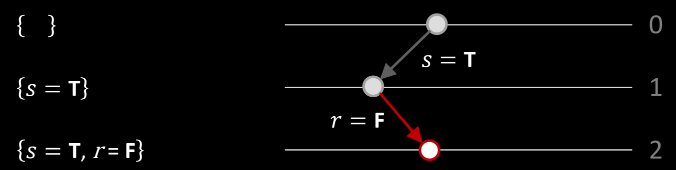
$\neg q \vee \neg r \vee s$ $\wedge$ = T

$p \vee \neg q \vee r \vee s$ $\wedge$ = 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($C, R, M \cup \{P =$ **T**$\}$)
   $\vee$
   DPLL(C, $R, M \cup \{P =$ **F**$\}$)

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

(q, **F**)

CLAUSES

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

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

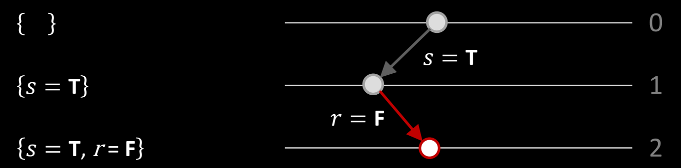
$\neg q \vee \neg r \vee s$ $\wedge$ = T

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

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

$\neg p$ $\wedge$

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



DPLL(C, S, M):

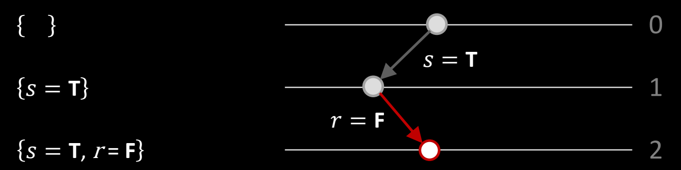
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u, M \cup \{u = v\}$)
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
$\neg p$	$\wedge$	
$p \vee \neg q$	$\wedge$	



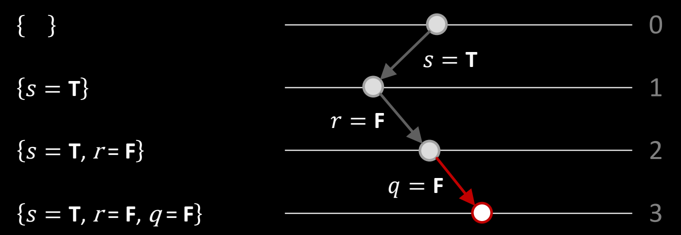
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\lor$ (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
$\neg p$	$\land$	
$p \lor \neg q$	$\land$	



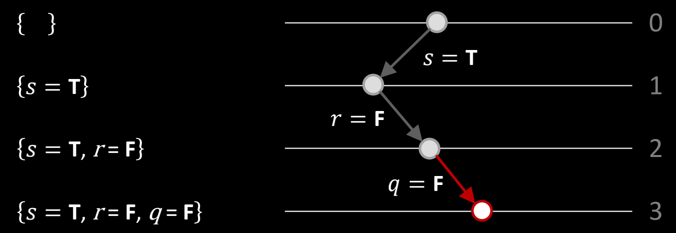
DPLL(C, S, M):

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



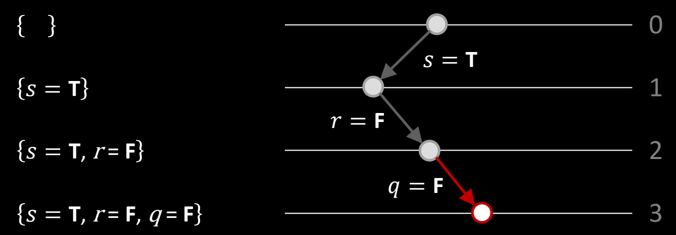
DPLL(C, S, M):

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

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

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



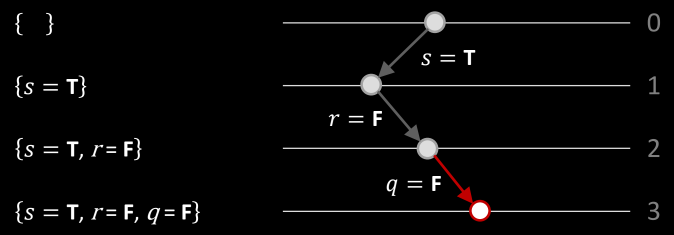
DPLL(C, S, M):

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



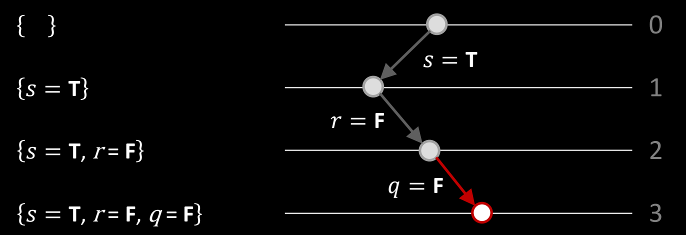
DPLL(C, S, M):

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, $R, M \cup \{P = \mathbf{F}\}$)

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

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



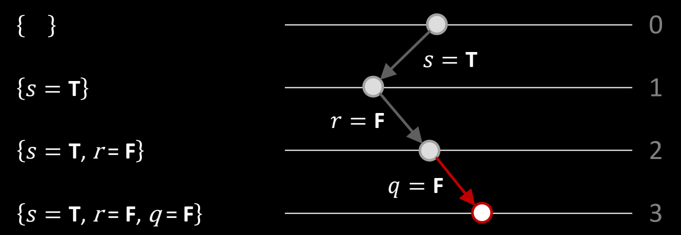
DPLL(C, S, M):

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

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

CLAUSES

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



DPLL(C, S, M):

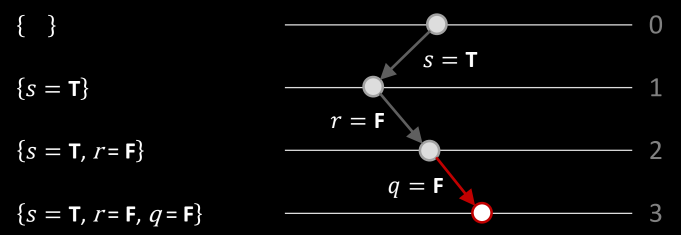
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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

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

$(p, \mathbf{F})$

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):

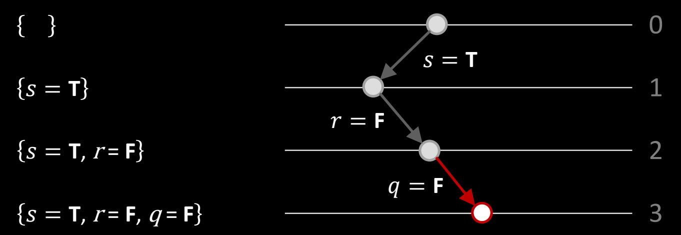
1. If (every c $\in C$ is **T**) $\vee$ (C is empty),
   return **T**
2. If C contains an empty clause,
   return **F**
3. If there is a (t, polarity v) = **pure symbol**(C),
   return DPLL($C, S - t, M \cup \{t = v\}$)
4. If there is a (u, polarity v) = **unit clause**(C),
   return DPLL($C, S - u,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



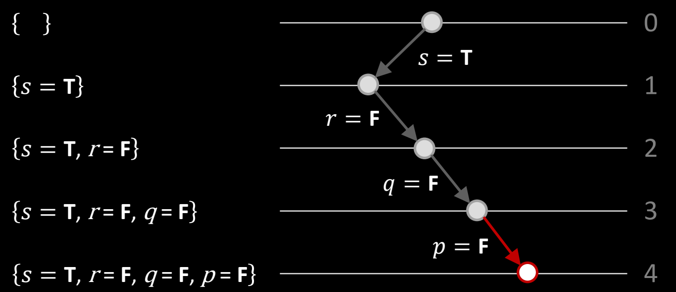
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) $\vee$ (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



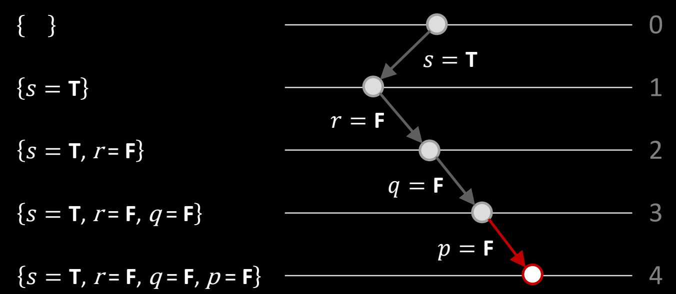
DPLL(C, S, M):

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



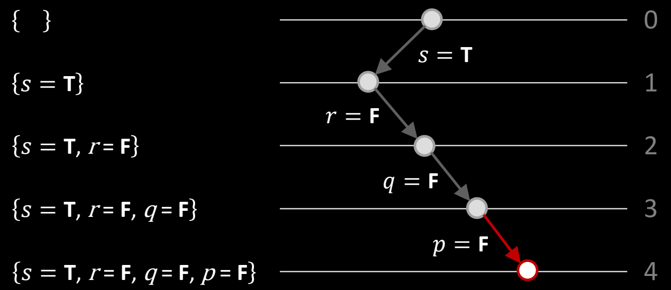
DPLL(C, S, M):

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



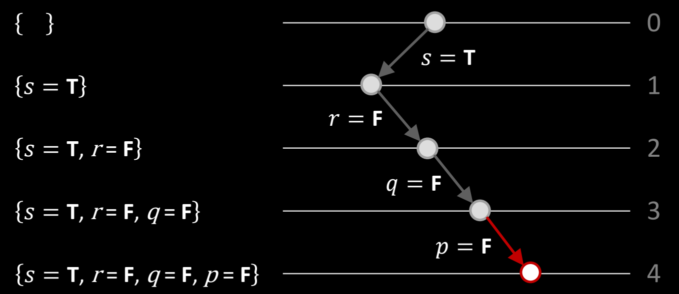
DPLL(C, S, M):

1. If (every c $\in C$ is **T**) ∨ (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

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



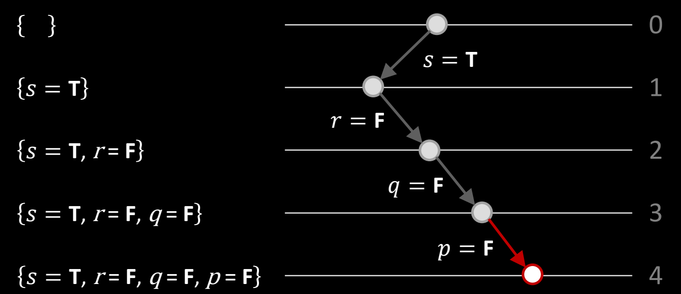
DPLL(C, S, M):

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, $R, M \cup \{P = \mathbf{F}\}$)

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

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 3.2