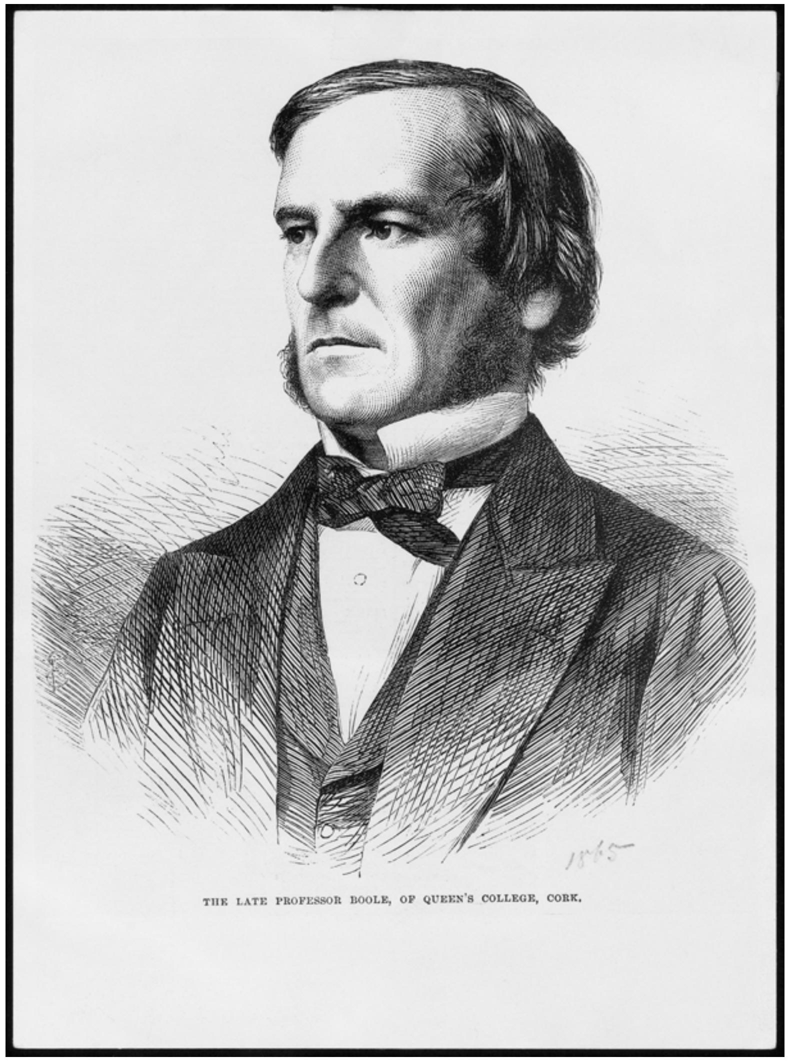
Propositional logic



Propositional logic

Vocabulary	Propositional letters: (aka Boolean variables)	$\Sigma = \{p, q, r, \ldots\}$
	Boolean connectives:	$\vee, \wedge, \neg, \rightarrow, \leftrightarrow$

Propositional logic

Vocabulary

Propositional letters: $\Sigma = \{p, q, r, \ldots\}$
(aka Boolean variables)

Boolean connectives: $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$

Syntax

$\phi: \quad p \mid q \mid \ldots \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi$

Propositional logic

Vocabulary

Propositional letters: $\Sigma = \{p, q, r, \ldots\}$
(aka Boolean variables)

Boolean connectives: $\vee, \wedge, \neg, \rightarrow, \leftrightarrow$

Syntax

$\phi : \quad p \mid q \mid \ldots \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi$

Semantics

Requires a model $M : \Sigma \rightarrow \{\text{true}, \text{false}\}$
Describes when ϕ holds on M $(M \vDash \phi)$

Propositional logic

Vocabulary

<u>Propositional letters</u>: $\Sigma = \{p, q, r, ...\}$
(aka Boolean variables)

<u>Boolean connectives</u>: $\vee, \wedge, \neg, \to, \leftrightarrow$

Syntax

$\phi : \quad p \mid q \mid \ldots \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \to \phi \mid \phi \leftrightarrow \phi$

Semantics

Requires a <u>model</u> $M : \Sigma \to \{\text{true}, \text{false}\}$
Describes when <u>ϕ holds on M</u> $(M \vDash \phi)$

$M \vDash p$	iff	$M(p) = \text{true}$
$M \vDash \phi_1 \vee \phi_2$	iff	$M \vDash \phi_1$ or $M \vDash \phi_2$
$M \vDash \phi_1 \wedge \phi_2$	iff	$M \vDash \phi_1$ and $M \vDash \phi_2$
$M \vDash \neg\phi$	iff	$M \nvDash \phi$
$M \vDash \phi_1 \to \phi_2$	iff	$M \nvDash \phi_1$ or $M \vDash \phi_2$

…

Logical equivalences

ϕ tautology	iff	for every model M, $M \models \phi$
ϕ contradiction	iff	for every model M, $M \not\models \phi$
ϕ_1 consequence of ϕ_2	iff	for every model M, $M \models \phi_1$ implies $M \models \phi_2$
ϕ_1 equivalent to ϕ_2	iff	for every model M, $M \models \phi_1$ iff $M \models \phi_2$

Logical equivalences

ϕ <u>tautology</u>	iff	for every model M, $M \models \phi$
ϕ <u>contradiction</u>	iff	for every model M, $M \not\models \phi$
ϕ_1 <u>consequence</u> of ϕ_2	iff	for every model M, $M \models \phi_1$ implies $M \models \phi_2$
ϕ_1 <u>equivalent</u> to ϕ_2	iff	for every model M, $M \models \phi_1$ iff $M \models \phi_2$

Example $\phi_1 \rightarrow \phi_2$ is equivalent to its <u>contrapositive</u> $(\neg\phi_2 \rightarrow \neg\phi_1)$
but not to its <u>converse</u> $(\phi_2 \rightarrow \phi_1)$

Algorithms

```
Model-check(φ,M)

    if φ = p then
        return M(p)

    else if φ = φ1 ∨ φ2 then
        return Model-check(φ1,M) OR
               Model-check(φ2,M)

    else if φ = φ1 ∧ φ2 then
        return Model-check(φ1,M) AND
               Model-check(φ2,M)

    else ...
```

Algorithms

```
Model-check(φ,M)

    if φ = p then
        return M(p)

    else if φ = φ1 ∨ φ2 then
        return Model-check(φ1,M) OR
               Model-check(φ2,M)

    else if φ = φ1 ∧ φ2 then
        return Model-check(φ1,M) AND
               Model-check(φ2,M)

    else ...
```

Complexity: **P**-complete

Algorithms

Satisfiable(φ)

```
let  p₁, p₂, ... = prop. letters in φ

for  M(p₁) ∈ {true, false}  do
    for  M(p₂) ∈ {true, false}  do
        ...
            if  Model-check(φ,M)  then
                return true

return false
```

Algorithms

Satisfiable(φ)

```
let p1, p2, ... = prop. letters in φ

for M(p1) ∈ {true, false} do
    for M(p2) ∈ {true, false} do
        ...
            if Model-check(φ,M) then
                return true

return false
```

Complexity: in **EXP** (actually, **NP**-complete ...)

Economy of syntax and normal forms

Syntax $\phi : \quad p \mid q \mid \ldots \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi \to \phi \mid \phi \leftrightarrow \phi$

Lemma $\wedge, \to, \leftrightarrow$ can be defined using $\vee, \neg$

Namely:

$\phi_1 \wedge \phi_2$ is equivalent to $\neg(\neg\phi_1 \vee \neg\phi_2)$

$\phi_1 \to \phi_2$ is equivalent to $\neg\phi_1 \vee \phi_2$

$\phi_1 \leftrightarrow \phi_2$ is equivalent to $\neg(\neg(\neg\phi_1 \vee \phi_2) \vee \neg(\neg\phi_2 \vee \phi_1))$

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\phi : \quad \phi \vee \phi \;\mid\; \phi \wedge \phi \;\mid\; \alpha$$
$$\alpha : \quad p \;\mid\; \neg p$$

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\phi : \quad \phi \vee \phi \ \mid \ \phi \wedge \phi \ \mid \ \alpha$$
$$\alpha : \quad p \ \mid \ \neg p$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Economy of syntax and normal forms

NNF (Negation Normal Form)

$$\begin{array}{ll} \phi: & \phi \vee \phi \mid \phi \wedge \phi \mid \alpha \\ \alpha: & p \mid \neg p \end{array}$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Proof Redefine $\rightarrow$ and push negations inside:

$$\begin{array}{rcl} \phi_1 \rightarrow \phi_2 & \rightsquigarrow & \neg\phi_1 \vee \phi_2 \\ \neg(\phi_1 \wedge \phi_2) & \rightsquigarrow & \neg\phi_1 \vee \neg\phi_2 \\ \neg(\phi_1 \vee \phi_2) & \rightsquigarrow & \neg\phi_1 \wedge \neg\phi_2 \end{array}$$

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\phi : \quad \phi \vee \phi \mid \phi \wedge \phi \mid \alpha$$
$$\alpha : \quad p \mid \neg p$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\phi: \quad \phi \vee \phi \mid \phi \wedge \phi \mid \alpha$$
$$\alpha: \quad p \mid \neg p$$

<u>CNF</u> (Conjunctive Normal Form)

$$\phi: \quad \phi \wedge \phi \mid \alpha$$
$$\alpha: \quad \alpha \vee \alpha \mid p \mid \neg p$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Economy of syntax and normal forms

NNF (Negation Normal Form)

$$\phi: \quad \phi \vee \phi \mid \phi \wedge \phi \mid \alpha$$
$$\alpha: \quad p \mid \neg p$$

CNF (Conjunctive Normal Form)

$$\phi: \quad \phi \wedge \phi \mid \alpha$$
$$\alpha: \quad \alpha \vee \alpha \mid p \mid \neg p$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Lemma Given ϕ, one can compute in polynomial time an *equi-satisfiable* formula ϕ^* in CNF

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\phi: \quad \phi \vee \phi \;\mid\; \phi \wedge \phi \;\mid\; \alpha$$
$$\alpha: \quad p \;\mid\; \neg p$$

<u>CNF</u> (Conjunctive Normal Form)

$$\phi: \quad \phi \wedge \phi \;\mid\; \alpha$$
$$\alpha: \quad \alpha \vee \alpha \;\mid\; p \;\mid\; \neg p$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Lemma Given ϕ, one can compute in polynomial time an *equi-satisfiable* formula ϕ^* in CNF

Corollary CNF-Sat is still **NP**-complete

Economy of syntax and normal forms

<u>NNF</u> (Negation Normal Form)

$$\begin{aligned} \phi &: \quad \phi \vee \phi \;\;|\;\; \phi \wedge \phi \;\;|\;\; \alpha \\ \alpha &: \quad p \;|\; \neg p \end{aligned}$$

<u>CNF</u> (Conjunctive Normal Form)

$$\begin{aligned} \phi &: \quad \phi \wedge \phi \;\;|\;\; \alpha \\ \alpha &: \quad \alpha \vee \alpha \;\;|\;\; p \;\;|\;\; \neg p \end{aligned}$$

<u>DNF</u> (Disjunctive Normal Form)

$$\begin{aligned} \phi &: \quad \phi \vee \phi \;\;|\;\; \alpha \\ \alpha &: \quad \alpha \wedge \alpha \;\;|\;\; p \;\;|\;\; \neg p \end{aligned}$$

Lemma Given ϕ ($\leftrightarrow$-free), one can compute in polynomial time an *equivalent* formula ϕ^* in NNF

Lemma Given ϕ, one can compute in polynomial time an *equi-satisfiable* formula ϕ^* in CNF

Corollary CNF-Sat is still **NP**-complete

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } \mathtt{M(p)} \in \{\mathsf{true}, \mathsf{false}\} \ldots\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

$\{ (p \vee \neg q) \wedge \neg p \}$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ ...$\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

$$\{\, (p \vee \neg q) \wedge \neg p \,\}$$

$$\downarrow$$

$$\{\, (p \vee \neg q) \wedge \neg p \,\}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ ...$\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

$$\boxed{\{(p \vee \neg q) \wedge \neg p\}}$$
$$\downarrow$$
$$\boxed{\{p \vee \neg q\ ,\ \neg p\}}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

$$\{\, (p \vee \neg q) \wedge \neg p \,\}$$
$$\downarrow$$
$$\{\, p \vee \neg q \;,\; \neg p \,\}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true, false}\} \dots\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \dots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

$$\begin{tikzcd}
 & \{(p \vee \neg q) \wedge \neg p\} \arrow[d] & \\
 & \{p \vee \neg q\,,\ \neg p\} \arrow[dl] \arrow[dr] & \\
\{p\,,\ \neg p\} & & \{\neg q\,,\ \neg p\}
\end{tikzcd}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$

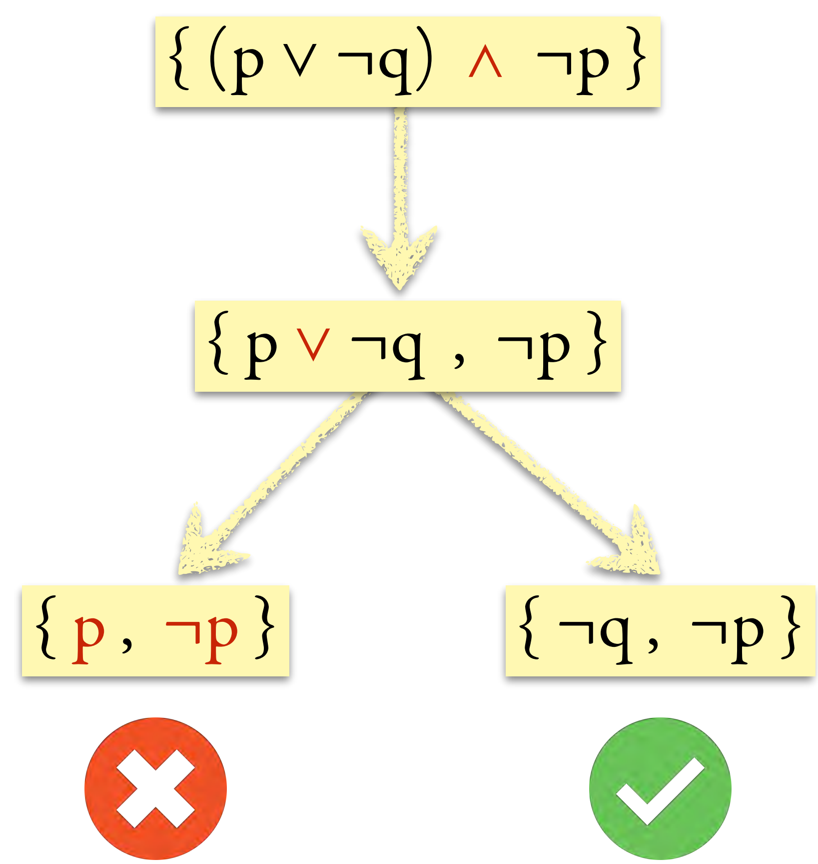
$$\begin{tikzcd}
 & \{ (p \vee \neg q) \wedge \neg p \} \arrow[d] & \\
 & \{ p \vee \neg q \,,\, \neg p \} \arrow[dl] \arrow[dr] & \\
\{ p \,,\, \neg p \} & & \{ \neg q \,,\, \neg p \}
\end{tikzcd}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge \neg p$



Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \dots\big)$
guess pieces of the formula to be satisfied $\big(\{a_1, a_2, \dots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$\{\, (p \vee \neg q) \wedge (\neg p \wedge q) \,\}$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for M(p) ∈ {true, false} …$\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\{ (p \vee \neg q) \wedge (\neg p \wedge q) \}$$

$$\downarrow$$

$$\{ (p \vee \neg q) \wedge (\neg p \wedge q) \}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ …$\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\{ (p \vee \neg q) \wedge (\neg p \wedge q) \}$$

$$\{ p \vee \neg q \ , \ \neg p \wedge q \}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model ($\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots$)

guess pieces of the formula to be satisfied ($\{\alpha_1, \alpha_2, \ldots\}$)

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\{(p \vee \neg q) \wedge (\neg p \wedge q)\}$$

$$\downarrow$$

$$\{p \vee \neg q \ , \ \neg p \wedge q\}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\begin{tikzcd}
 & \{\, (p \vee \neg q) \wedge (\neg p \wedge q) \,\} \arrow[d] & \\
 & \{\, p \vee \neg q \,,\, \neg p \wedge q \,\} \arrow[dl] \arrow[dr] & \\
\{\, p \,,\, \neg p \wedge q \,\} & & \{\, \neg q \,,\, \neg p \wedge q \,\}
\end{tikzcd}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\{\, (p \vee \neg q) \wedge (\neg p \wedge q) \,\}$$

$$\downarrow$$

$$\{\, p \vee \neg q \,,\, \neg p \wedge q \,\}$$

$$\swarrow \qquad \searrow$$

$$\{\, p \,,\, \neg p \wedge q \,\} \qquad \{\, \neg q \,,\, \neg p \wedge q \,\}$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big(\text{for } M(p) \in \{\text{true}, \text{false}\} \ldots\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$
\begin{tikzcd}
 & \{\, (p \vee \neg q) \wedge (\neg p \wedge q) \,\} \arrow[d] & \\
 & \{\, p \vee \neg q \,,\, \neg p \wedge q \,\} \arrow[dl] \arrow[dr] & \\
\{\, p \,,\, \neg p \wedge q \,\} \arrow[d] & & \{\, \neg q \,,\, \neg p \wedge q \,\} \arrow[d] \\
\{\, p \,,\, \neg p \wedge q \,\} & & \{\, \neg q \,,\, \neg p \wedge q \,\}
\end{tikzcd}
$$

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ ...$\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$
\begin{tikzcd}
 & \{ (p \vee \neg q) \wedge (\neg p \wedge q) \} \arrow[d] & \\
 & \{ p \vee \neg q \,,\, \neg p \wedge q \} \arrow[dl] \arrow[dr] & \\
\{ p \,,\, \neg p \wedge q \} \arrow[d] & & \{ \neg q \,,\, \neg p \wedge q \} \arrow[d] \\
\{ p \,,\, \neg p \,,\, q \} & & \{ \neg q \,,\, \neg p \,,\, q \}
\end{tikzcd}
$$

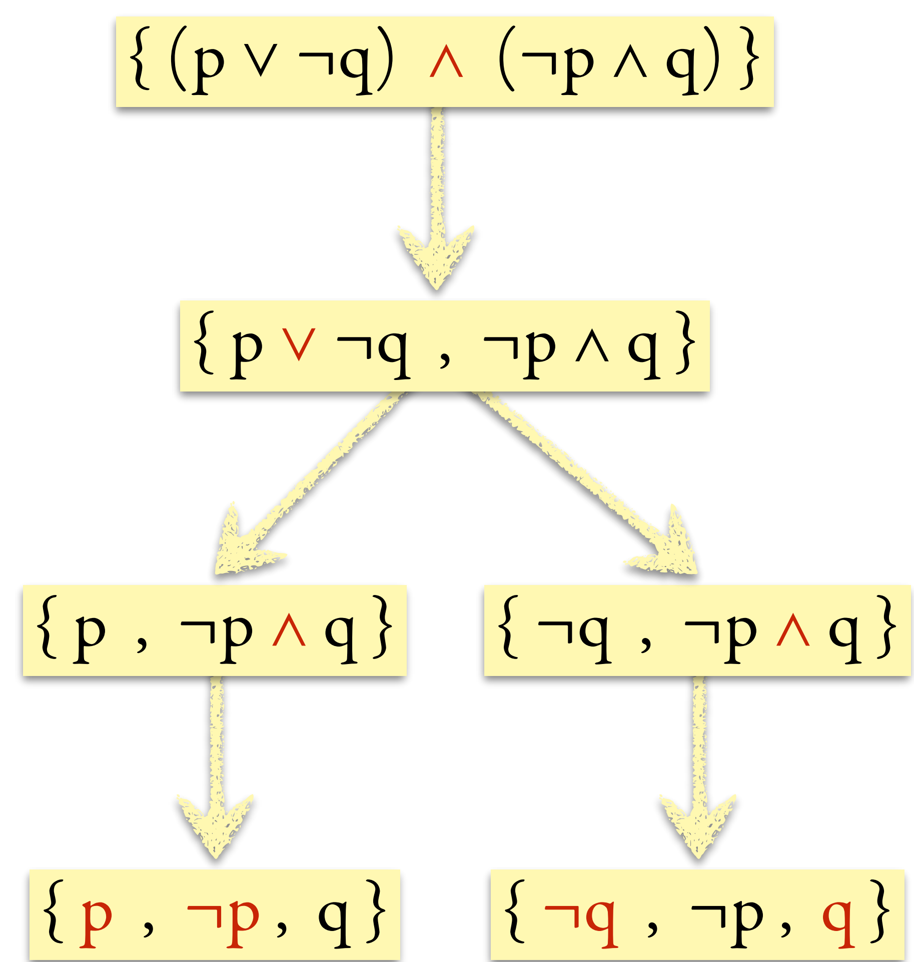
Note: different choices give different tableaux!

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ ...$\big)$

guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, \ldots\}\big)$

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$



Note: different choices give different tableaux!

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model (for $M(p) \in \{true, false\}$ ...)
guess pieces of the formula to be satisfied ($\{\alpha_1, \alpha_2, \dots\}$)

Example $\phi = (p \vee \neg q) \wedge (\neg p \wedge q)$

$$\{ (p \vee \neg q) \wedge (\neg p \wedge q) \}$$

$$\{ p \vee \neg q \,,\, \neg p \wedge q \}$$

$$\{ p \,,\, \neg p \wedge q \} \qquad \{ \neg q \,,\, \neg p \wedge q \}$$

$$\{ p \,,\, \neg p \,,\, q \} \qquad \{ \neg q \,,\, \neg p \,,\, q \}$$

Note: different choices give different tableaux!

Back to algorithms — Tableaux for satisfiability

Motto Instead of guessing pieces of a model $\big($for $M(p) \in \{\text{true}, \text{false}\}$ ...$\big)$
guess pieces of the formula to be satisfied $\big(\{\alpha_1, \alpha_2, ...\}\big)$

Tableaux construction:

1. assume ϕ is in Negation Normal Form (not necessary, but simplifies cases)
2. start with singleton tree $\{\, \phi \,\}$
3. choose <u>unblocked leaf</u> $F = \{\, \alpha_1, \alpha_2, ... \,\}$ and subformula $\alpha \in F$
4. expand tree under leaf F by adding
   - *one child* $F_1 = (F \setminus \alpha) \cup \{\, \beta, \gamma \,\}$ if $\alpha = \beta \wedge \gamma$
   - *two children* $F_1 = (F \setminus \alpha) \cup \{\, \beta \,\}$ and $F_2 = (F \setminus \alpha) \cup \{\, \gamma \,\}$ if $\alpha = \beta \vee \gamma$
5. new leaves that contain both p and $\neg p$ are declared <u>blocked</u>
6. stop with success as soon as there is an unblocked leaf with only <u>literals</u>

Application example — a simple transition system

Consider an electronic device whose internal state can be encoded by k bits

- *initial state* is described by a propositional formula $\phi_{\text{init}}(\bar{\mathsf{p}})$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{init}(\bar{p})$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{init}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by k bits

- *initial state* is described by a propositional formula $\phi_{\text{init}}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{\text{goal}}(\bar{p})$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{\text{init}}(\bar{p})$ $\phi_{\text{goal}}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{\text{init}}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{\text{goal}}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{\text{trans}}(\bar{p},\bar{p}')$

k bits $\left\{\right.$

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{\text{init}}(\bar{p})$ $\phi_{\text{trans}}(\bar{p},\bar{p}')$ $\phi_{\text{goal}}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{init}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{goal}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{trans}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{init}(\bar{p})$ $\phi_{trans}(\bar{p},\bar{p}')$ $\phi_{goal}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose internal state can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{init}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{goal}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{trans}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{init}(\bar{p})$ $\phi_{trans}(\bar{p},\bar{p}')$ $\phi_{goal}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{\text{init}}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{\text{goal}}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{\text{trans}}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{\text{init}}(\bar{p})$ $\phi_{\text{trans}}(\bar{p},\bar{p}')$ $\phi_{\text{goal}}(\bar{p})$

Application example — a simple transition system

Consider an electronic device whose internal state can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{\text{init}}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{\text{goal}}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{\text{trans}}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{\text{init}}(\bar{p})$ $\phi_{\text{trans}}(\bar{p},\bar{p}')$ $\phi_{\text{goal}}(\bar{p})$

Problem does the device admit a run from *initial state* to *desired final state* ?

Application example — a simple transition system

Consider an electronic device whose <u>internal state</u> can be encoded by k bits

- *initial state* is described by a propositional formula $\phi_{init}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{goal}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{trans}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{init}(\bar{p})$ $\phi_{trans}(\bar{p},\bar{p}')$ $\phi_{goal}(\bar{p})$

Problem does the device admit a <u>run</u> from *initial state* to *desired final state* ?

Note sufficient to consider runs of at most $n = 2^k$ transitions

Application example — a simple transition system

Consider an electronic device whose internal state can be encoded by *k* bits

- *initial state* is described by a propositional formula $\phi_{init}(\bar{p})$
- *desired final state* is described by another propositional formula $\phi_{goal}(\bar{p})$
- *discrete transitions* between states also described by a formula $\phi_{trans}(\bar{p},\bar{p}')$

k bits

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

$\phi_{init}(\bar{p})$ $\phi_{trans}(\bar{p},\bar{p}')$ $\phi_{goal}(\bar{p})$

Problem does the device admit a run from *initial state* to *desired final state* ?

Note sufficient to consider runs of at most $n = 2^k$ transitions

Test satisfiability of $\phi_{reach} = \phi_{init}[\bar{p}_1] \wedge \phi_{goal}[\bar{p}_n] \wedge \bigwedge_{i=2,\dots,n} \phi_{trans}[\bar{p}_{i-1},\bar{p}_i]$

Things to remember

Things to remember

- Propositional logic is very simple, yet useful

- Algorithms (model-checking and satisfiability) are conceptually simple

  but they can be presented and implemented in different ways (e.g. tableaux)

- There are efficiently computable normal forms (NNF, CNF)