بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه یازدهم: پیچیدگی محاسبات (۲)

Theory of computation - 002 - S11 - computational complexity (2)

Review & Reduction

Review & Reduction

NP = All languages where can verify membership quickly

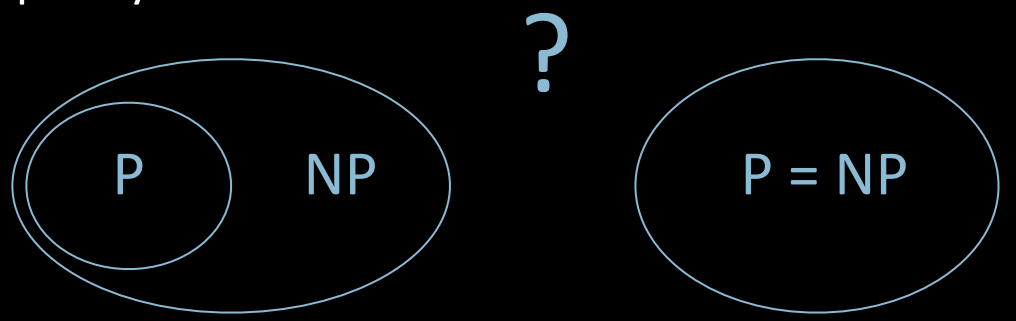
P = All languages where can test membership quickly

Review & Reduction

NP = All languages where can verify membership quickly

P = All languages where can test membership quickly

P versus NP question: Does P = NP?

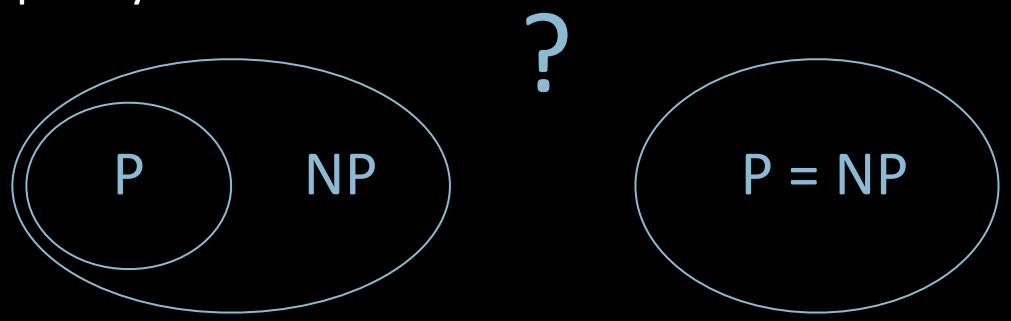


Review & Reduction

NP = All languages where can verify membership quickly

P = All languages where can test membership quickly

P versus NP question: Does P = NP?



Satisfiability Problem

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Defn: $SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Defn: $SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Theorem (Cook, Levin 1971): $SAT \in$ P $\rightarrow$ P = NP

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Defn: $SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Theorem (Cook, Levin 1971): $SAT \in$ P $\rightarrow$ P = NP
Proof method: polynomial time (mapping) reducibility

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Defn: $SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Theorem (Cook, Levin 1971): $SAT \in$ P $\rightarrow$ P = NP
Proof method: polynomial time (mapping) reducibility

Check-in 14.3

Satisfiability Problem

Defn: A *Boolean formula* ϕ has Boolean variables (TRUE/FALSE values) and Boolean operations AND ($\wedge$), OR ($\vee$), and NOT ($\neg$).

Defn: ϕ is *satisfiable* if ϕ evaluates to TRUE for some assignment to its variables.
Sometimes we use 1 for True and 0 for False.

Example: Let $\phi = (x \vee y) \wedge (\overline{x} \vee \overline{y})$ (Notation: $\overline{x}$ means $\neg x$)

Then ϕ is satisfiable (x=1, y=0)

Defn: $SAT = \{\langle \phi \rangle \mid \phi$ is a satisfiable Boolean formula$\}$

Theorem (Cook, Levin 1971): $SAT \in$ P $\rightarrow$ P = NP
Proof method: polynomial time (mapping) reducibility

Check-in 14.3

Is $SAT \in$ NP?

(a) Yes.

(b) No.

(c) I don't know.

(d) No one knows.

Check-in 14.3

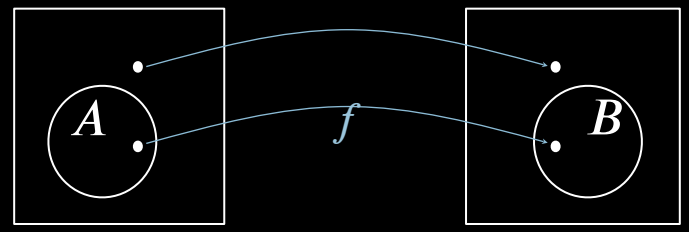
Polynomial Time Reducibility

Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

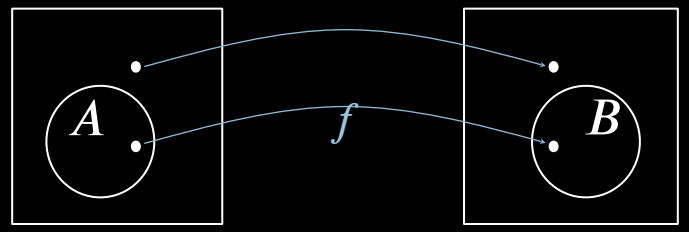


f is computable in polynomial time

Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.

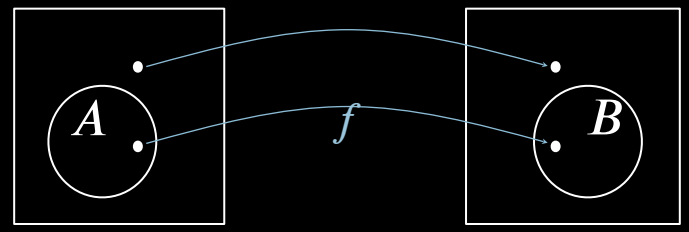


f is computable in polynomial time

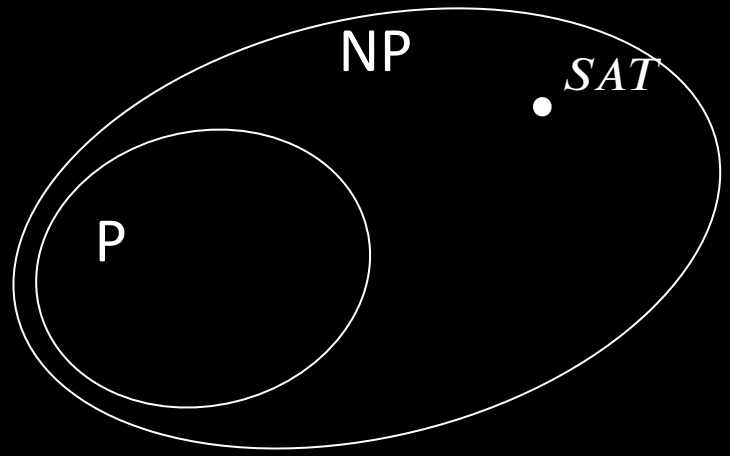
Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time

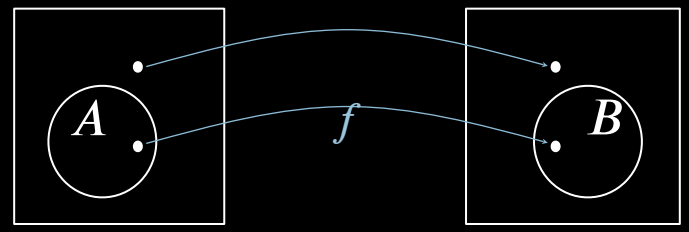


Idea to show $SAT \in$ P $\rightarrow$ P = NP

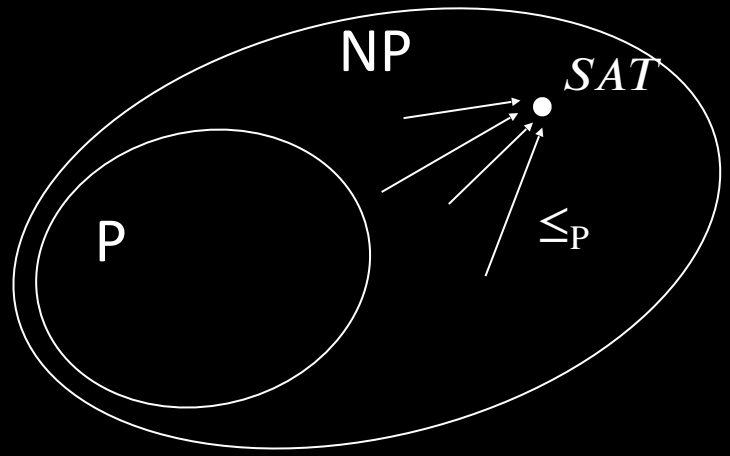
Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time

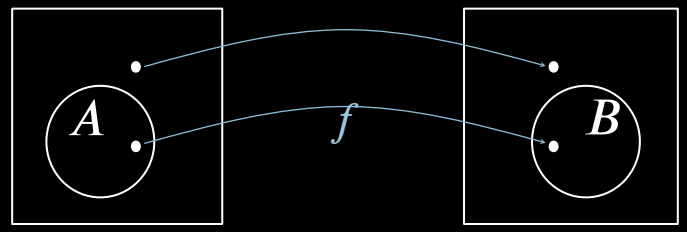


Idea to show $SAT \in$ P $\rightarrow$ P = NP

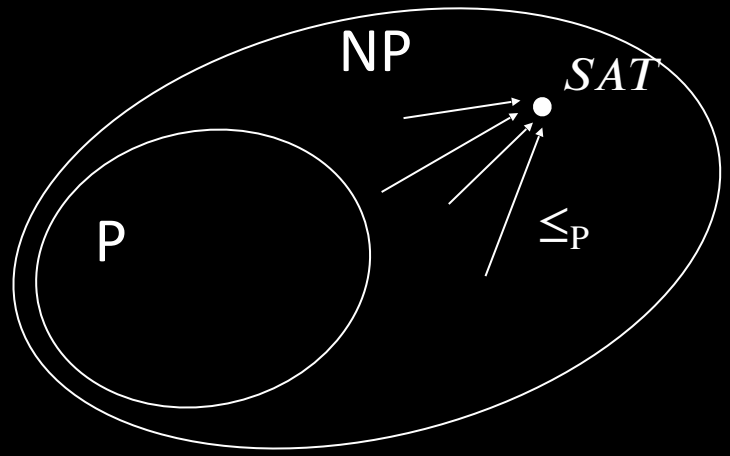
Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

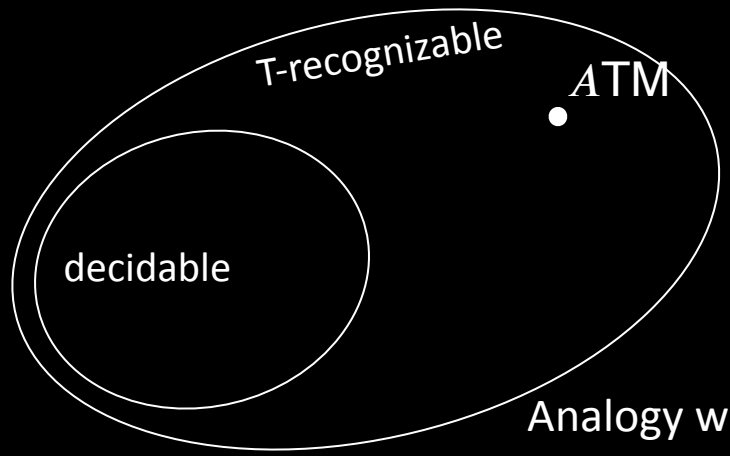
Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time



Idea to show $SAT \in$ P $\rightarrow$ P = NP

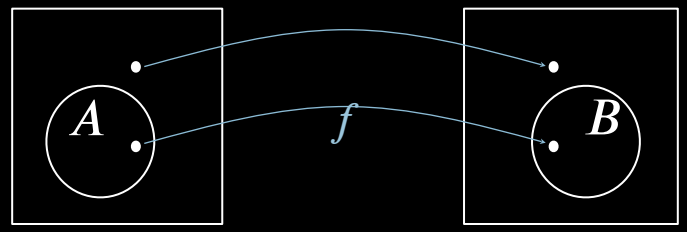


Analogy with ATM

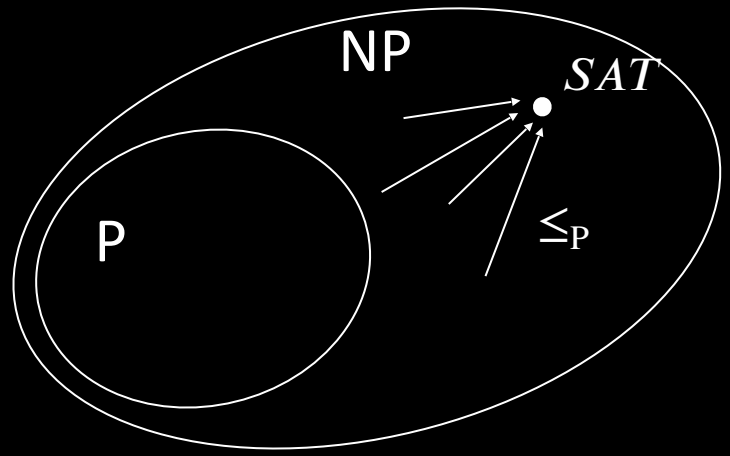
Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

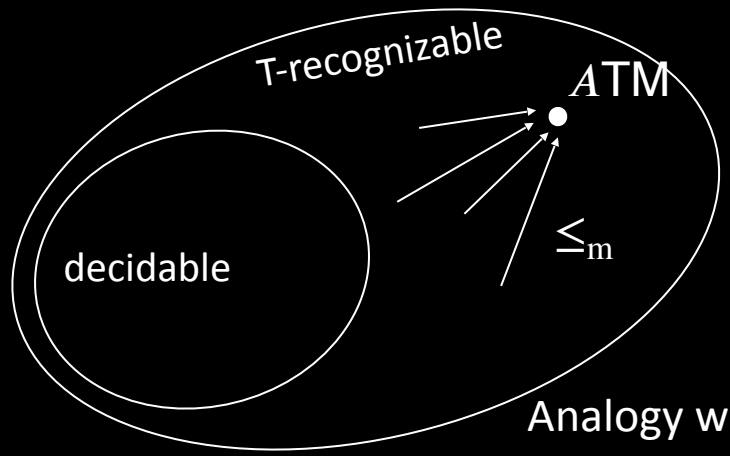
Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time



Idea to show $SAT \in$ P $\rightarrow$ P = NP

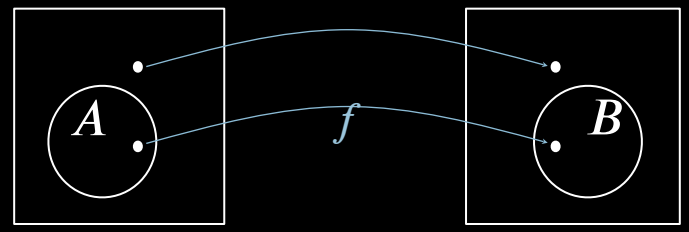


Analogy with ATM

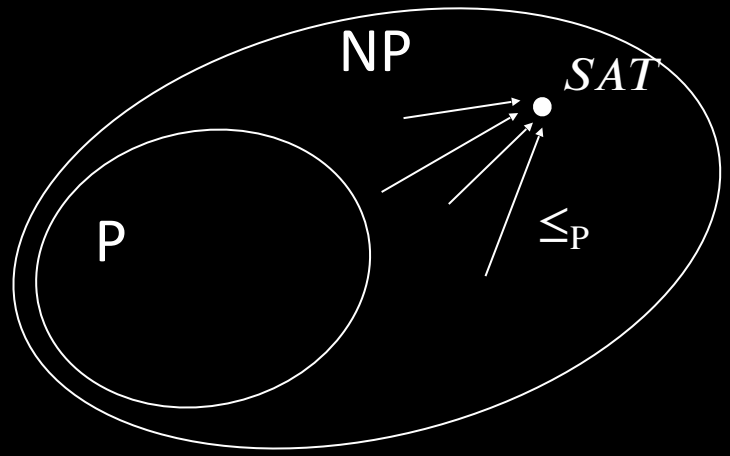
Polynomial Time Reducibility

Defn: A is polynomial time reducible to B ($A \leq_{\mathrm{P}} B$) if $A \leq_{\mathrm{m}} B$ by a reduction function that is computable in polynomial time.

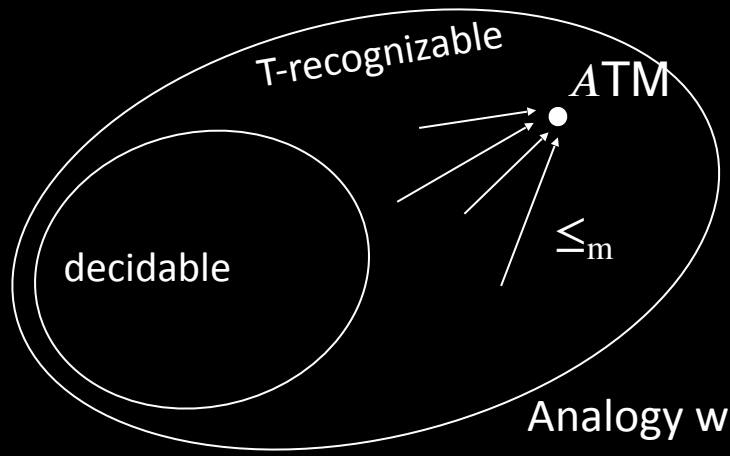
Theorem: If $A \leq_{\mathrm{P}} B$ and $B \in$ P then $A \in$ P.



f is computable in polynomial time



Idea to show $SAT \in$ P $\rightarrow$ P = NP



Analogy with ATM

$\leq_P$ Example: $3SAT$ and $CLIQUE$

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

literals

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause

literals

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

CNF: an AND ($\wedge$) of clauses.

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

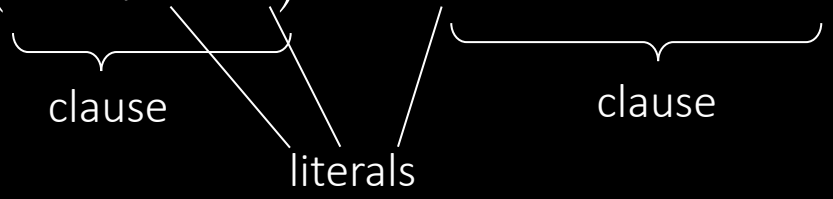
CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$



Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

CNF: an AND ($\wedge$) of clauses.

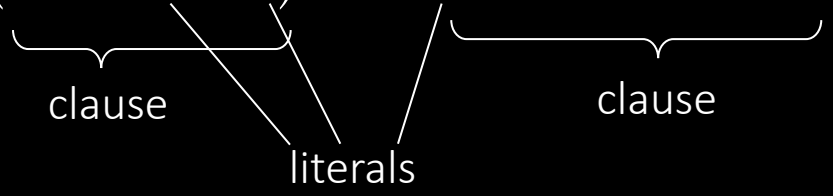
3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$



Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

Defn: A k-clique in a graph is a subset of $\mathbf{k}$ nodes all directly connected by edges.

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

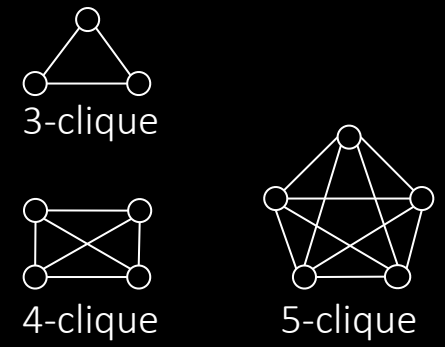
Clause: an OR ($\vee$) of literals.

CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

Defn: A k-clique in a graph is a subset of $\mathbf{k}$ nodes all directly connected by edges.



$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

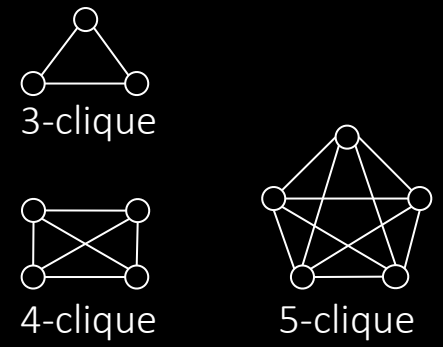
CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

Defn: A k-clique in a graph is a subset of $\mathbf{k}$ nodes all directly connected by edges.

$$CLIQUE = \{\langle G, k \rangle \mid \text{graph } G \text{ contains a } k\text{-clique}\}$$



$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

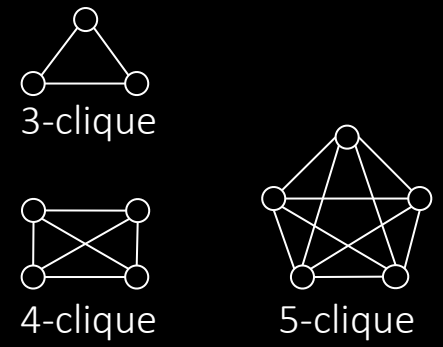
CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

Defn: A k-clique in a graph is a subset of $\mathbf{k}$ nodes all directly connected by edges.

$$CLIQUE = \{\langle G, k \rangle \mid \text{graph } G \text{ contains a } k\text{-clique}\}$$



Will show: $3SAT \leq_P CLIQUE$

$\leq_P$ Example: $3SAT$ and $CLIQUE$

Defn: A Boolean formula ϕ is in Conjunctive Normal Form (CNF) if it has the form

$$\phi = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{s} \vee z \vee u) \wedge \cdots \wedge (\bar{z} \vee \bar{u})$$

clause clause literals

Literal: a variable or a negated variable

Clause: an OR ($\vee$) of literals.

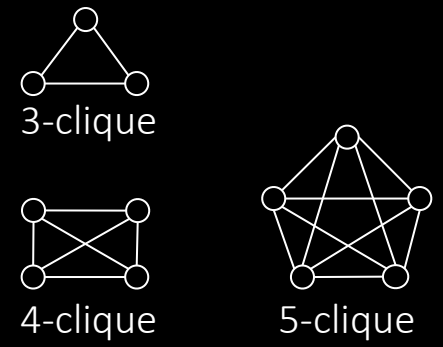
CNF: an AND ($\wedge$) of clauses.

3CNF: a CNF with exactly 3 literals in each clause.

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF formula}\}$$

Defn: A k-clique in a graph is a subset of $\mathbf{k}$ nodes all directly connected by edges.

$$CLIQUE = \{\langle G, k \rangle \mid \text{graph } G \text{ contains a } k\text{-clique}\}$$



Will show: $3SAT \leq_P CLIQUE$

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k where ϕ is satisfiable iff G has a k-clique.

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k where ϕ is satisfiable iff G has a k-clique.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

G

k

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$$\boxed{\begin{matrix} G \\ \\ k \end{matrix}} =$$

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

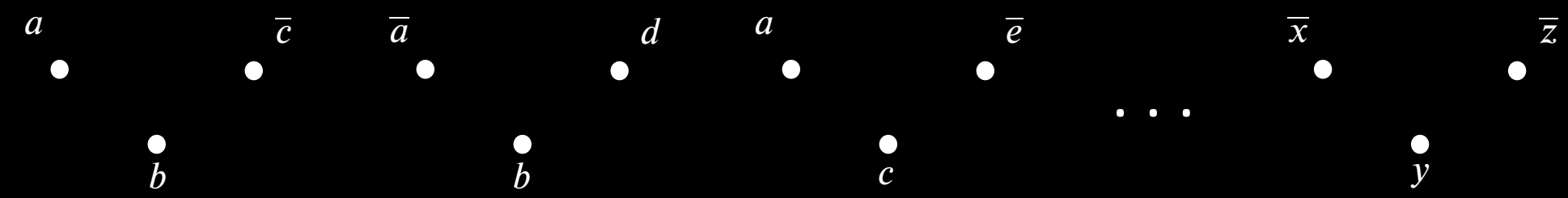
A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k



$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

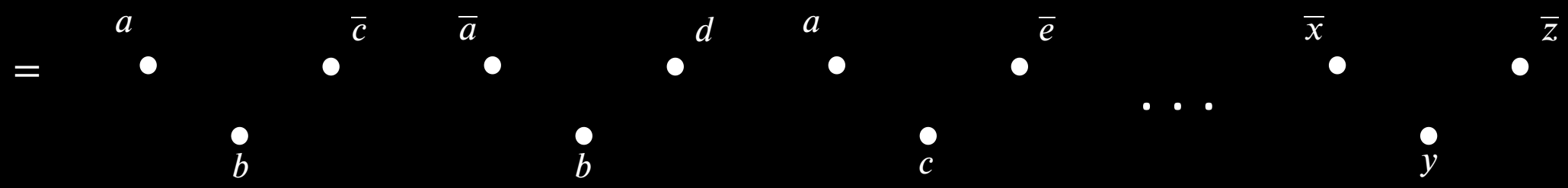
A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k



Forbidden edges:

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

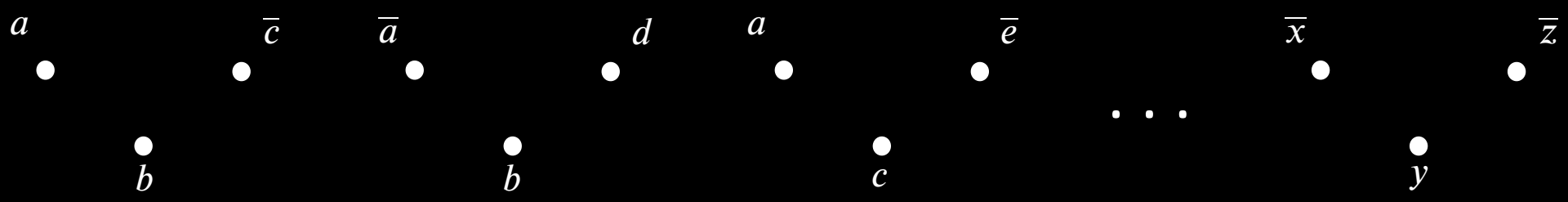
Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G \quad k}$ =



Forbidden edges:
1) within a clause

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

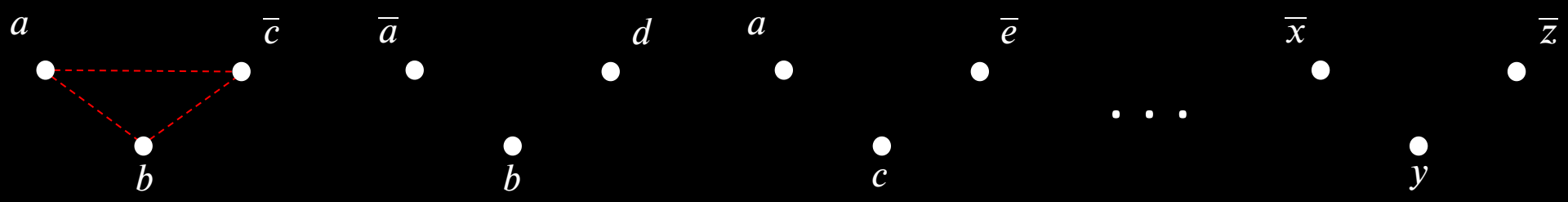
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

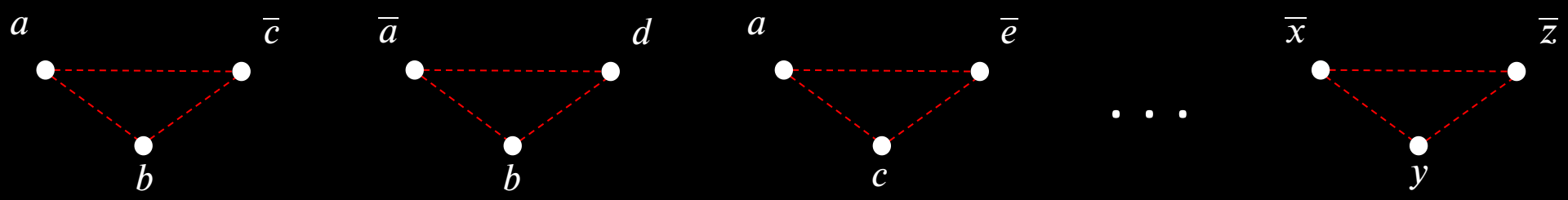
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

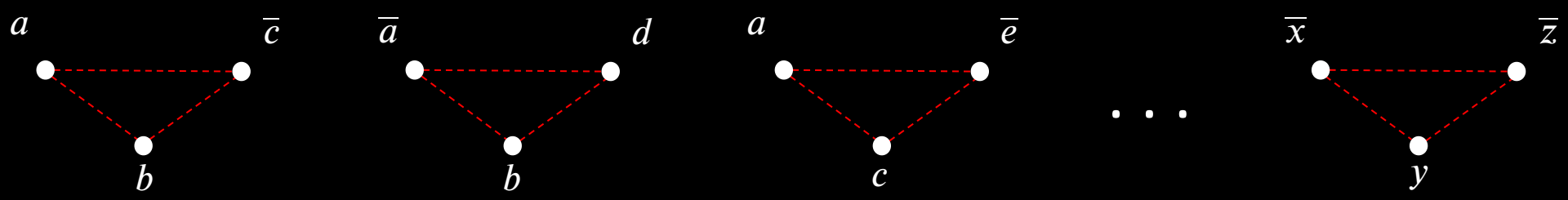
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

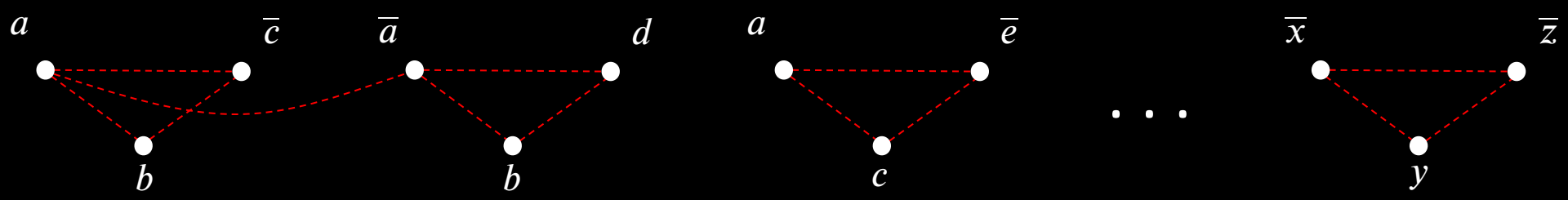
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

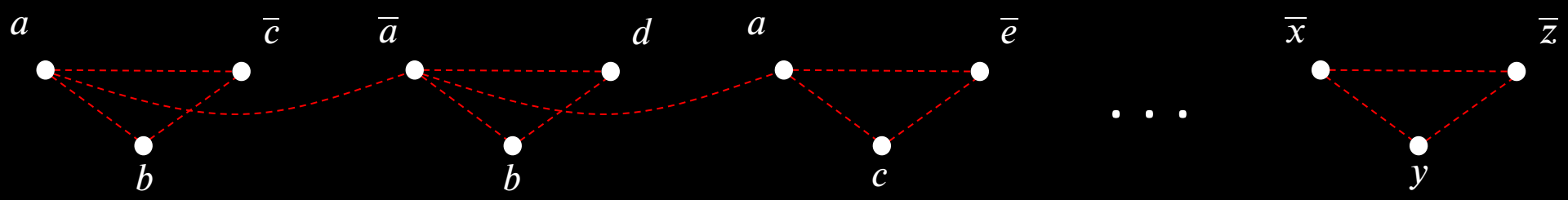
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

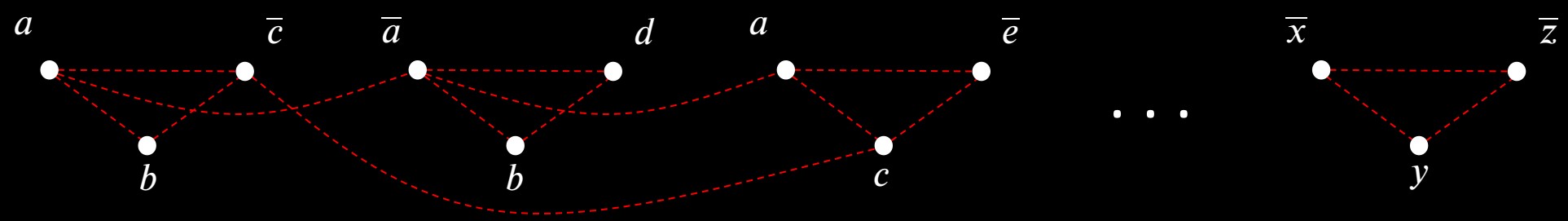
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

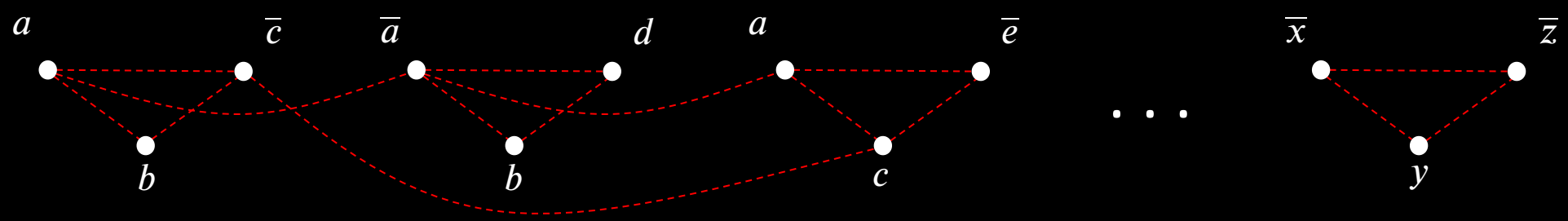
Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

$\boxed{G \quad k}$ =



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

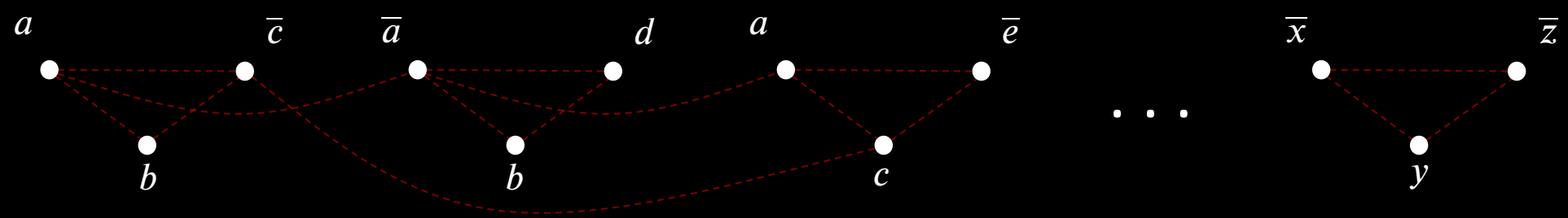
Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

$\boxed{G \quad k}$ =



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

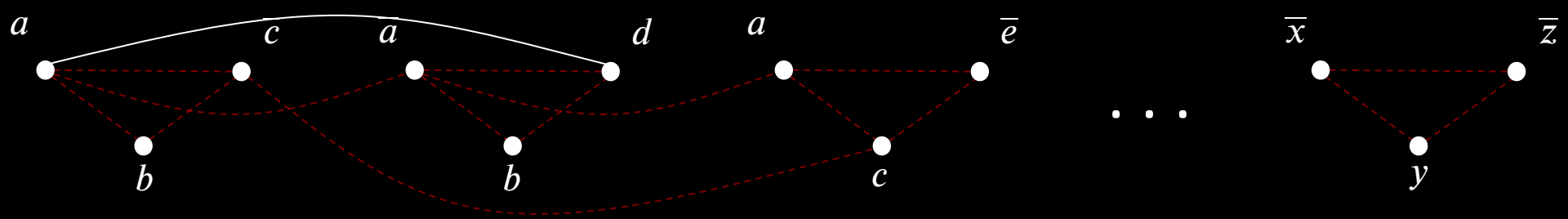
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

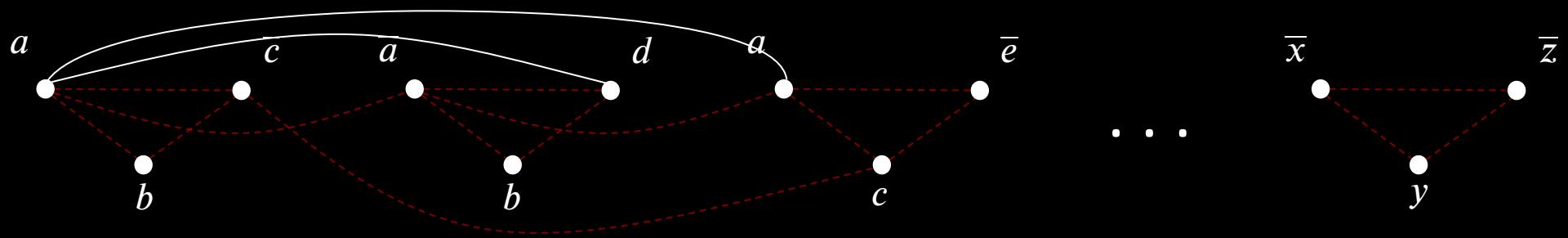
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

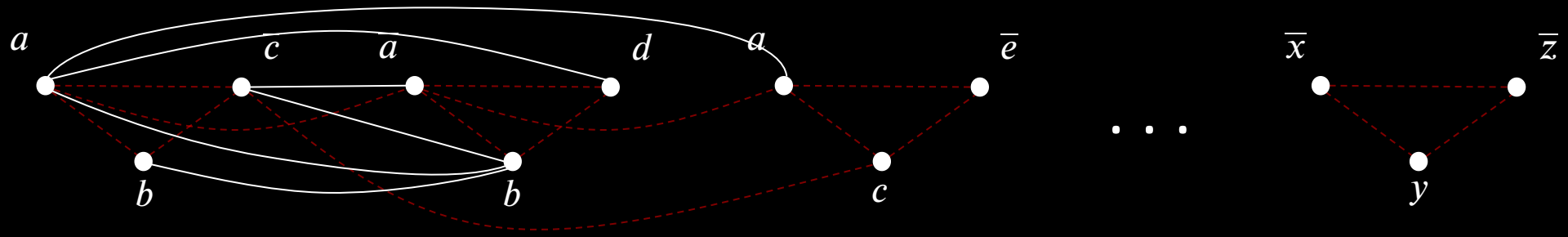
$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f

G

k

=



Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

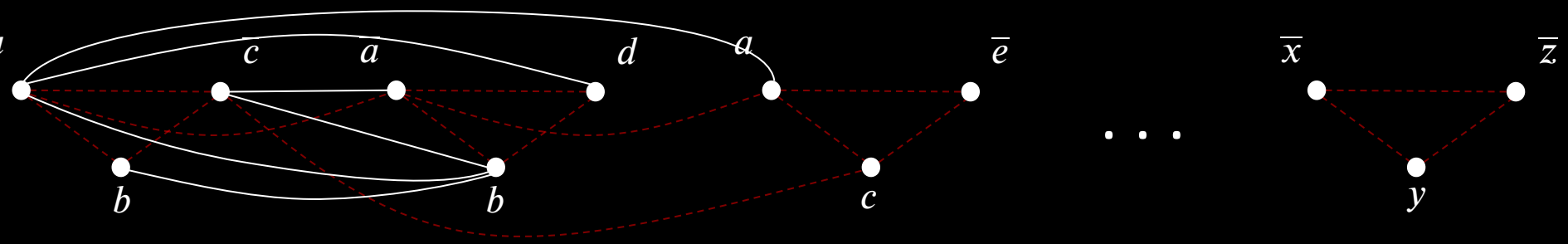
Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G = 

k = # clauses

Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

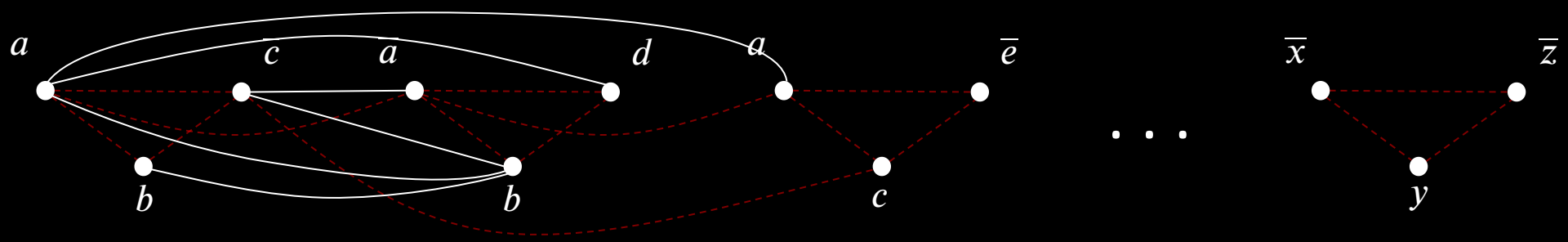
Theorem: $3SAT \leq_P CLIQUE$

Proof: Give polynomial-time reduction f that maps ϕ to G, k
where ϕ is satisfiable iff G has a k-clique.

A satisfying assignment to a CNF formula has ≥1 true literal in each clause.

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

f ↓

G = 

k = # clauses

Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\bar{a}$)

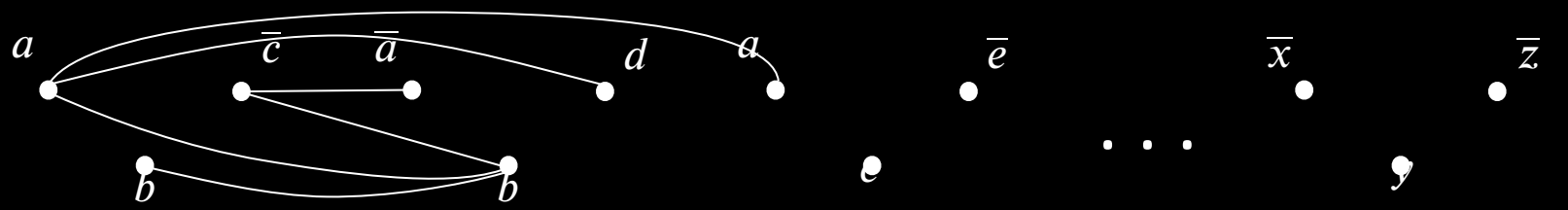
G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

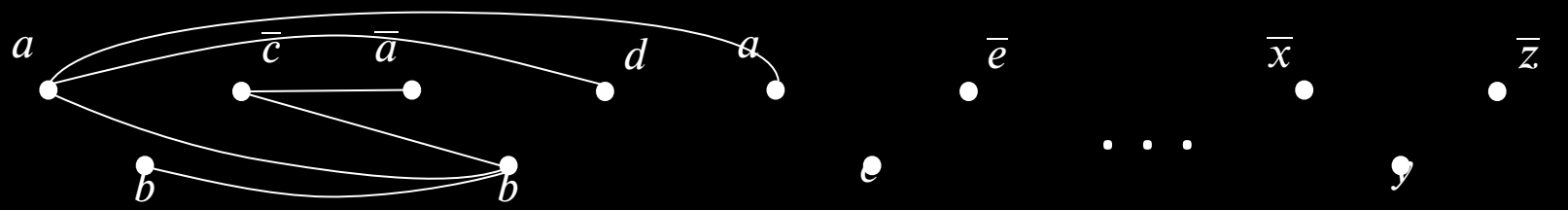
Claim: ϕ is satisfiable iff G has a k-clique

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

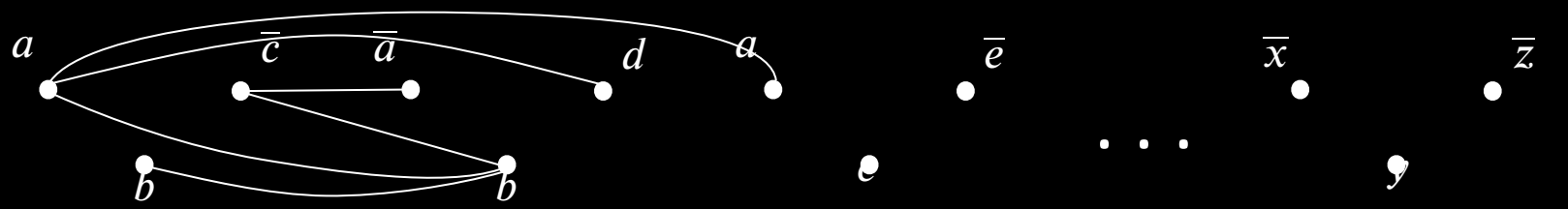
($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} = \boxed{k} =$ # clauses



Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

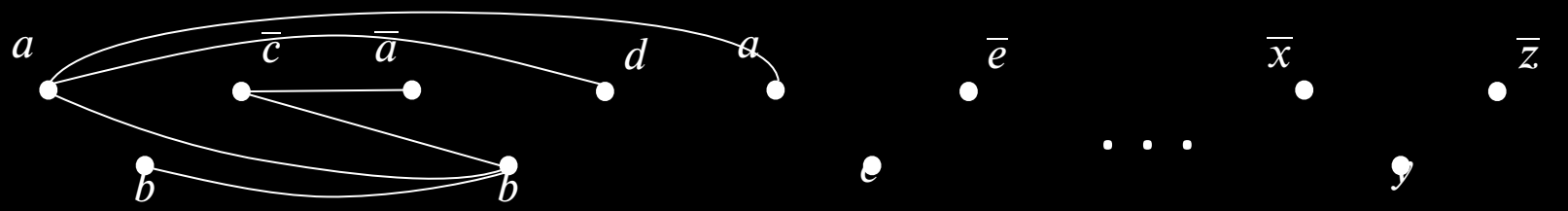
The corresponding nodes in G are a k-clique because they don't have forbidden edges.

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

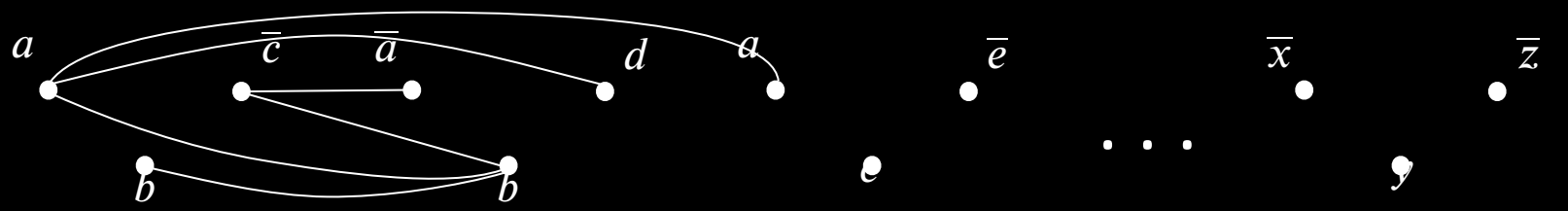
($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

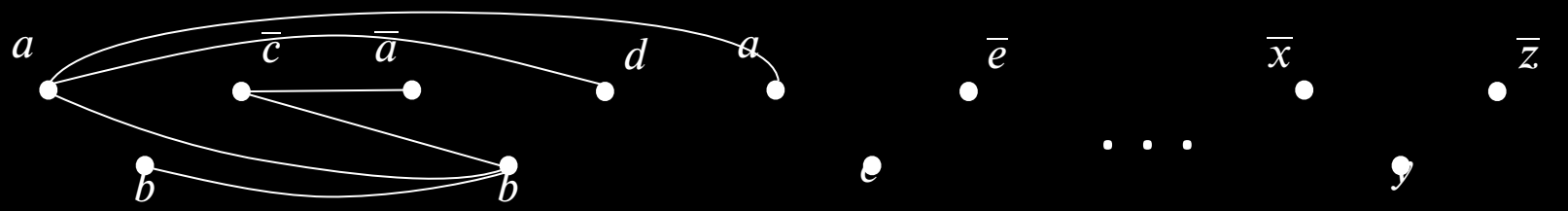
Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

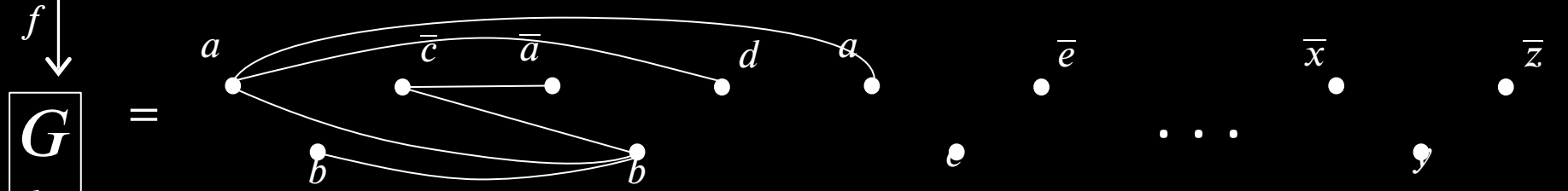
Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

The reduction f is computable in polynomial time.

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$



k = # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

The reduction f is computable in polynomial time.

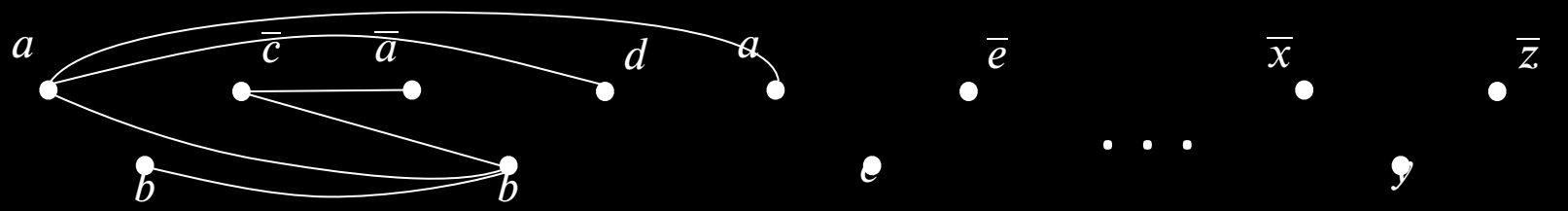
Corollary: $CLIQUE \in \mathrm{P} \rightarrow 3SAT \in \mathrm{P}$

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

The reduction f is computable in polynomial time.

Corollary: $CLIQUE \in \mathrm{P} \rightarrow 3SAT \in \mathrm{P}$

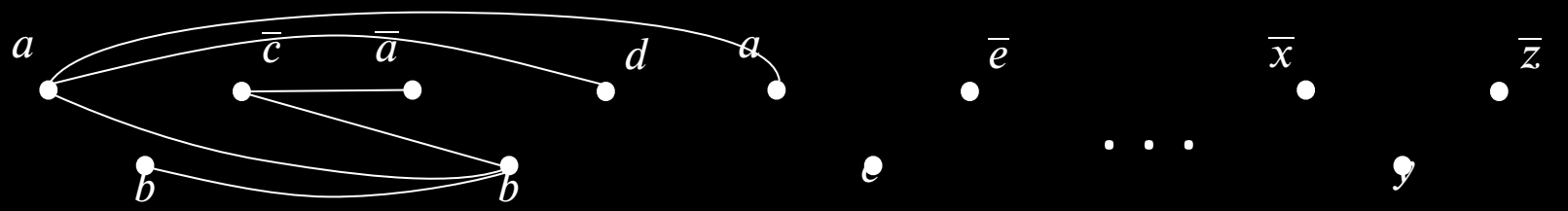
Check-in 15.1

$3SAT \leq_P CLIQUE$ conclusion

$$\phi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee b \vee d) \wedge (a \vee c \vee \bar{e}) \wedge \cdots \wedge (\bar{x} \vee y \vee \bar{z})$$

$f \downarrow$

$\boxed{G} =$



$\boxed{k} =$ # clauses

Claim: ϕ is satisfiable iff G has a k-clique

($\rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause.

The corresponding nodes in G are a k-clique because they don't have forbidden edges.

($\leftarrow$) Take any k-clique in G. It must have 1 node in each clause.

Set each corresponding literal TRUE. That gives a satisfying assignment to ϕ.

The reduction f is computable in polynomial time.

Corollary: $CLIQUE \in \mathrm{P} \rightarrow 3SAT \in \mathrm{P}$

Check-in 15.1

Does this proof require 3 literals per clause?

(a) Yes, to prove the claim.

(b) Yes, to show it is in poly time.

(c) No, it works for any size clauses.

Check-in 15.1

NP-completeness

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_{\mathrm{P}} B$

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_{\mathrm{P}} B$

If B is NP-complete and $B \in$ P then P = NP.

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

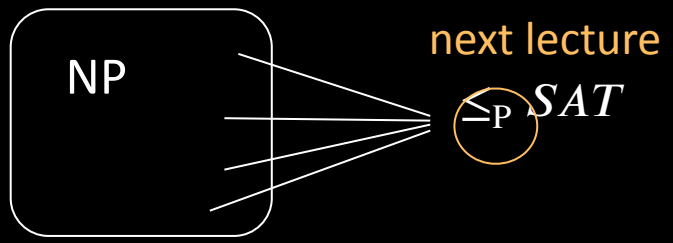
Proof: Next lecture; assume true

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Cook-Levin Theorem: SAT is NP-complete

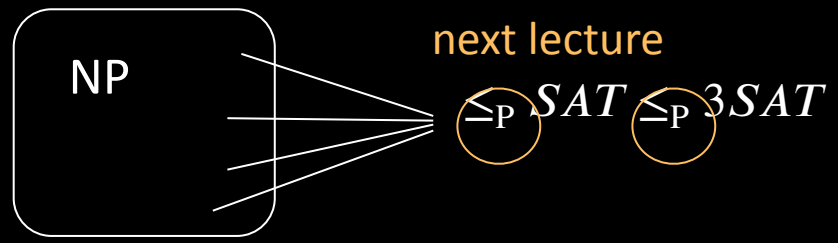
Proof: Next lecture; assume true

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true

NP-completeness

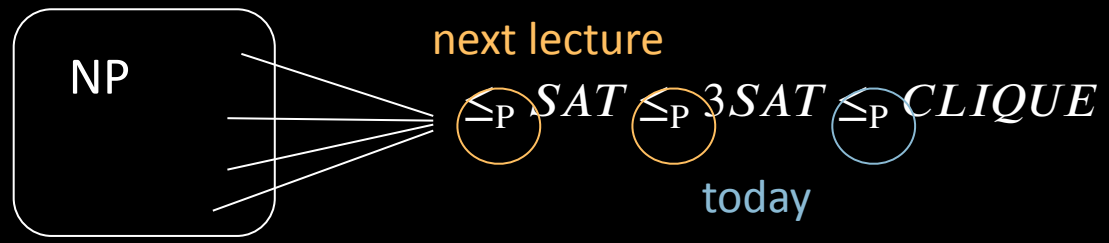
Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



NP-completeness

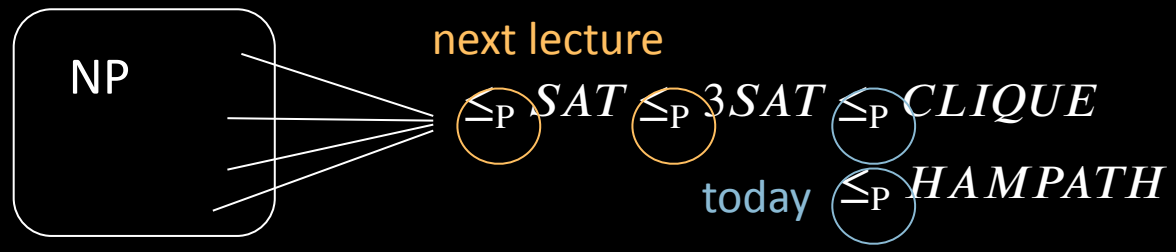
Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



NP-completeness

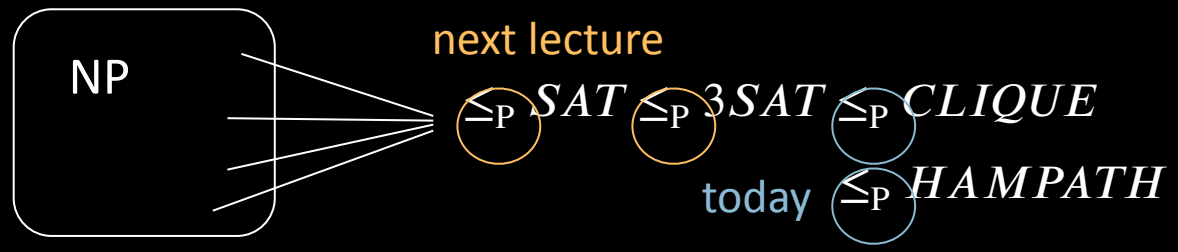
Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



To show some language C is NP-complete, show $3SAT \leq_P C$.

NP-completeness

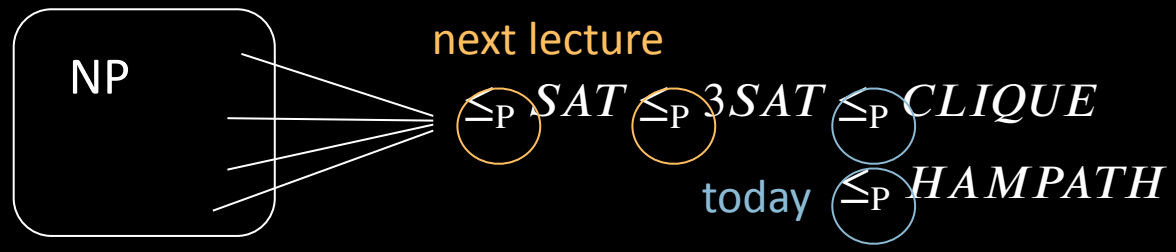
Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

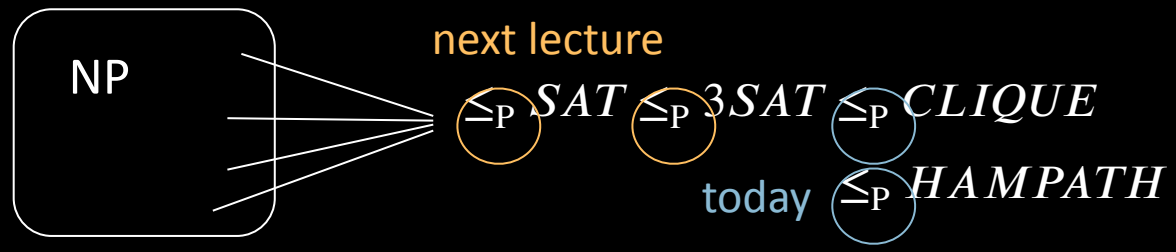
NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete
Proof: Next lecture; assume true



To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

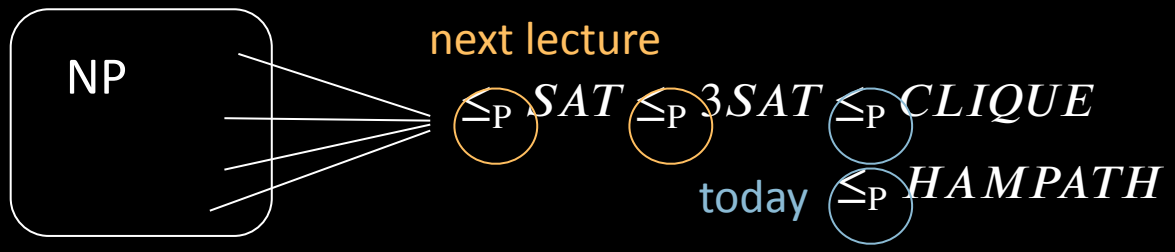
Importance of NP-completeness

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Cook-Levin Theorem: SAT is NP-complete
Proof: Next lecture; assume true

To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Importance of NP-completeness

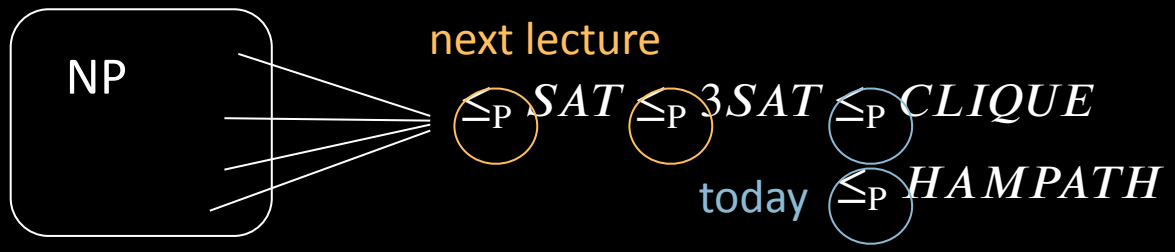
1) Showing B is NP-complete is evidence of computational intractability.

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Cook-Levin Theorem: SAT is NP-complete
Proof: Next lecture; assume true

To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Importance of NP-completeness

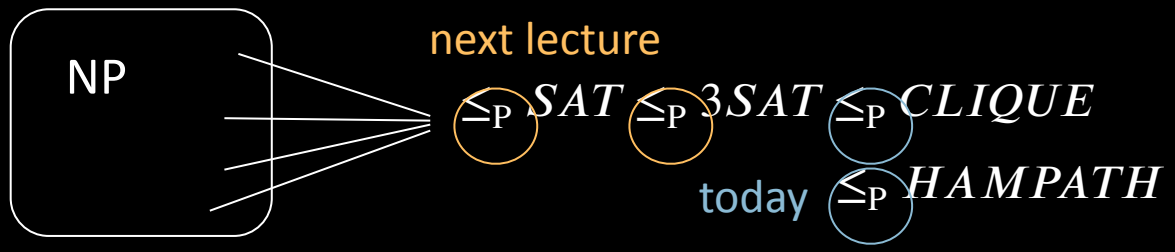
1) Showing B is NP-complete is evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

NP-completeness

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Cook-Levin Theorem: SAT is NP-complete
Proof: Next lecture; assume true

To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Importance of NP-completeness

1) Showing B is NP-complete is evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

Check-in 15.2

NP-completeness

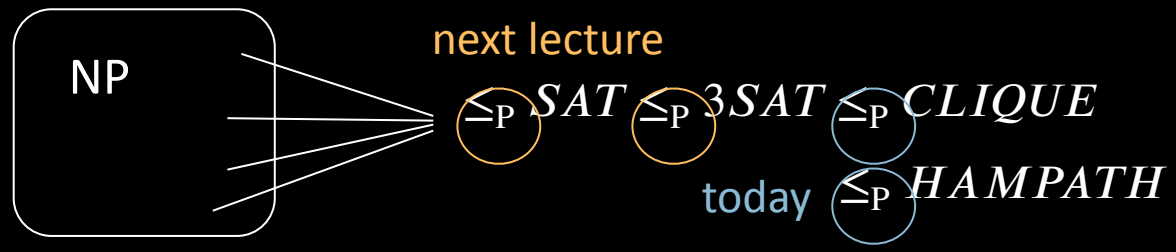
Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Cook-Levin Theorem: SAT is NP-complete

Proof: Next lecture; assume true



To show some language C is NP-complete, show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Check-in 15.2

What language that we've previously seen is most analogous to SAT?

(a) A_{TM}

(b) E_{TM}

(c) $\{0^k 1^k \mid k \geq 0\}$

Check-in 15.2

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.

Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete,
show $3SAT \leq_P C$.

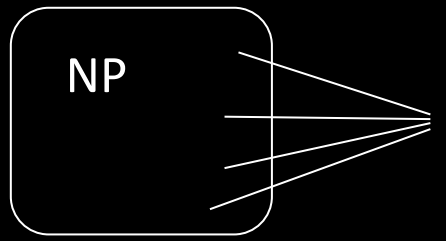
$3SAT$ — or some other previously shown NP-complete language

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



$\leq_P SAT \leq_P 3SAT \leq_P CLIQUE$

$\leq_P SUBSET\text{-}SUM$

$\leq_P HAMPATH$

$\leq_P UHAMPATH$

Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete,

show $3SAT \leq_P C$.

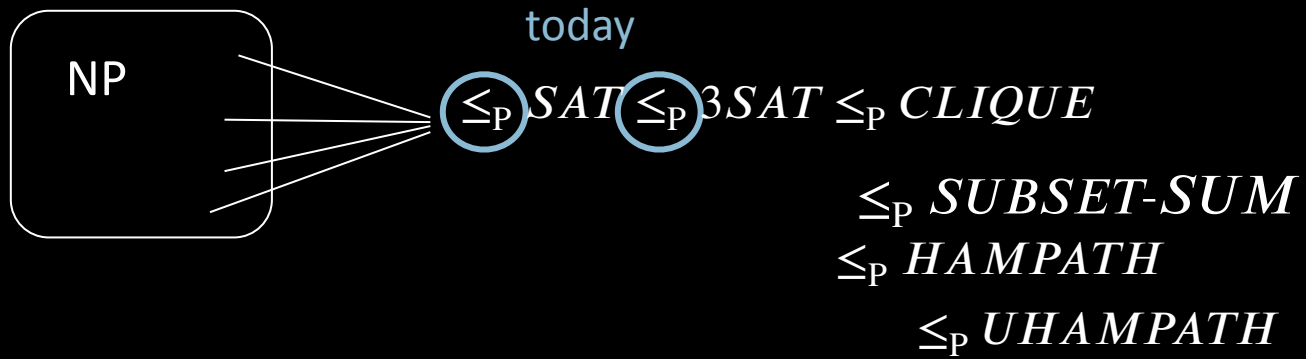
or some other previously shown NP-complete language

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete,
show $3SAT \leq_P C$.

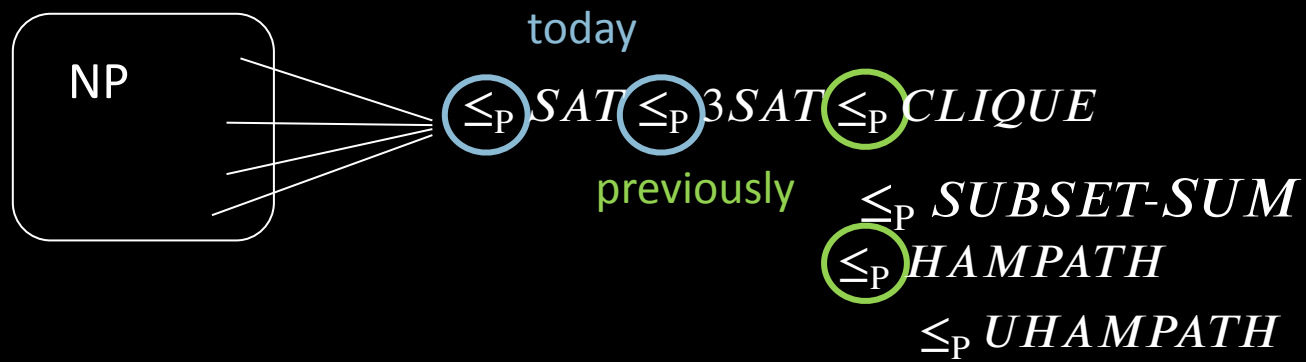
or some other previously shown NP-complete language

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete,
show $3SAT \leq_P C$.

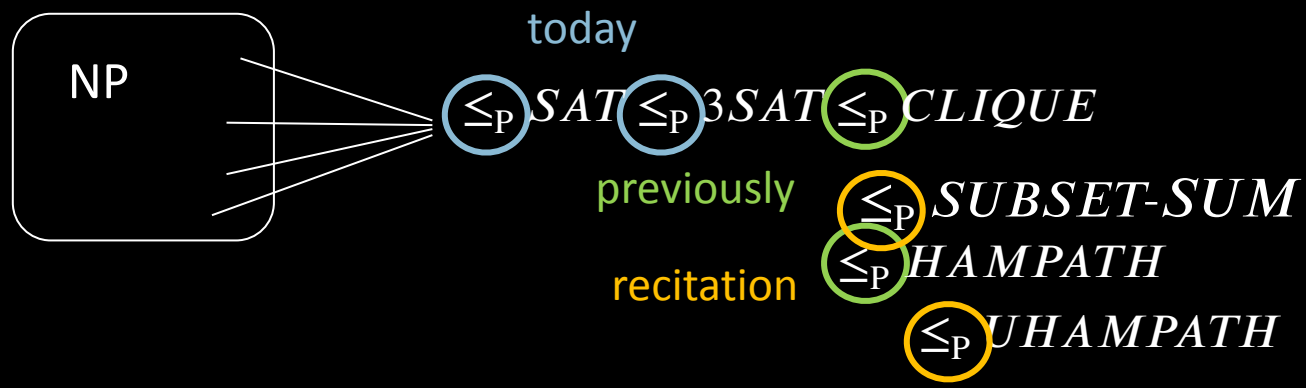
or some other previously shown NP-complete language

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

If B is NP-complete and $B \in$ P then P = NP.



Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete,
show $3SAT \leq_P C$.

or some other previously shown NP-complete language

Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

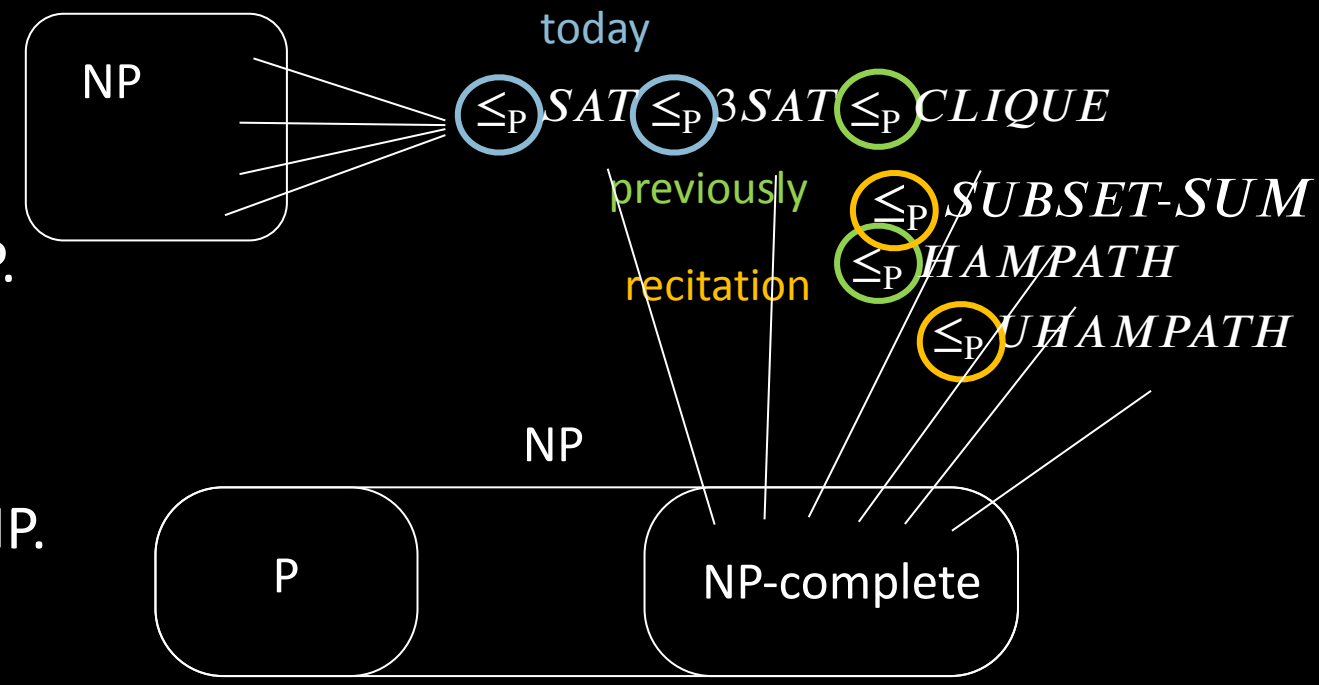
If B is NP-complete and $B \in$ P then P = NP.

Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete, show $3SAT \leq_P C$.

$3SAT$: or some other previously shown NP-complete language



Quick Review

Defn: B is NP-complete if

1) $B \in$ NP
2) For all $A \in$ NP, $A \leq_P B$

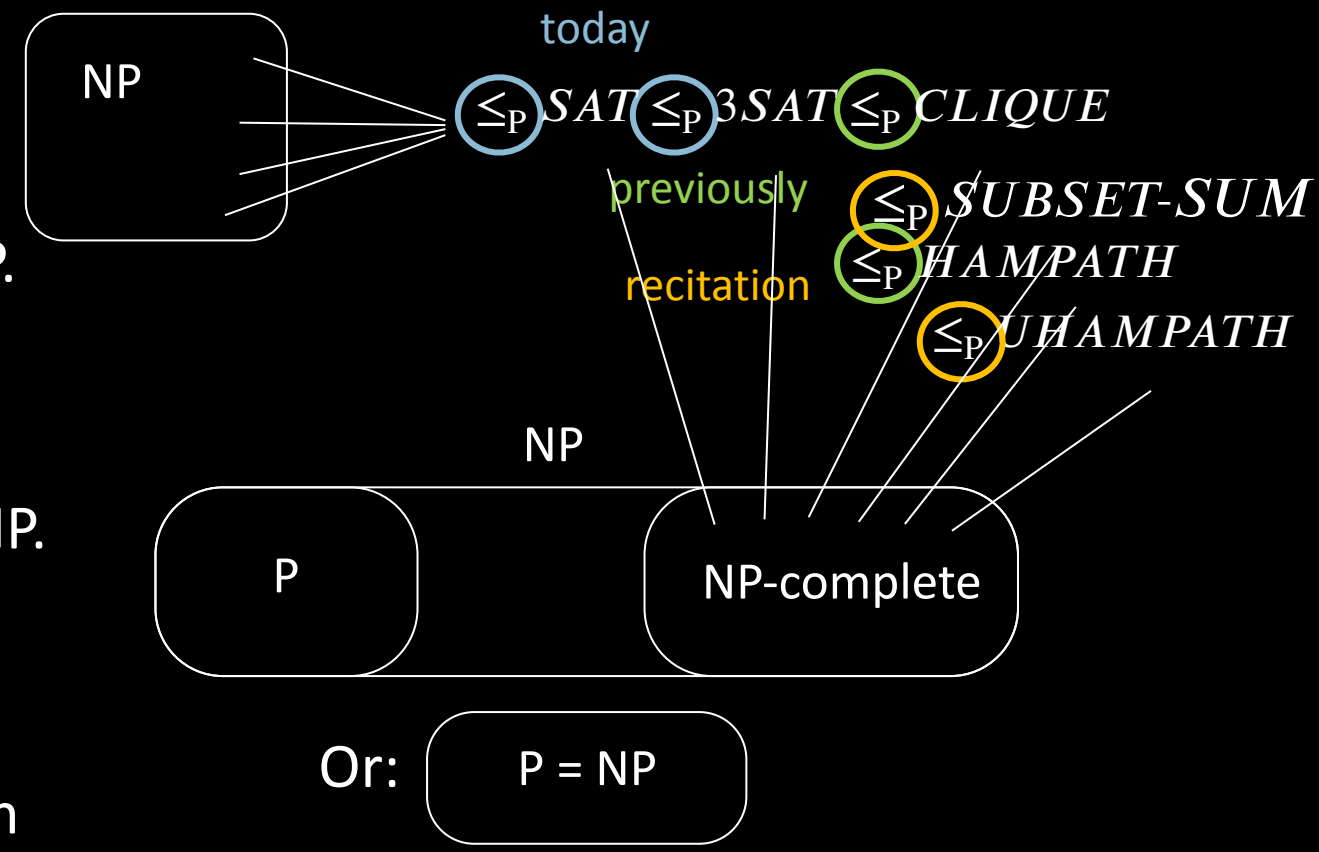
If B is NP-complete and $B \in$ P then P = NP.

Importance of NP-completeness

1) Evidence of computational intractability.
2) Gives a good candidate for proving P ≠ NP.

To show some language C is NP-complete, show $3SAT \leq_P C$.

$3SAT$: or some other previously shown NP-complete language



Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

f: $\Sigma^* \rightarrow$ formulas

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable

Idea: $\phi_{M,w}$ simulates M on w. Design $\phi_{M,w}$ to "say" M accepts w.

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable

Idea: $\phi_{M,w}$ simulates M on w. Design $\phi_{M,w}$ to "say" M accepts w.

Satisfying assignment to $\phi_{M,w}$ is a computation history for M on w.

Cook-Levin Theorem (idea)

Theorem: SAT is NP-complete

Proof: 1) $SAT \in NP$ (done)

2) Show that for each $A \in NP$ we have $A \leq_P SAT$:

Let $A \in NP$ be decided by NTM M in time n^k.

Give a polynomial-time reduction f mapping A to SAT.

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable

Idea: $\phi_{M,w}$ simulates M on w. Design $\phi_{M,w}$ to "say" M accepts w.

Satisfying assignment to $\phi_{M,w}$ is a computation history for M on w.

Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.

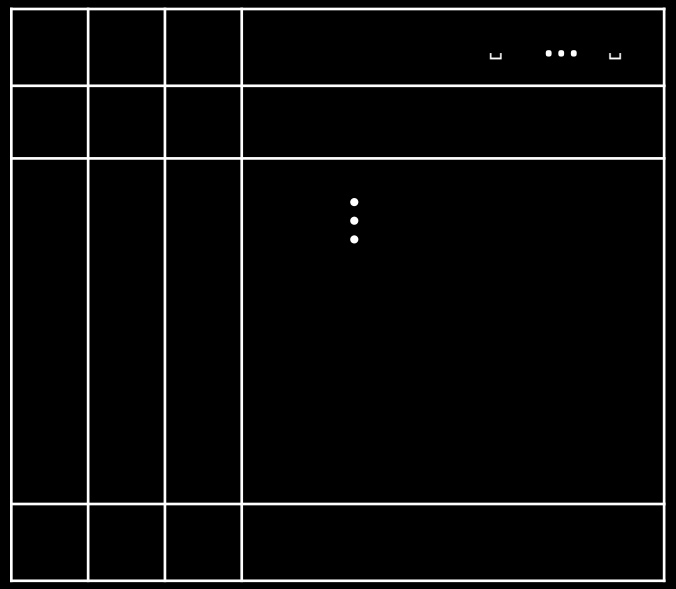


Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.

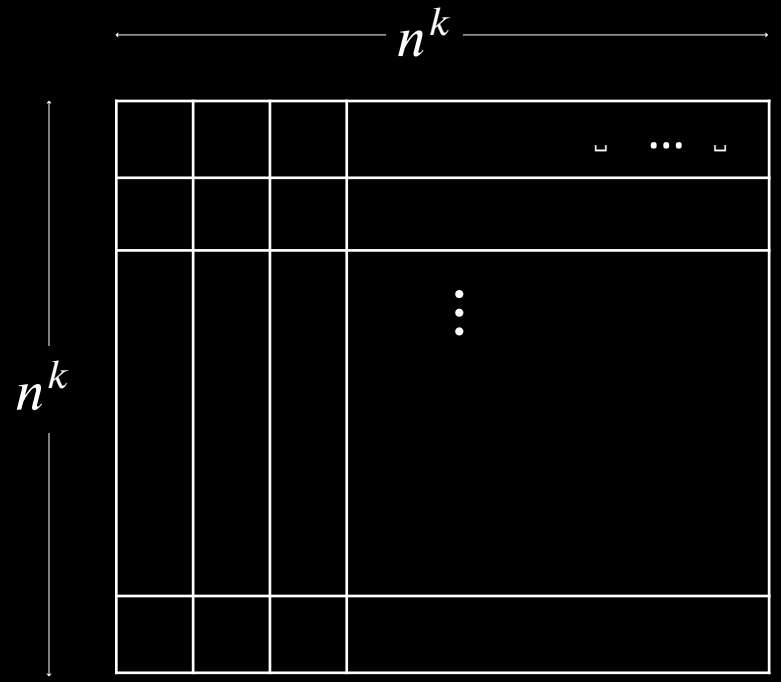


Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.

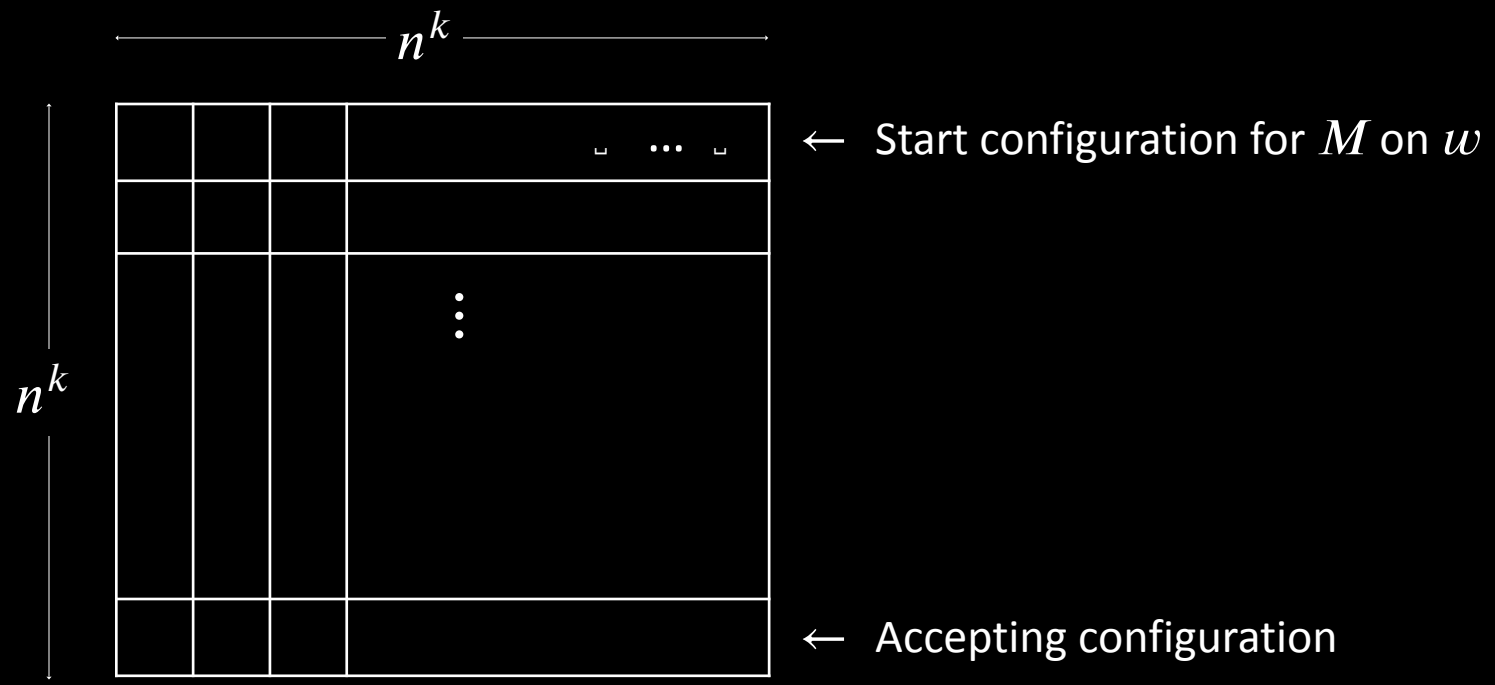


Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.

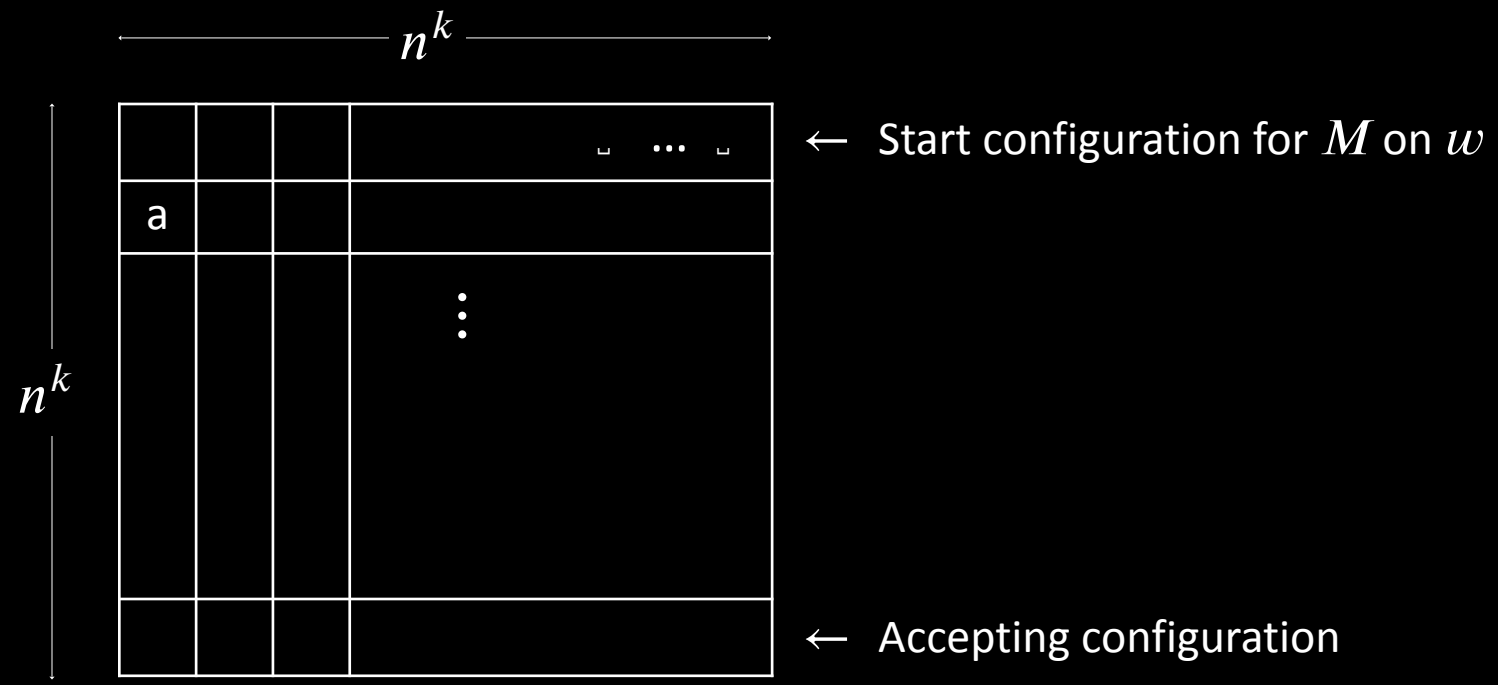
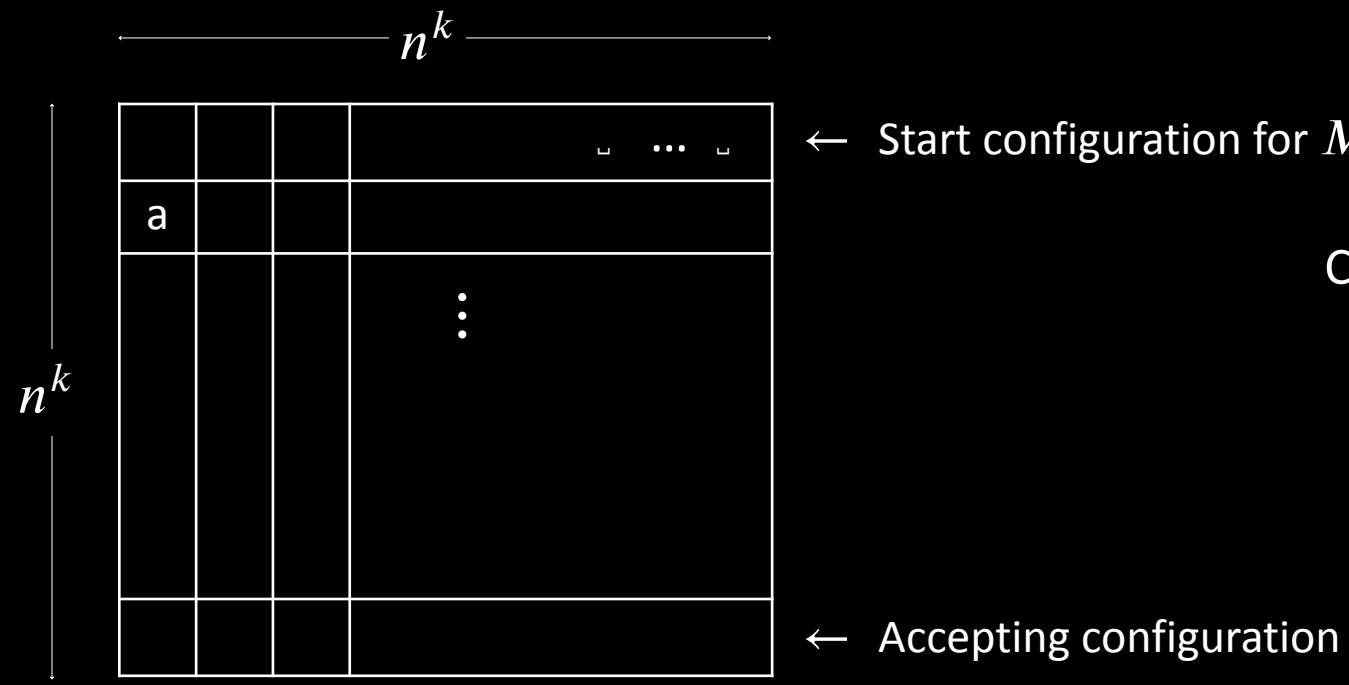


Tableau for M on w

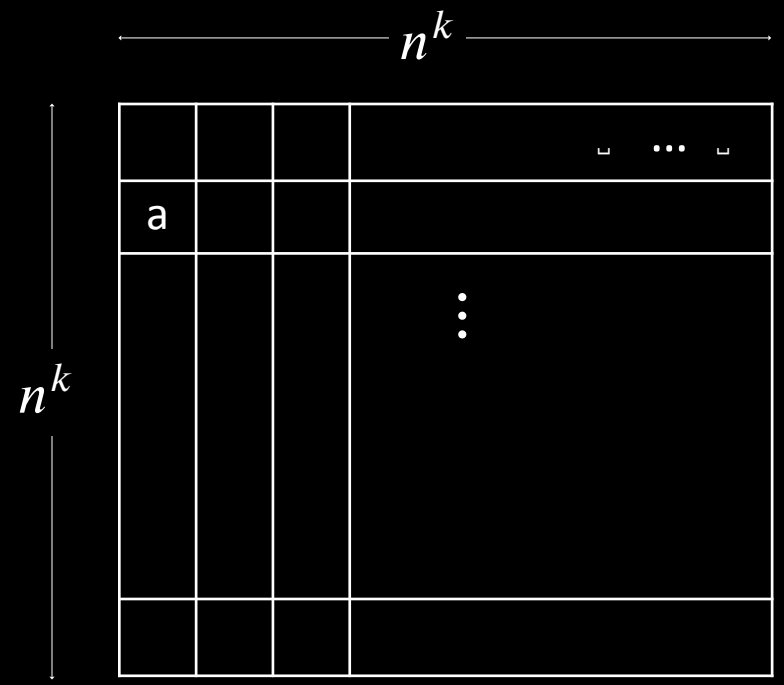
Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.



Construct $\phi_{M,w}$ to "say" M accepts w.

Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.



← Start configuration for M on w

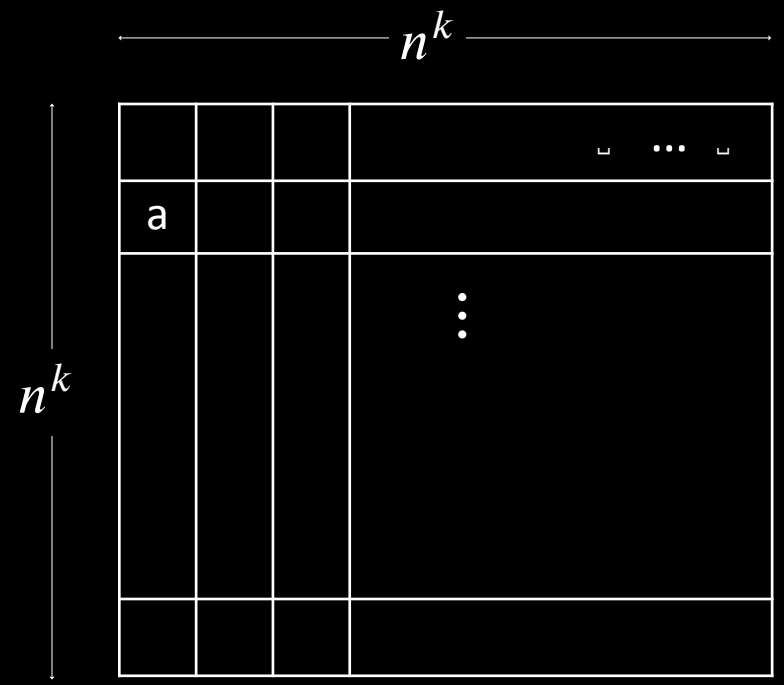
← Accepting configuration

Construct $\phi_{M,w}$ to "say" M accepts w.

$\phi_{M,w}$ "says" a tableau for M on w exists.

Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.



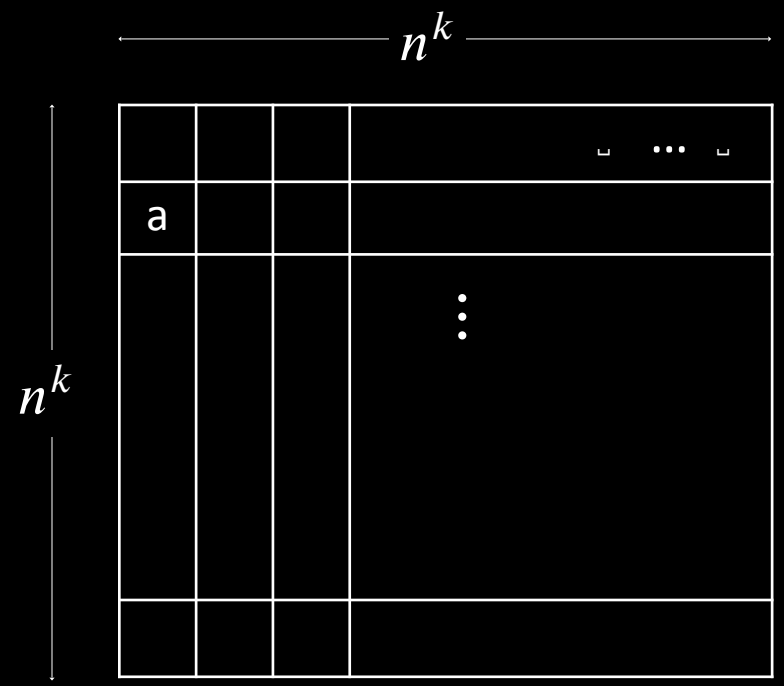
Construct $\phi_{M,w}$ to "say" M accepts w.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

Tableau for M on w

Defn: An (accepting) tableau for NTM M on w is an $n^k \times n^k$ table representing an computation history for M on w on an accepting branch of the nondeterministic computation.



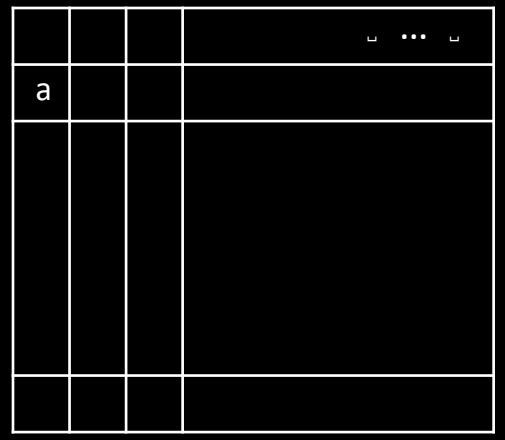
Construct $\phi_{M,w}$ to "say" M accepts w.

$\phi_{M,w}$ "says" a tableau for M on w exists.

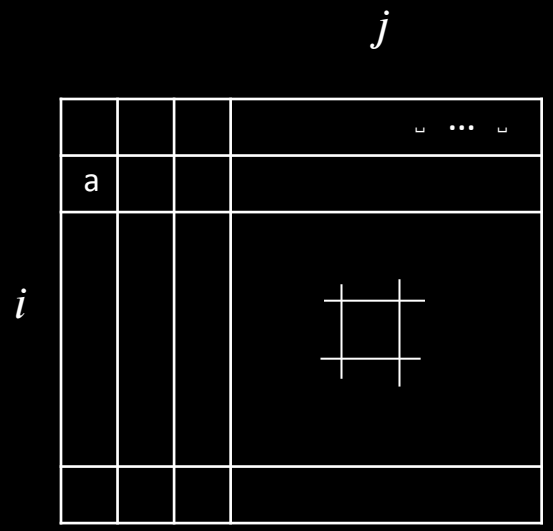
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

Constructing $\phi_{M,w}$: ϕcell

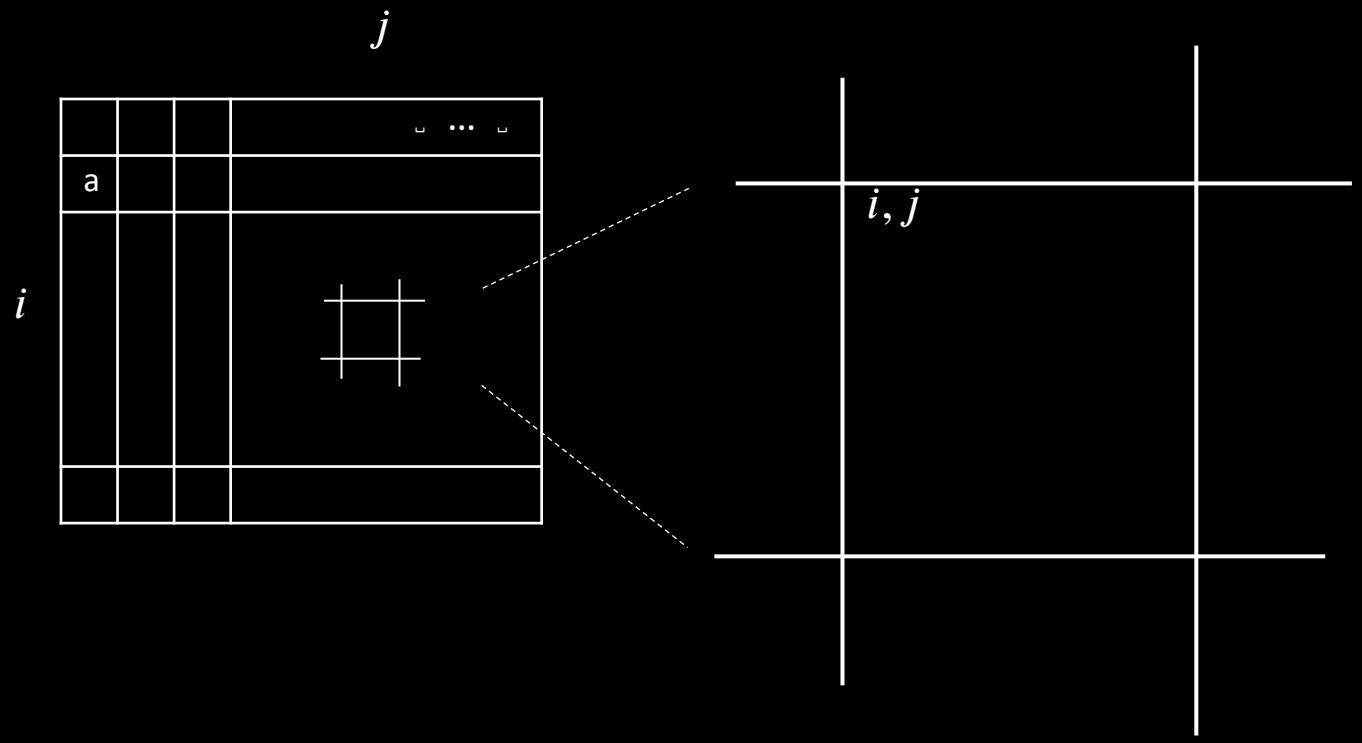
Constructing $\phi_{M,w}$: ϕcell



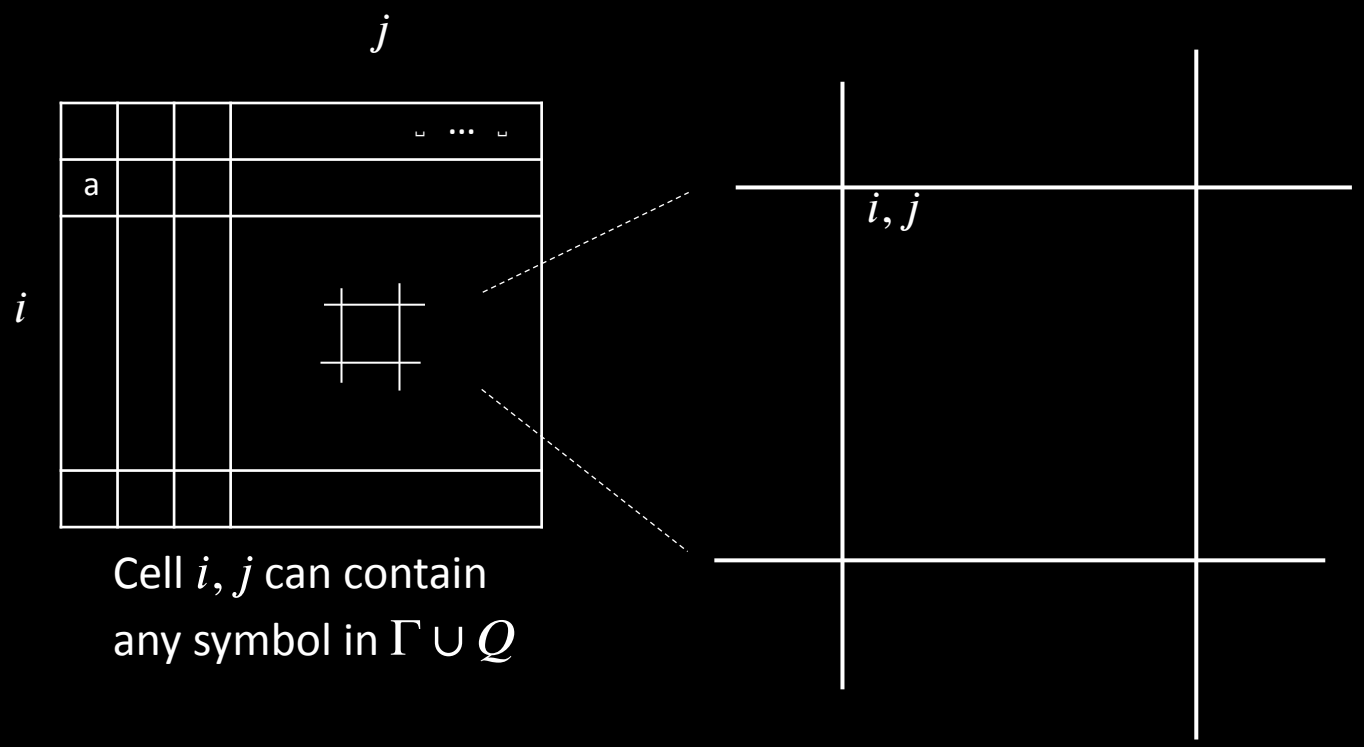
Constructing $\phi_{M,w}$: ϕcell



Constructing $\phi_{M,w}$: ϕcell

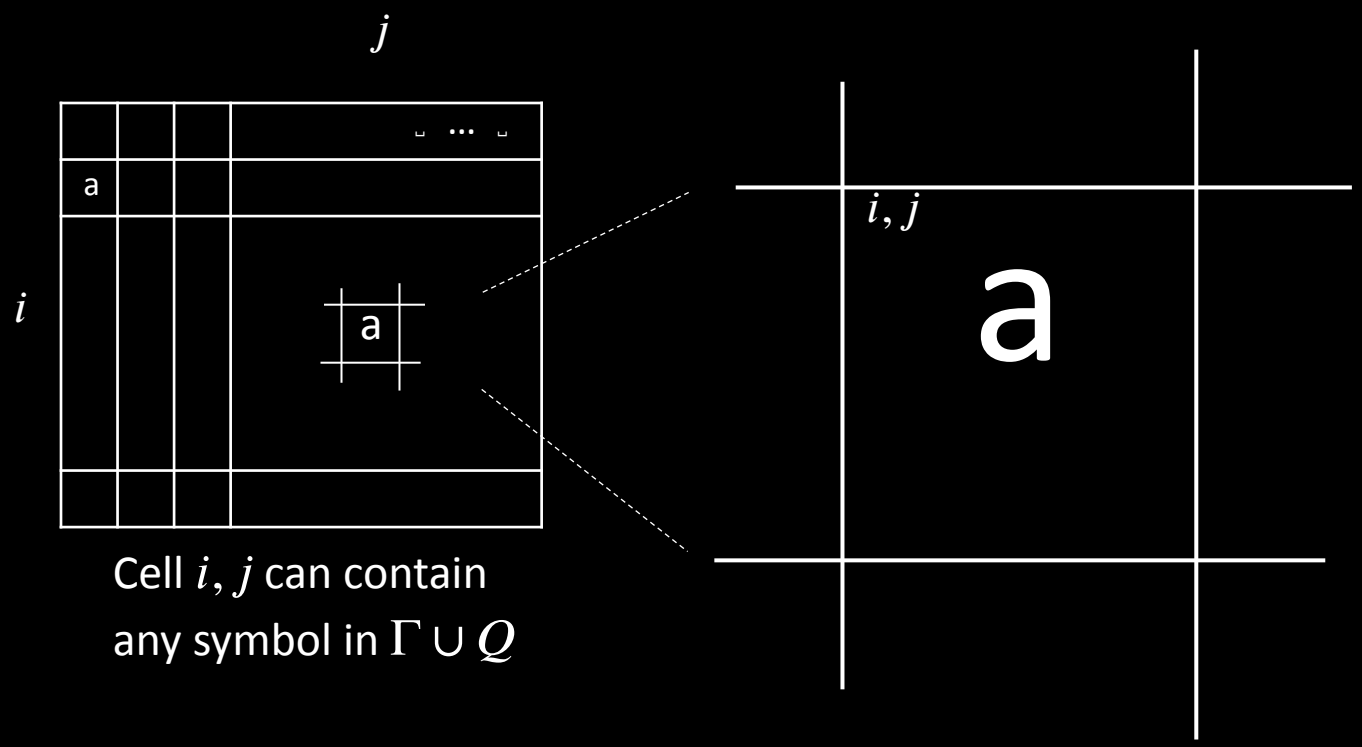


Constructing $\phi_{M,w}$: ϕcell



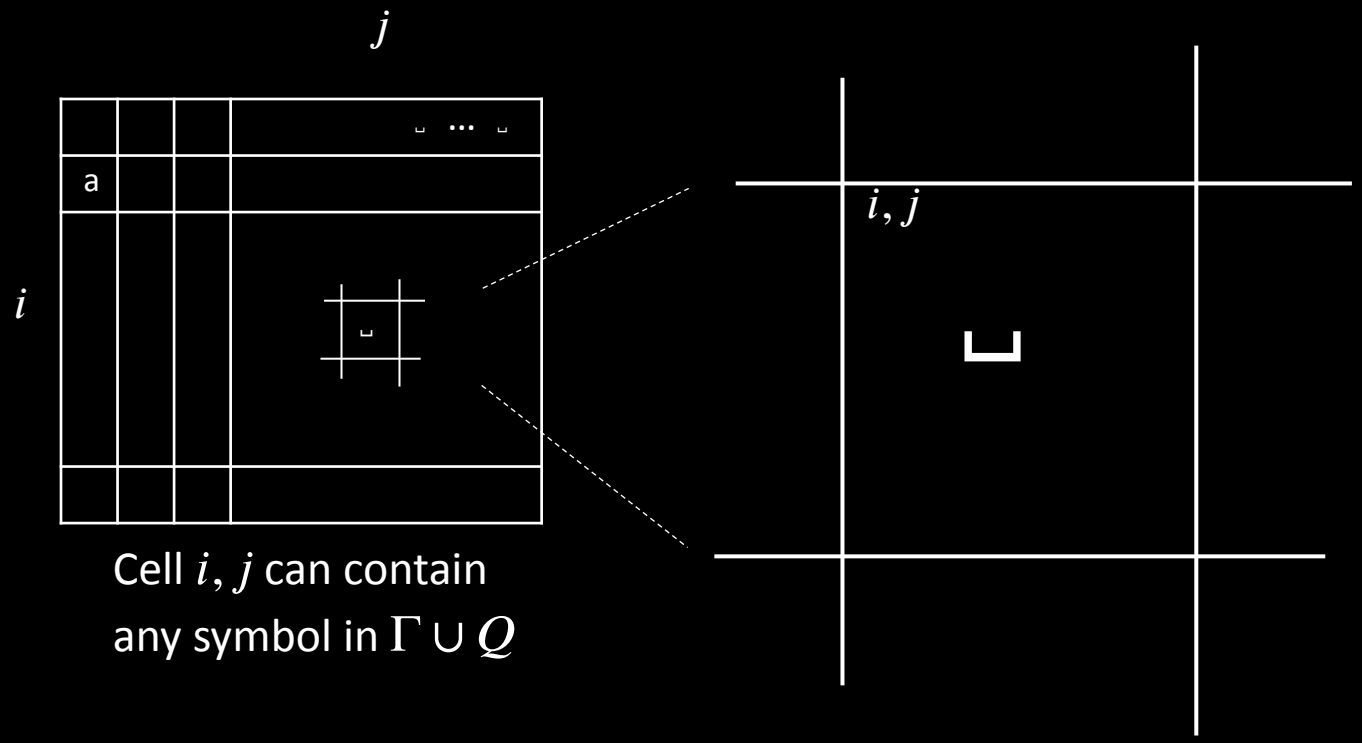
Cell i, j can contain any symbol in $\Gamma \cup Q$

Constructing $\phi_{M,w}$: ϕcell



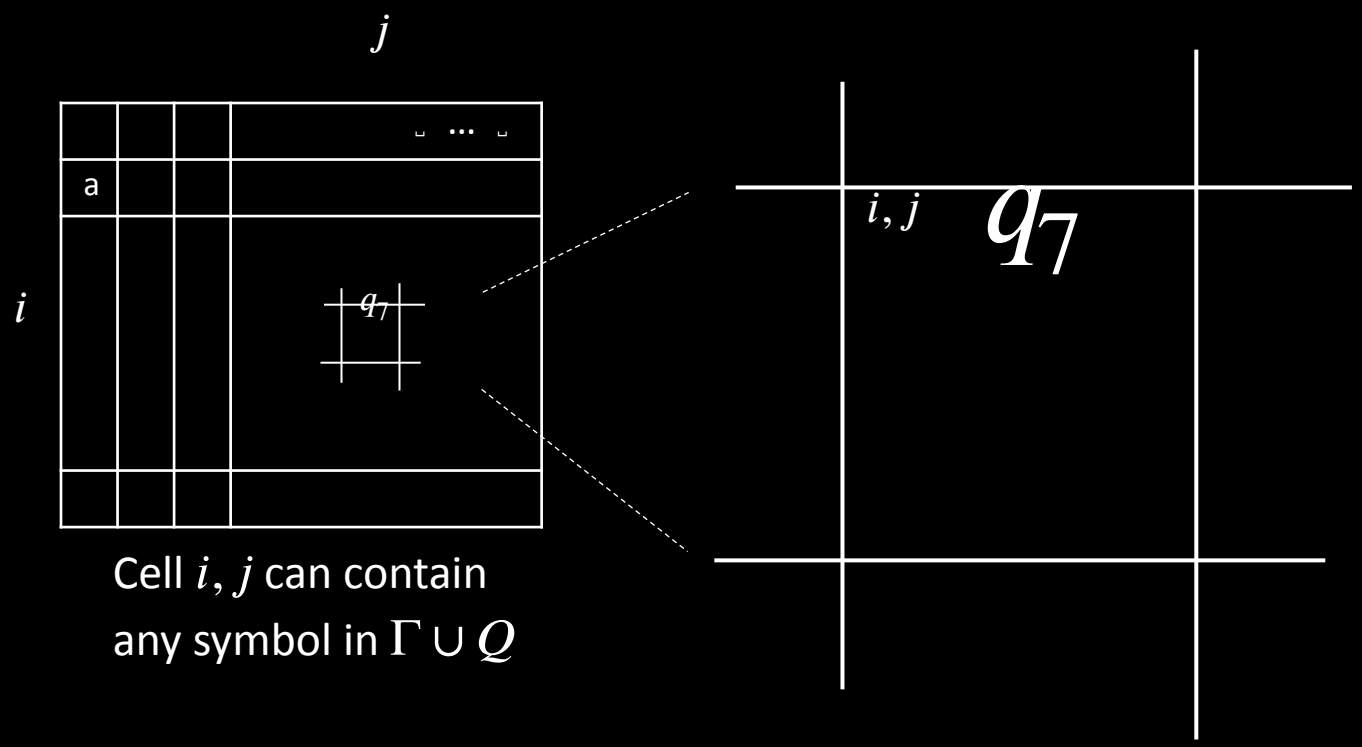
Cell i, j can contain any symbol in $\Gamma \cup Q$

Constructing $\phi_{M,w}$: ϕcell



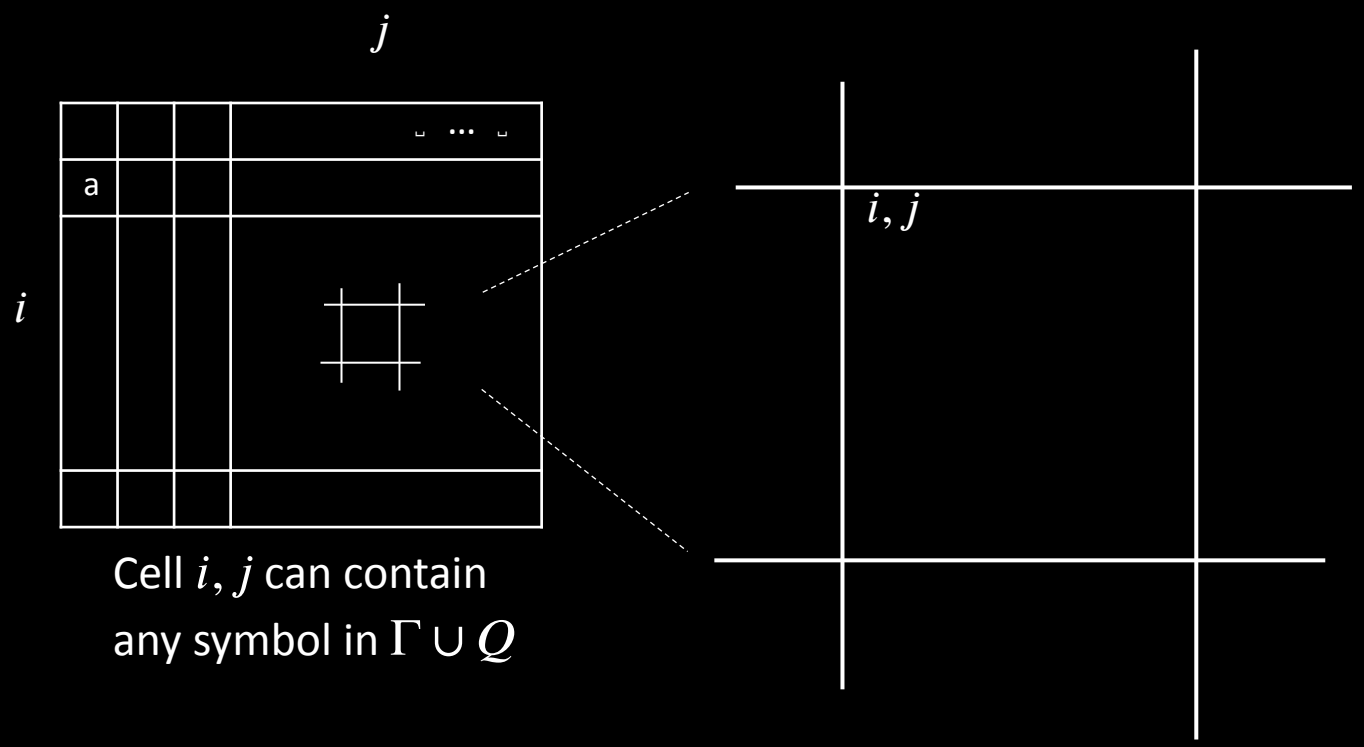
Cell i, j can contain any symbol in $\Gamma \cup Q$

Constructing $\phi_{M,w}$: ϕcell



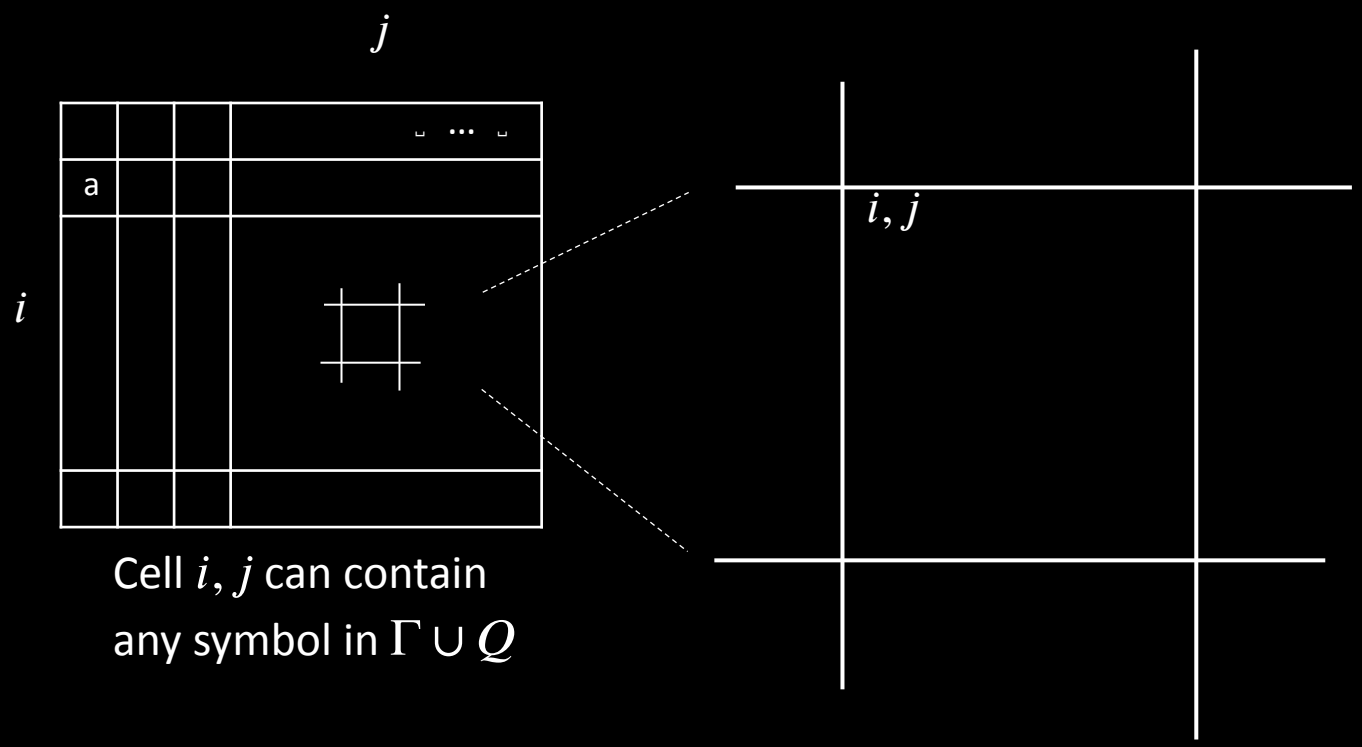
Cell i, j can contain any symbol in $\Gamma \cup Q$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

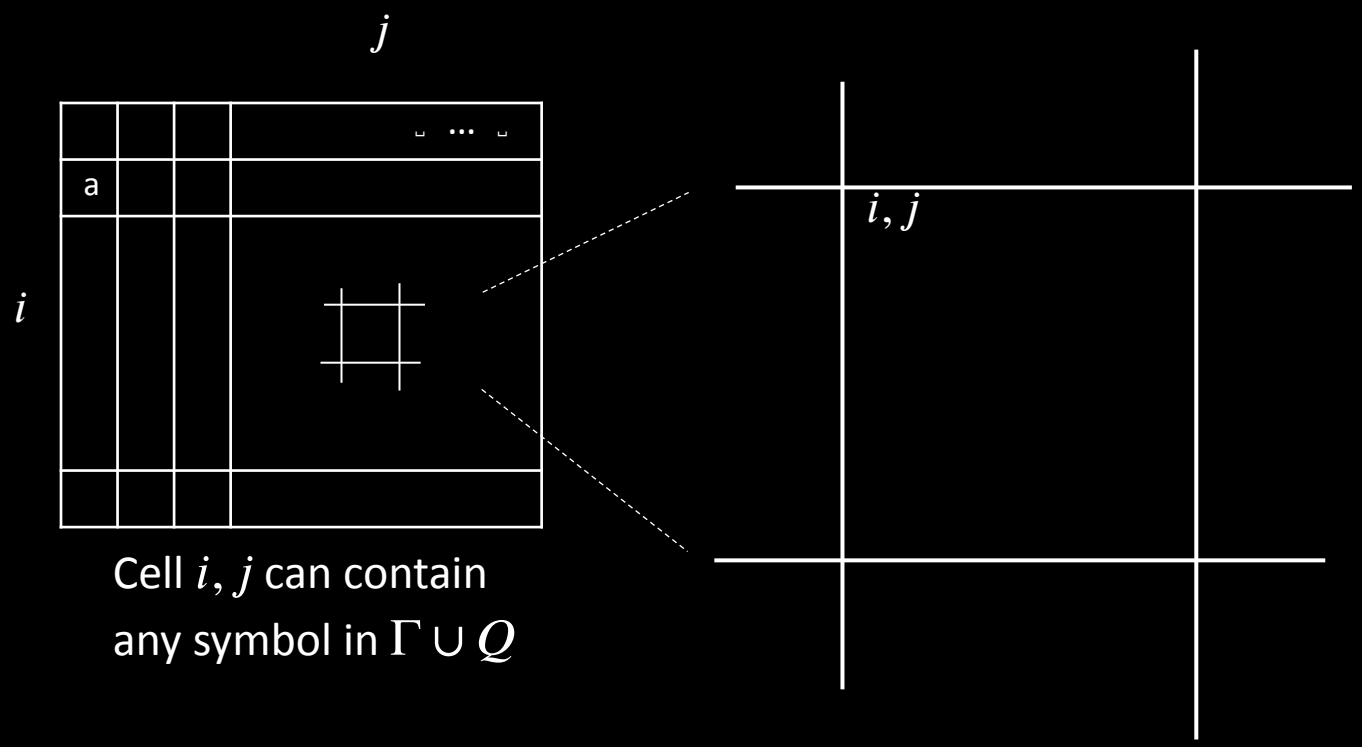
Constructing $\phi_{M,w}$: ϕcell



The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

Constructing $\phi_{M,w}$: ϕcell

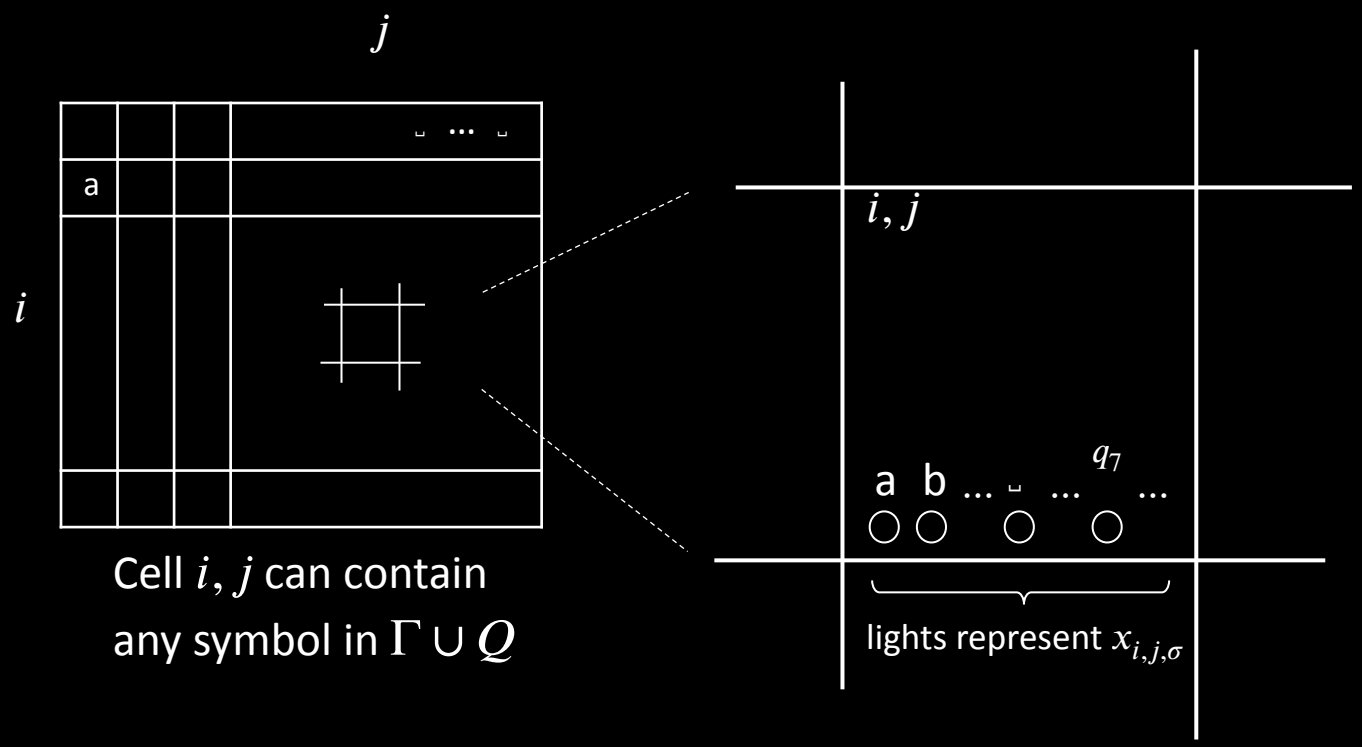


The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell



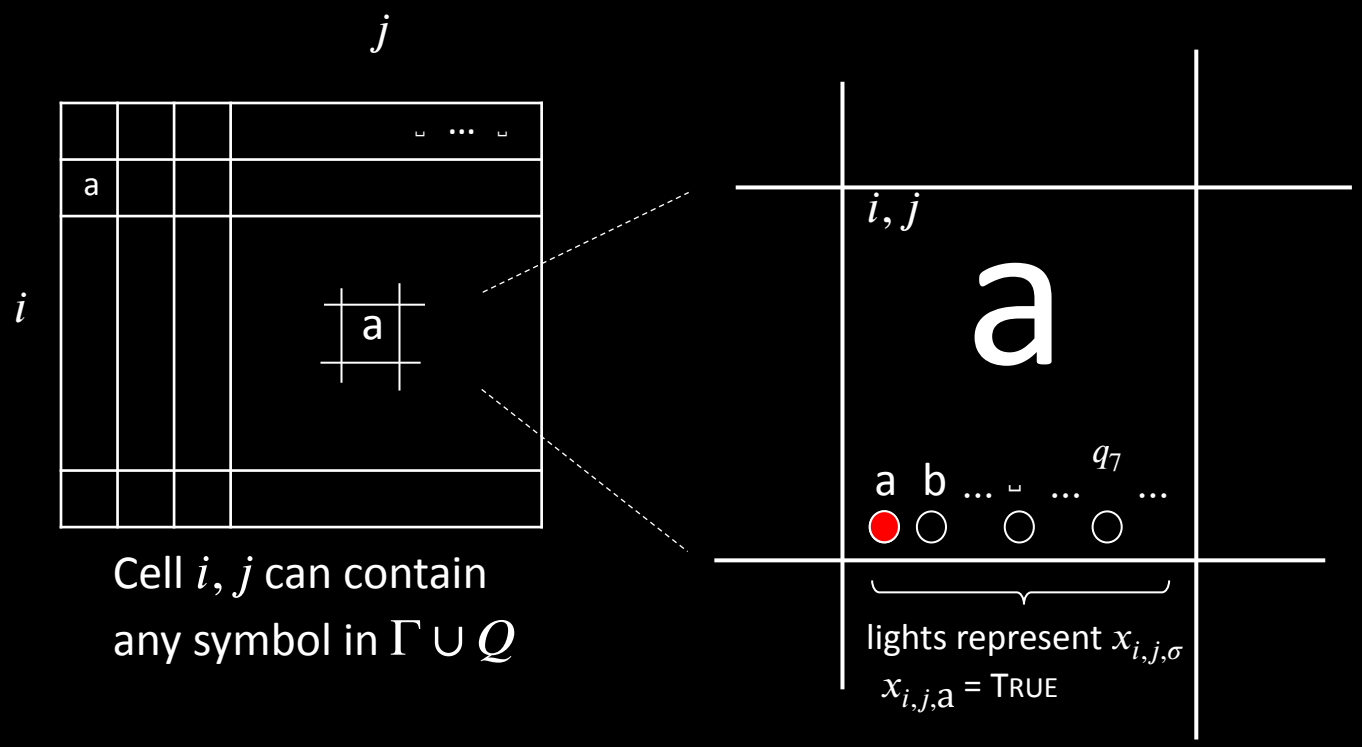
Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell

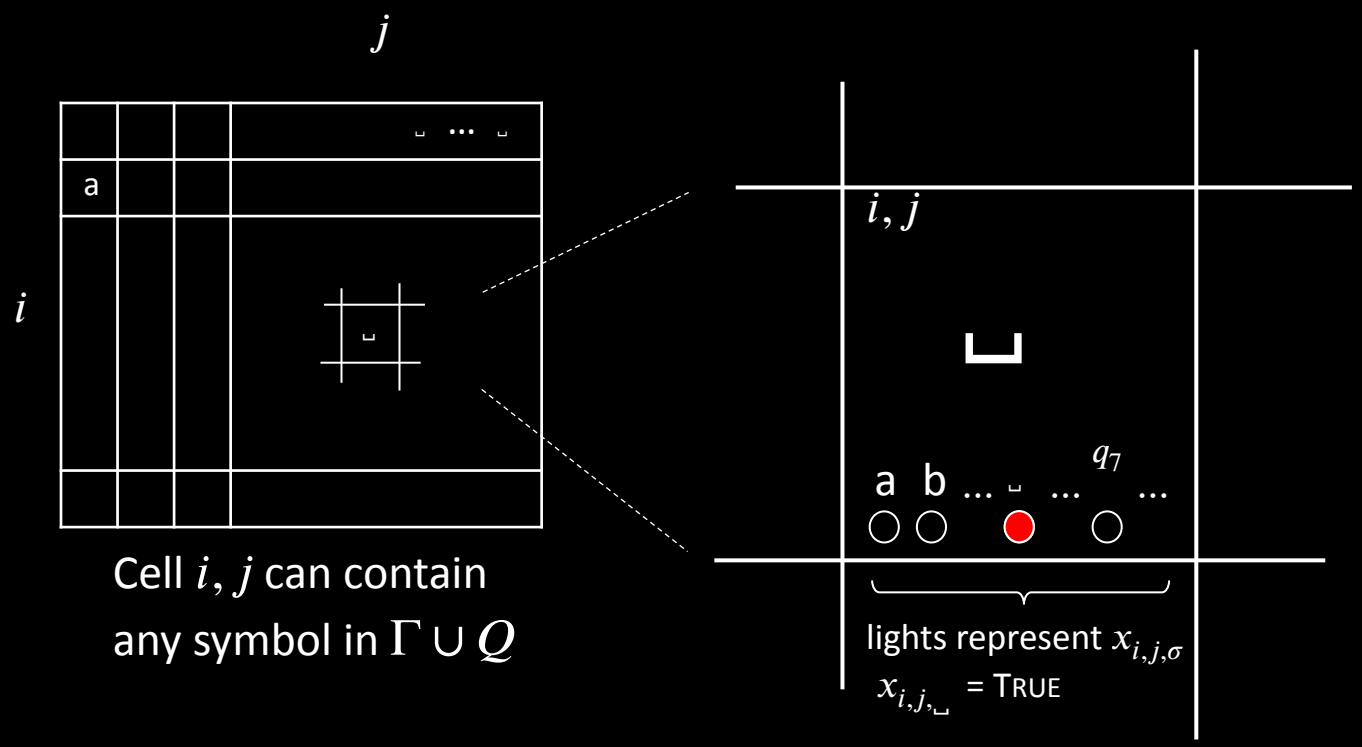


The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell

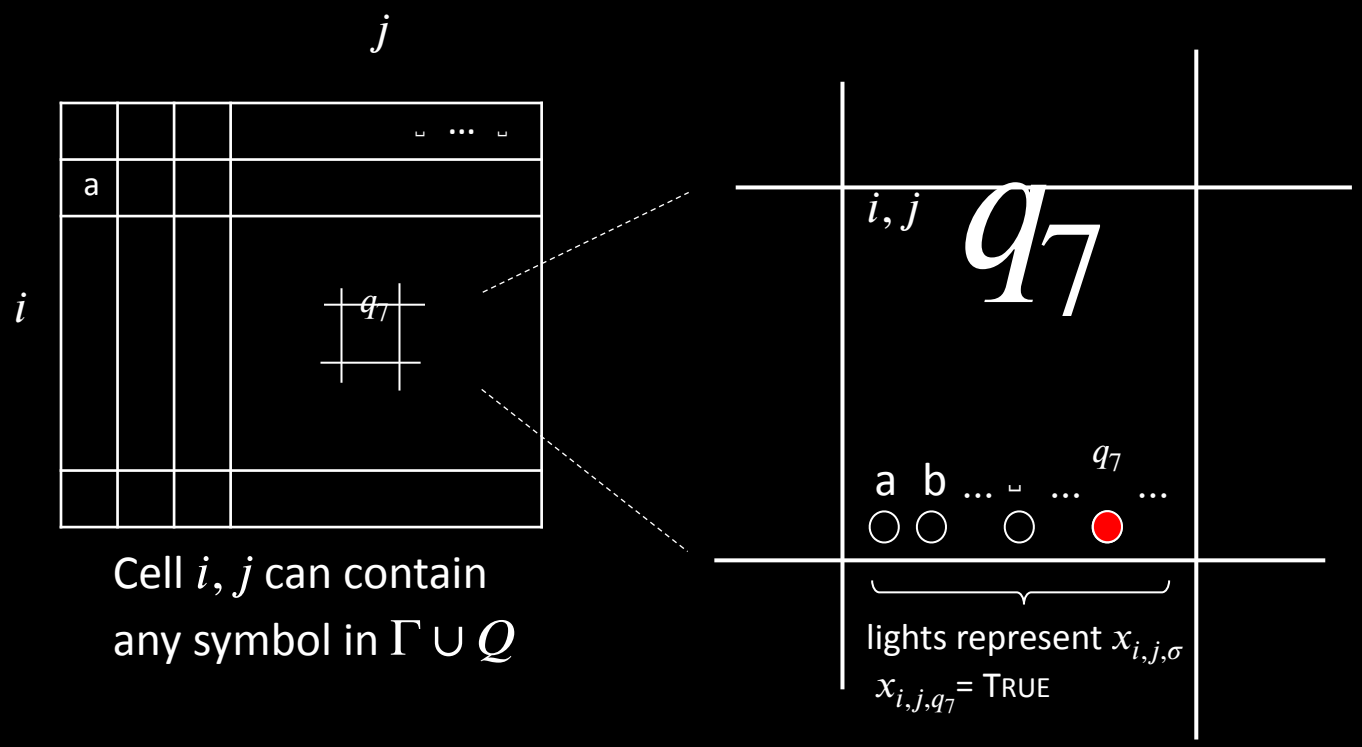


Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell

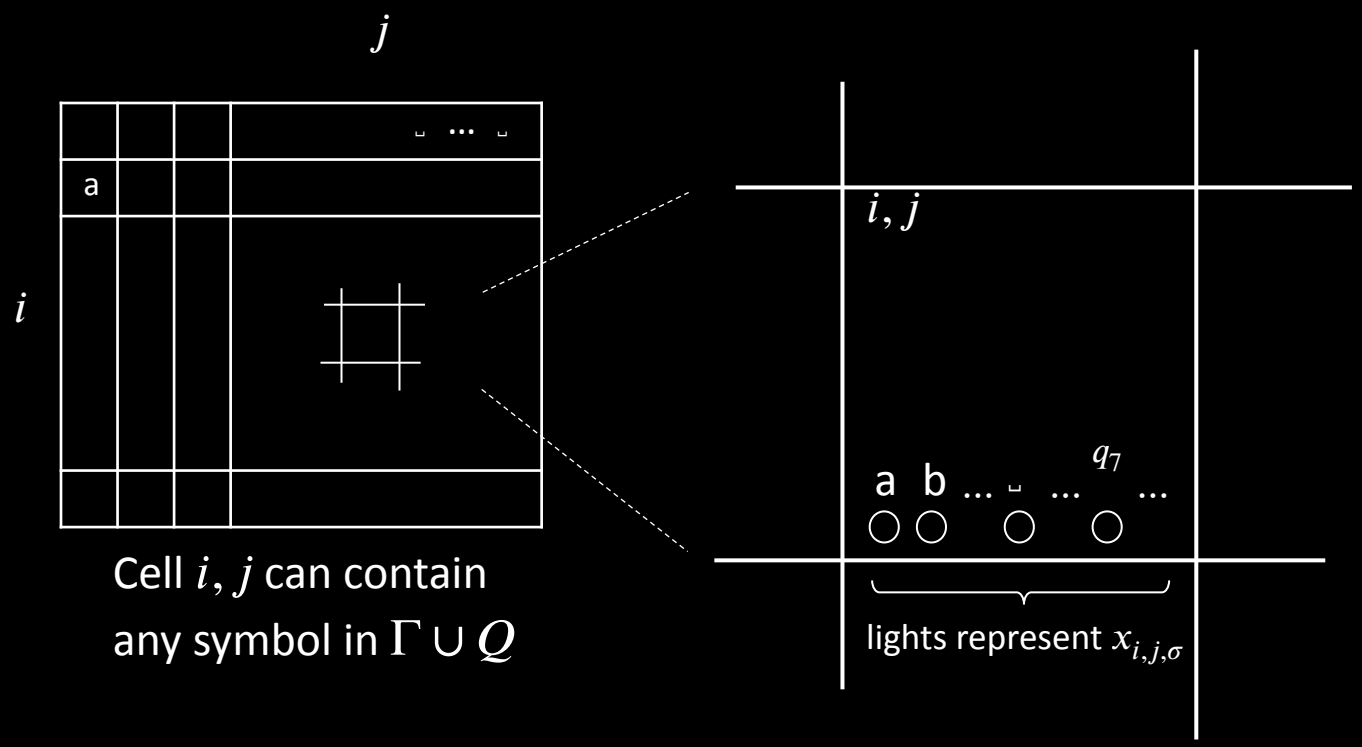


The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell



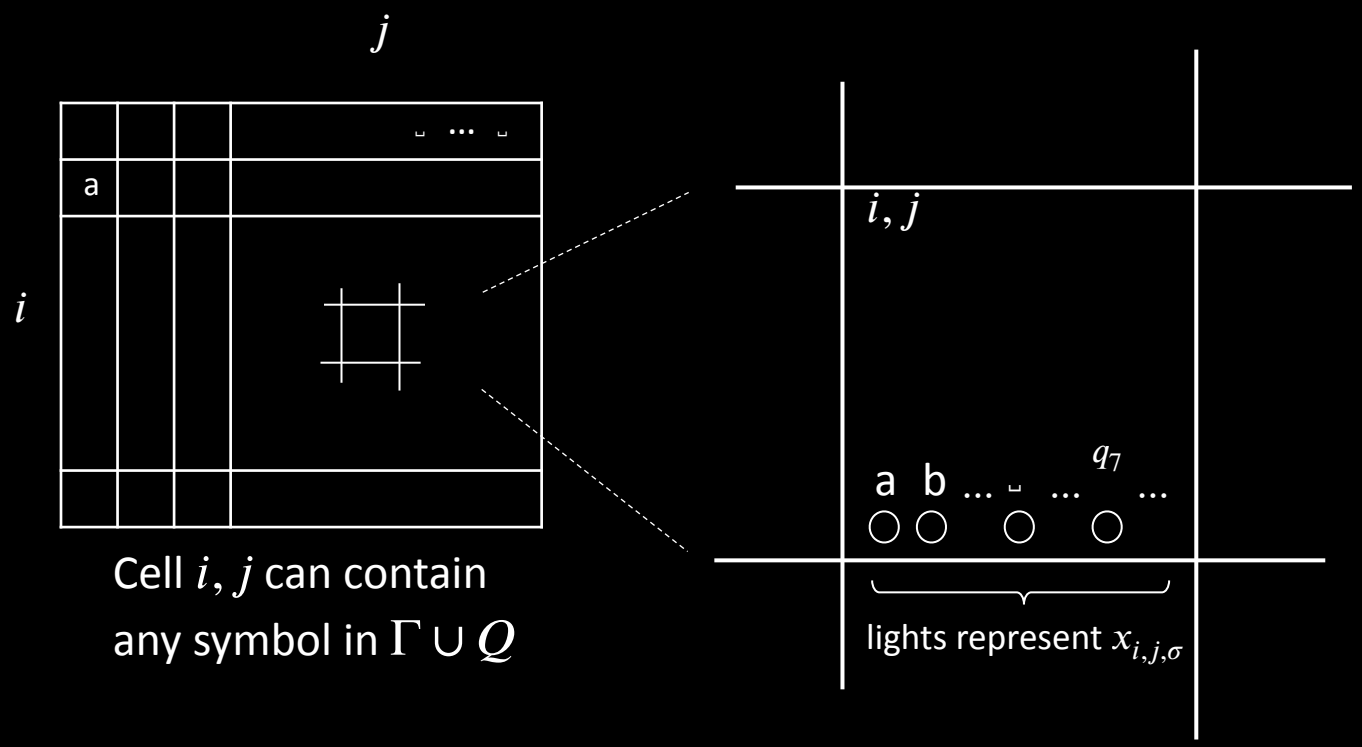
Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

$\phi_{M,w}$ "says" a tableau for M on w exists.

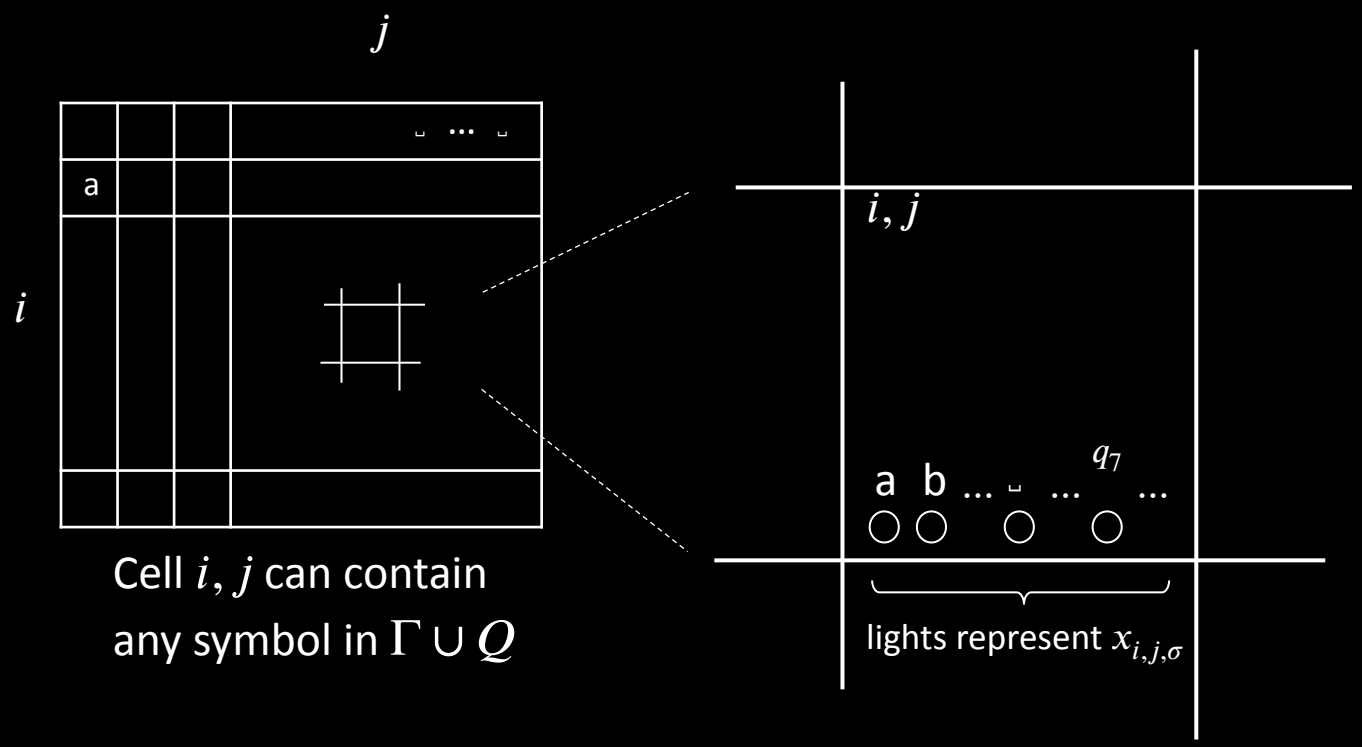
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

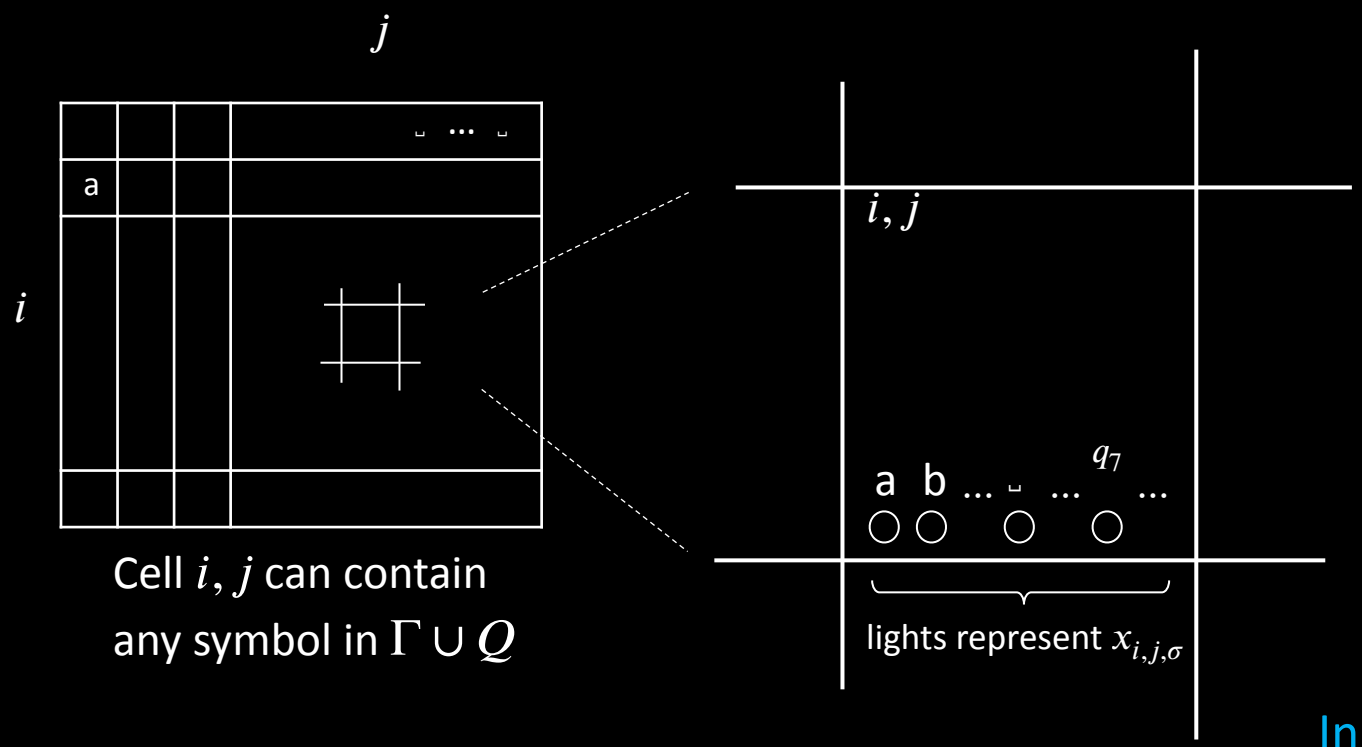
$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma} =$ TRUE for each i, j .

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

ϕcell "says" exactly one light is on per cell

i.e., exactly one $x_{i,j,\sigma} =$ TRUE for each i, j .

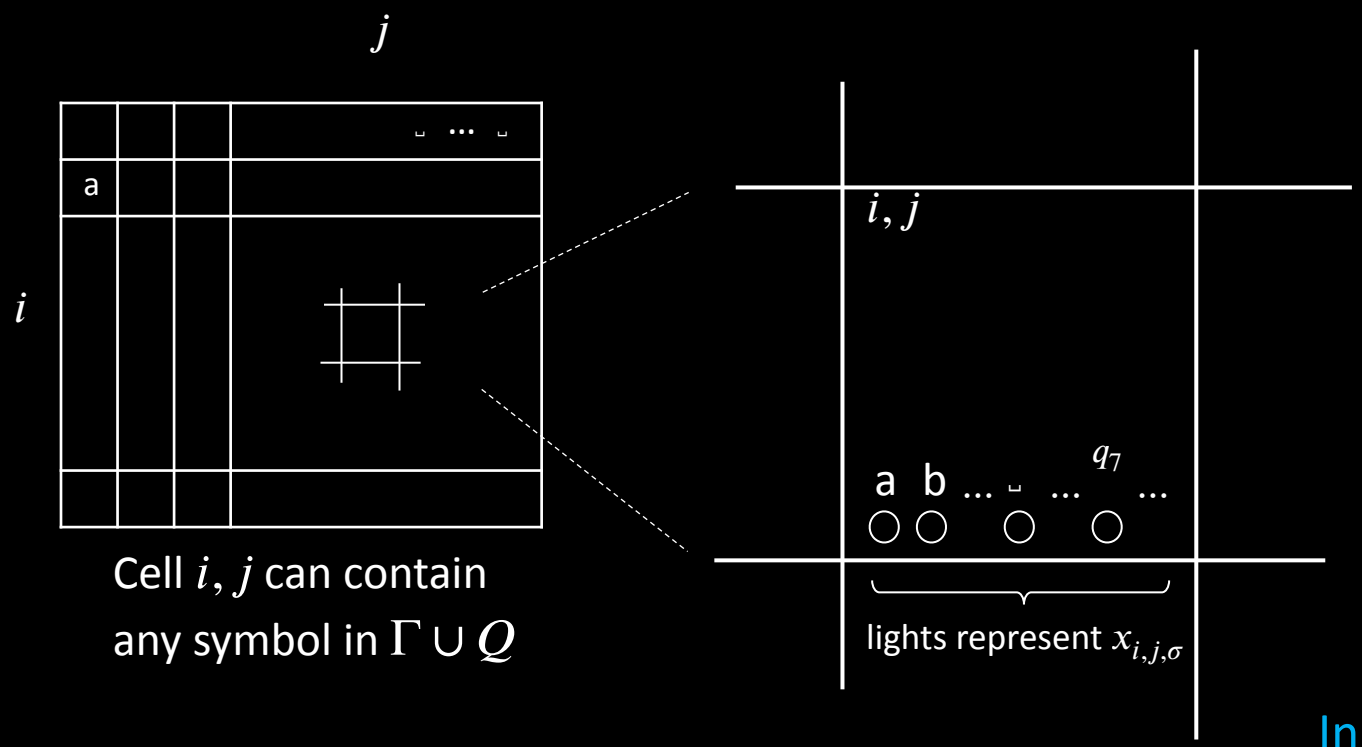
In every cell at least one light and at most one light

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$

for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

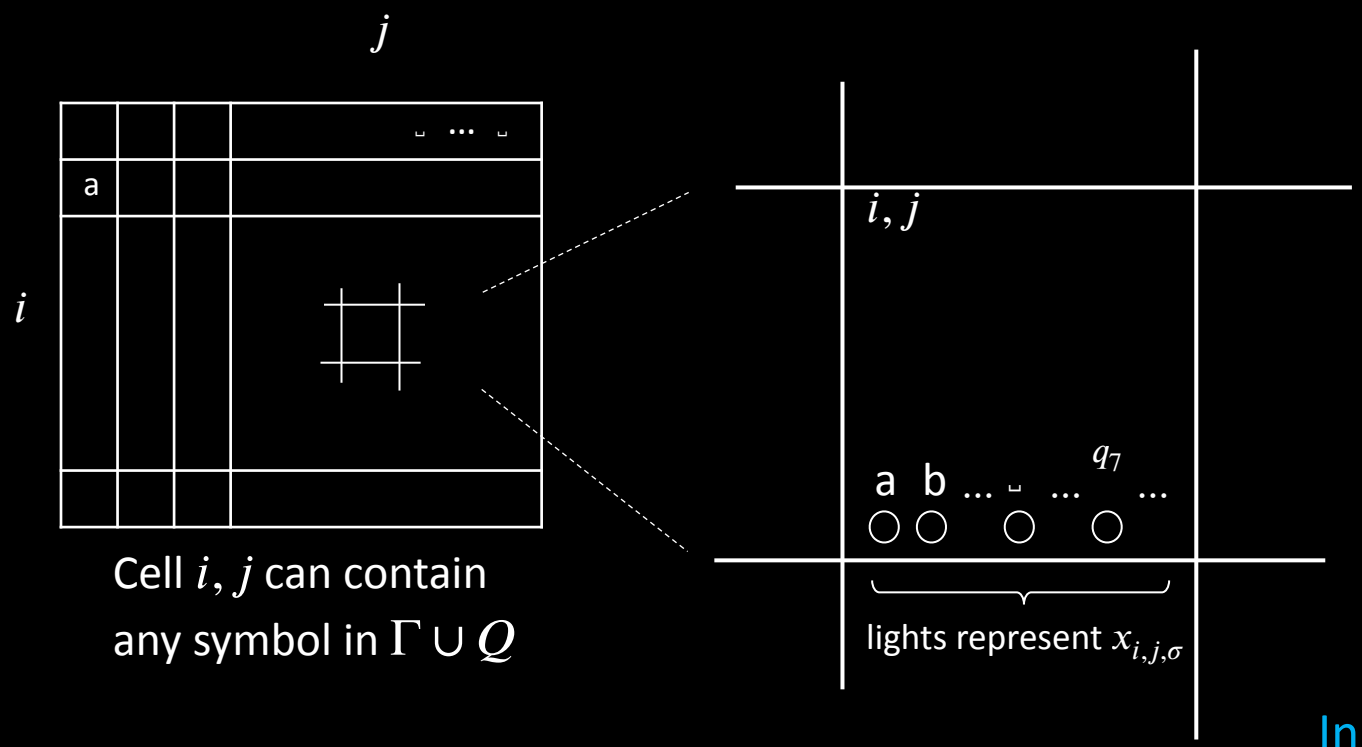
ϕcell "says" exactly one light is on per cell

i.e., exactly one $x_{i,j,\sigma} =$ TRUE for each i, j .

In every cell at least one light and at most one light

$x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

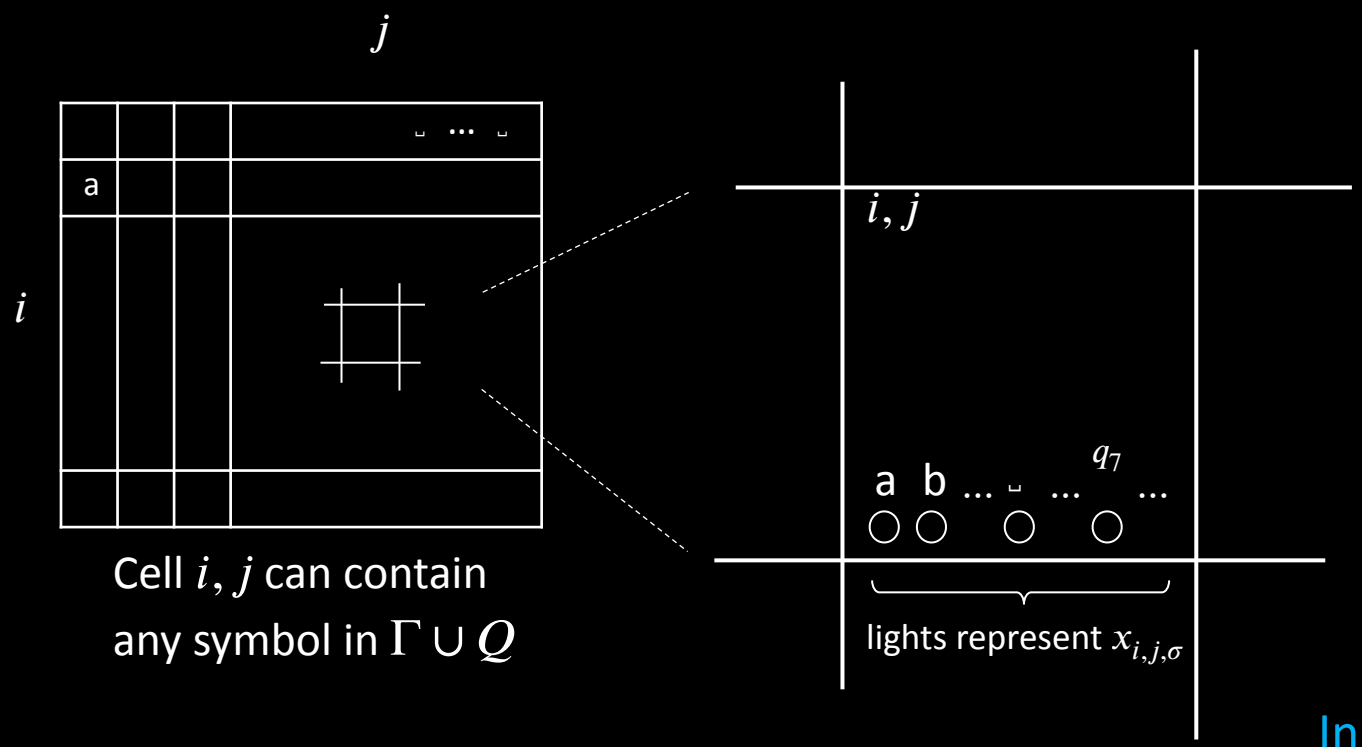
ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma} =$ TRUE for each i, j .

In every cell at least one light and at most one light

$$x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}$$

$$\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma}$$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma}$ = TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

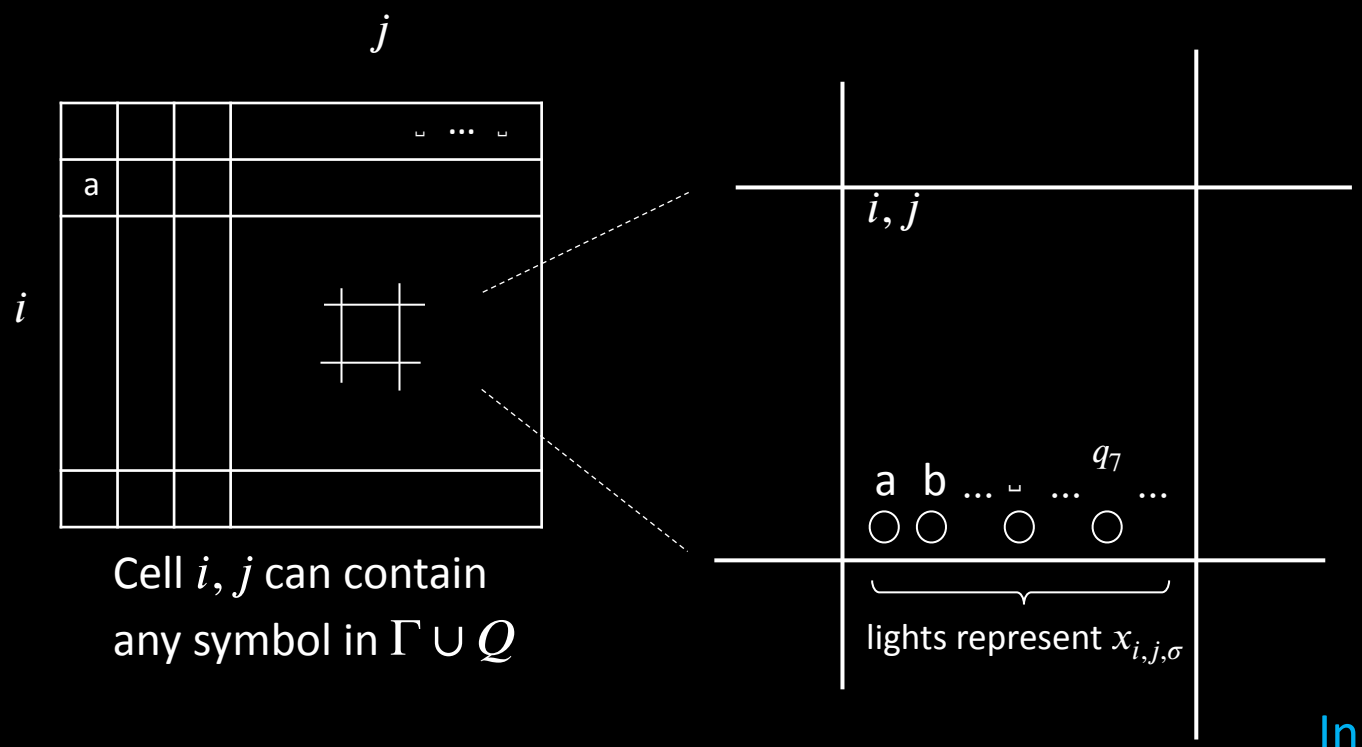
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma}$ = TRUE for each i, j .

In every cell at least one light and at most one light

$$\underbrace{x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}}_{\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma}} \qquad \bigwedge_{\substack{\sigma,\tau \in \Gamma \cup Q \\ \sigma \neq \tau}} \left(\overline{x_{i,j,\sigma} \wedge x_{i,j,\tau}} \right)$$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

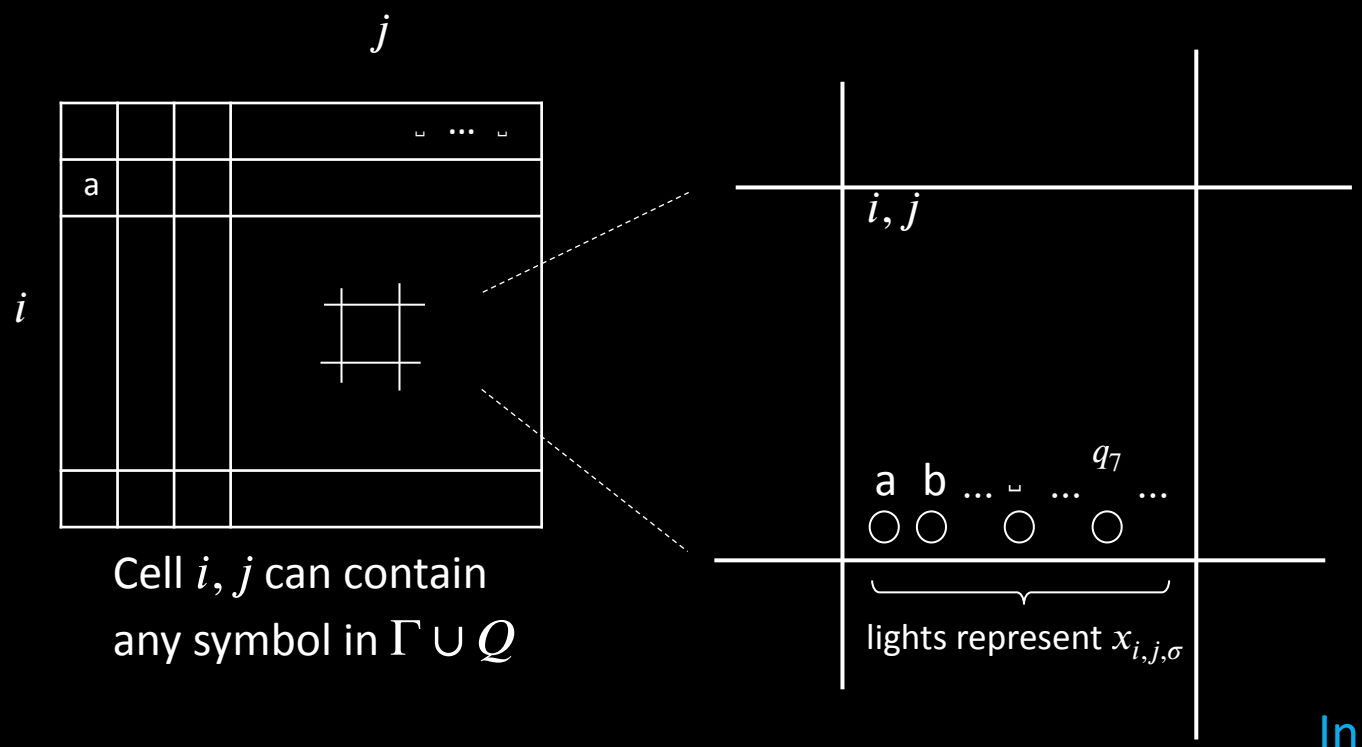
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma} =$ TRUE for each i, j .

In every cell at least one light and at most one light

$$\underbrace{x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}}_{\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma}} \wedge \bigwedge_{\substack{\sigma,\tau \in \Gamma \cup Q \\ \sigma \neq \tau}} \left(\overline{x_{i,j,\sigma} \wedge x_{i,j,\tau}} \right)$$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma}$ = TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

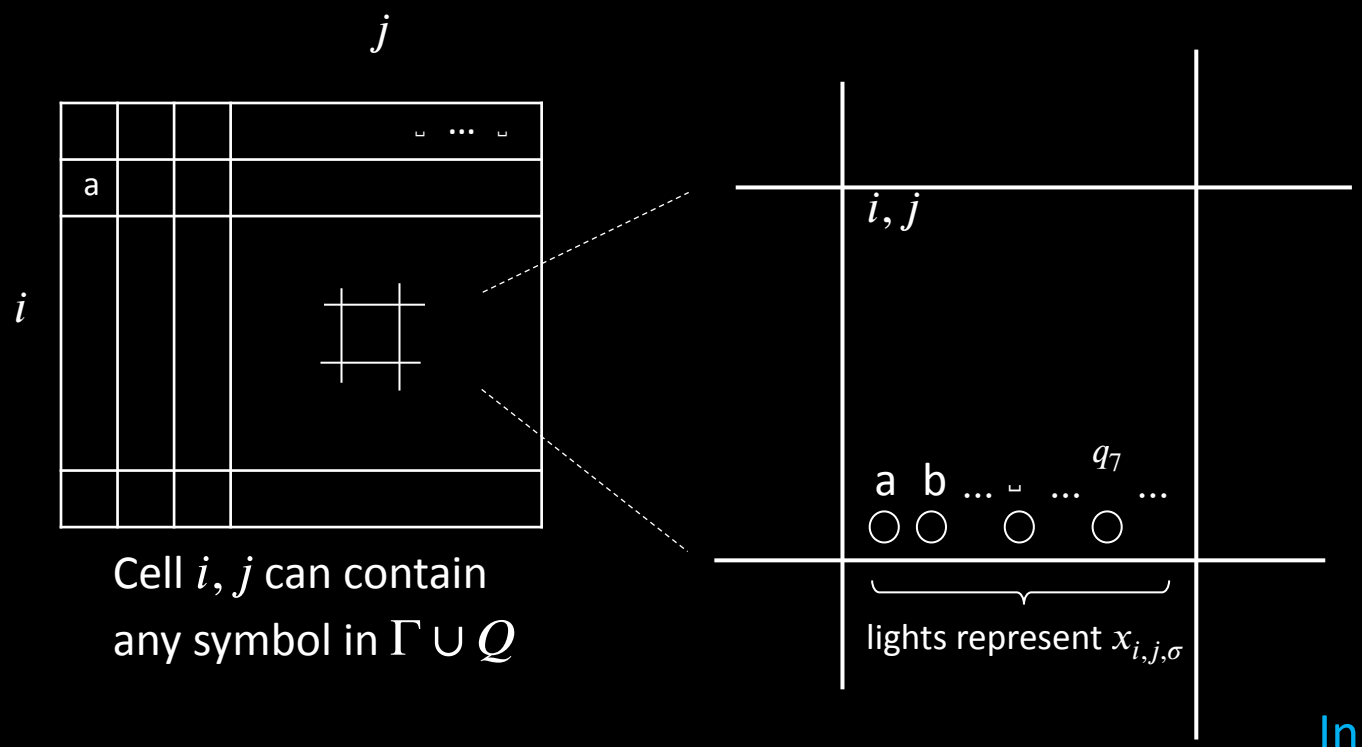
ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma}$ = TRUE for each i, j .

In every cell at least one light and at most one light

$$x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}$$

$$\bigwedge_{1 \le i,j \le n^k} \left(\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma} \wedge \bigwedge_{\substack{\sigma,\tau \in \Gamma \cup Q \\ \sigma \neq \tau}} \left(\overline{x_{i,j,\sigma} \wedge x_{i,j,\tau}} \right) \right)$$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma}$ = TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

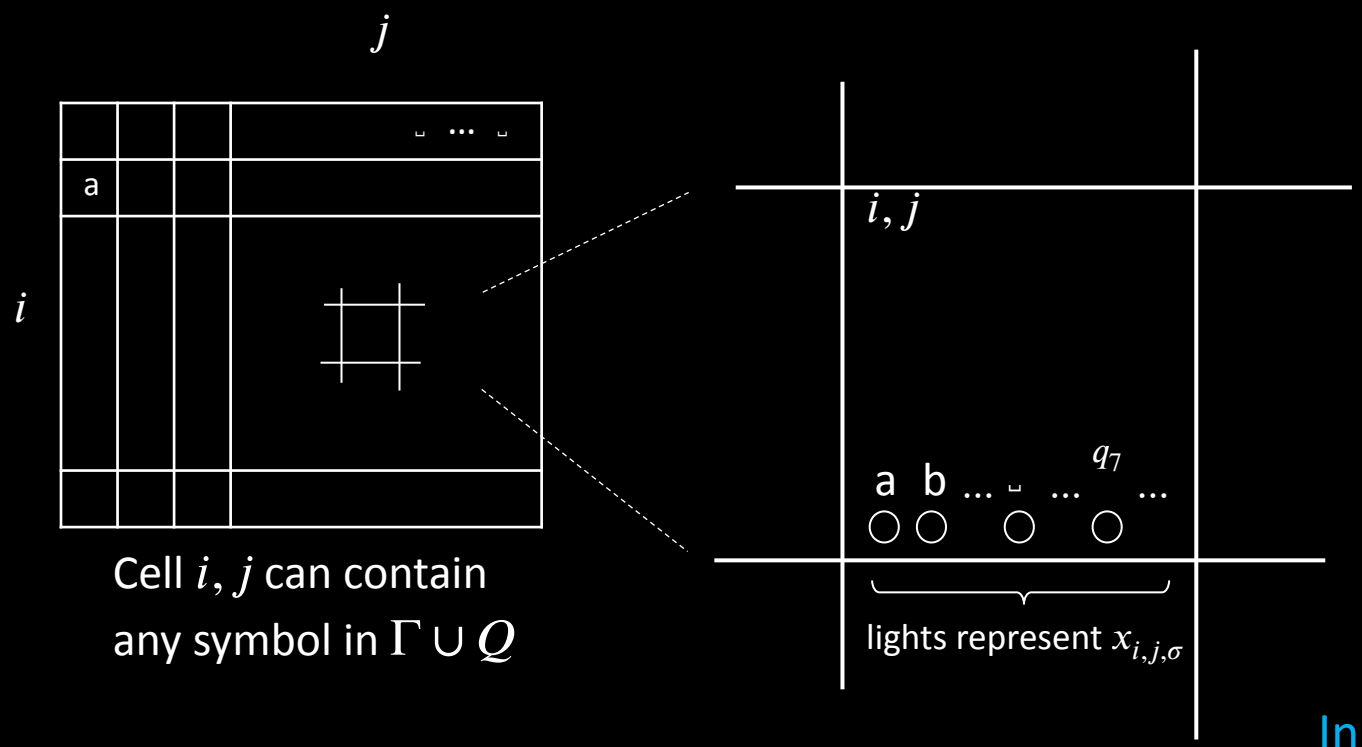
ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma}$ = TRUE for each i, j .

In every cell at least one light and at most one light

$$x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}$$

$$\phi\text{cell} = \bigwedge_{1 \le i,j \le n^k} \left(\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma} \wedge \bigwedge_{\substack{\sigma,\tau \in \Gamma \cup Q \\ \sigma \neq \tau}} \left(\overline{x_{i,j,\sigma} \wedge x_{i,j,\tau}} \right) \right)$$

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \le i, j \le n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma}$ = TRUE means cell i, j contains σ.

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{ac}$

ϕcell "says" exactly one light is on per cell i.e., exactly one $x_{i,j,\sigma}$ = TRUE for each i, j.

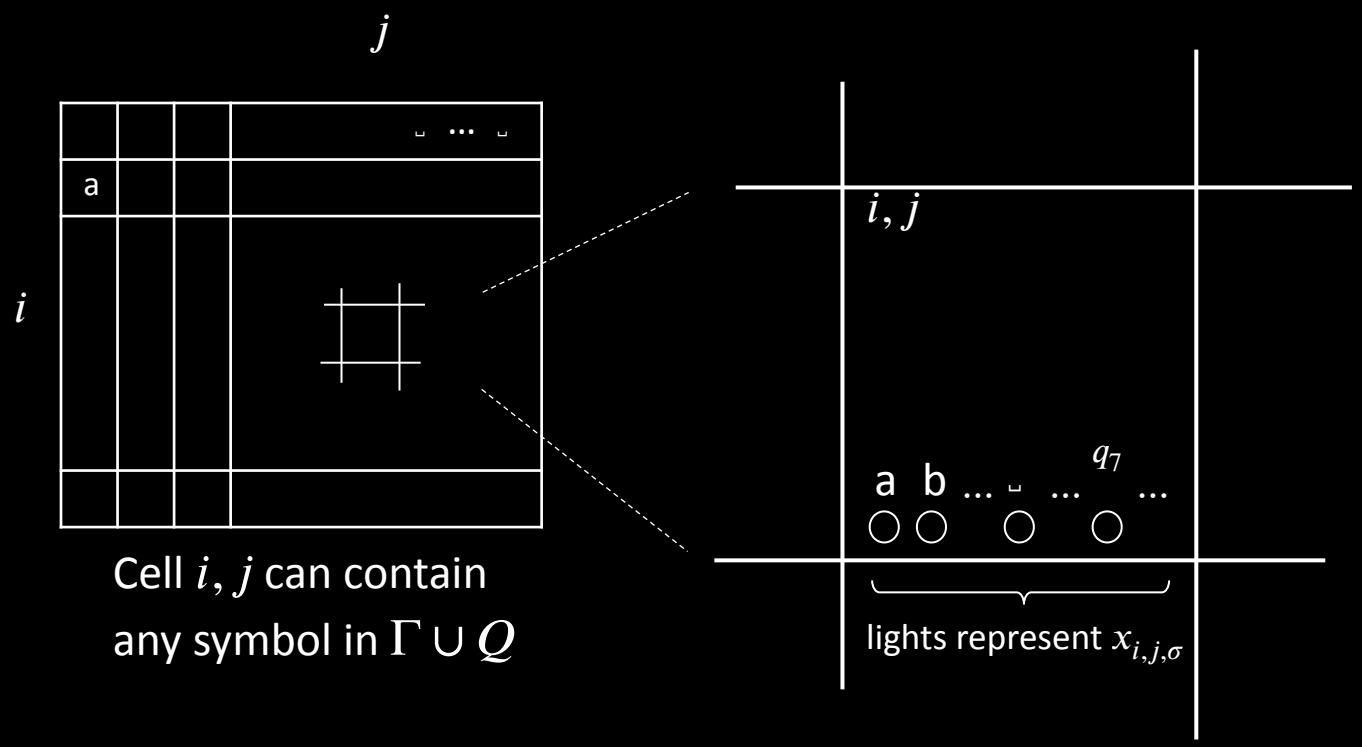
In every cell at least one light and at most one light

$$x_{i,j,\sigma_1} \vee x_{i,j,\sigma_2} \vee \cdots \vee x_{i,j,\sigma_t}$$

$$\phi\text{cell} = \bigwedge_{1 \le i,j \le n^k} \left(\bigvee_{\sigma \in \Gamma \cup Q} x_{i,j,\sigma} \wedge \bigwedge_{\substack{\sigma,\tau \in \Gamma \cup Q \\ \sigma \neq \tau}} \left(\overline{x_{i,j,\sigma} \wedge x_{i,j,\tau}} \right) \right)$$

Check-in 16.2

Constructing $\phi_{M,w}$: ϕcell



Cell i, j can contain any symbol in $\Gamma \cup Q$

The variables of $\phi_{M,w}$ are $x_{i,j,\sigma}$ for $1 \leq i, j \leq n^k$ and $\sigma \in \Gamma \cup Q$.

$x_{i,j,\sigma} =$ TRUE means cell i, j contains σ.

Check-in 16.2

How many variables does $\phi_{M,w}$ have?

Recall that $n = |w|$.

(a) $O(n)$

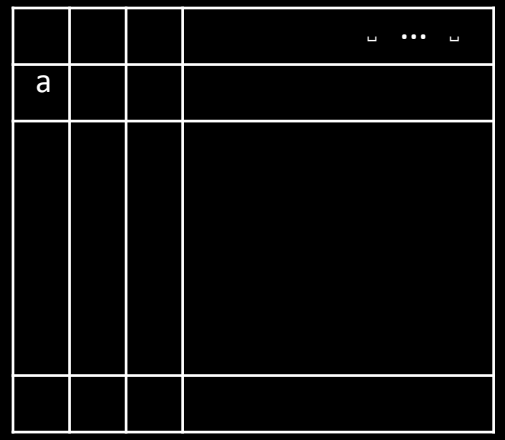
(b) $O(n^2)$

(c) $O(n^k)$

(d) $O(n^{2k})$

Check-in 16.2

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



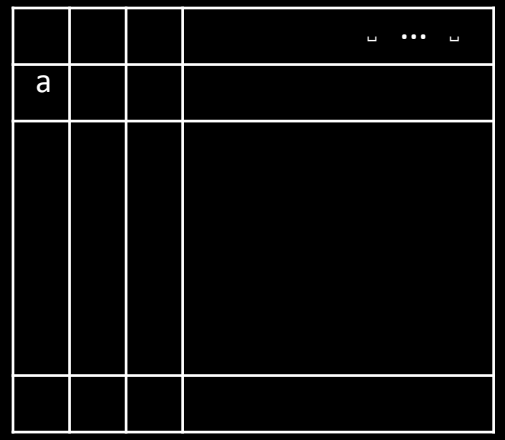
← Start configuration

← Accepting configuration

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



← Start configuration

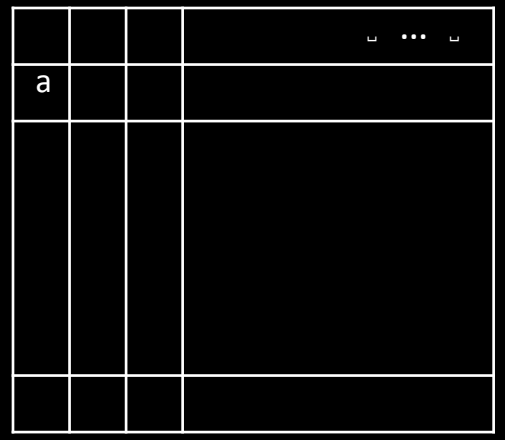
← Accepting configuration

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



← Start configuration

← Accepting configuration

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

ϕstart

ϕaccept =

Constructing $\phi_{M,w}$: ϕstart and ϕaccept

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

$\phi\text{start} = x_{1,1,q_0}$

$\phi\text{accept} =$

Constructing $\phi_{M,w}$: ϕstart and ϕaccept

$\phi_{M,w}$ "says" a tableau for M on w exists.

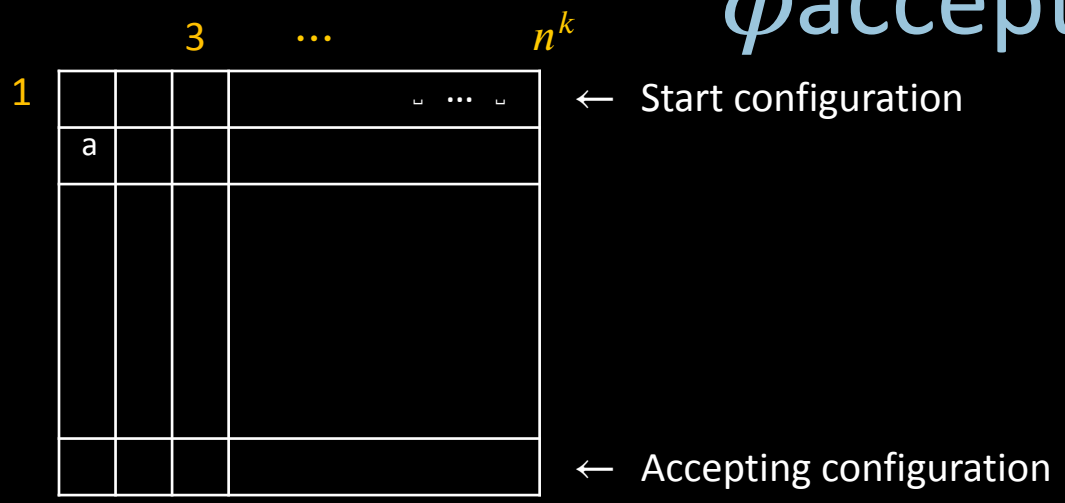
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

$\phi\text{start} = x_{1,1,q_0} \wedge x_{1,2,w_1}$

$\phi\text{accept} =$

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



$\phi_{M,w}$ "says" a tableau for M on w exists.

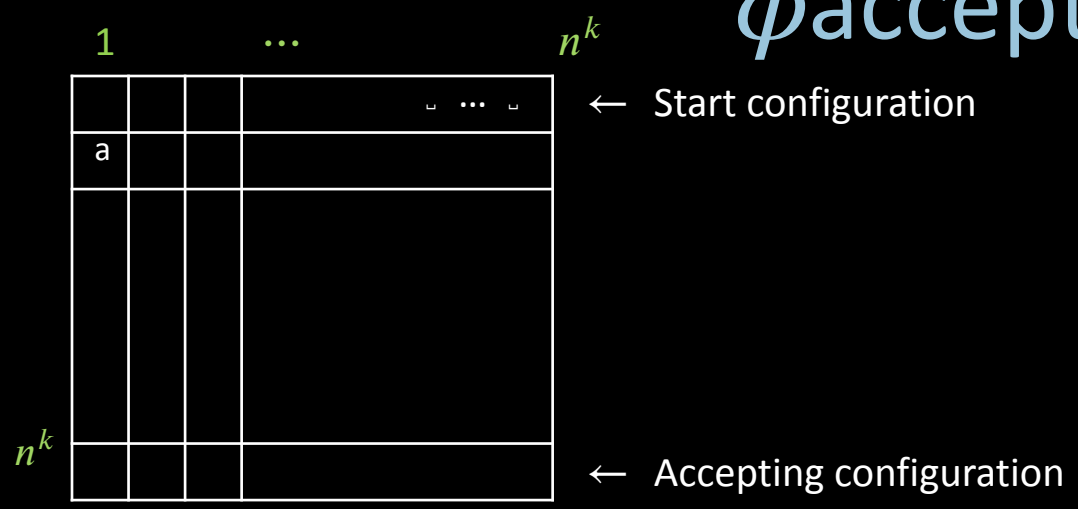
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

$\phi\text{start} = x_{1,1,q_0} \wedge x_{1,2,w_1} \wedge x_{1,3,w_2} \wedge \cdots \wedge x_{1,n^k,\sqcup}$

$\phi\text{accept} =$

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



← Start configuration

← Accepting configuration

$\phi_{M,w}$ "says" a tableau for M on w exists.

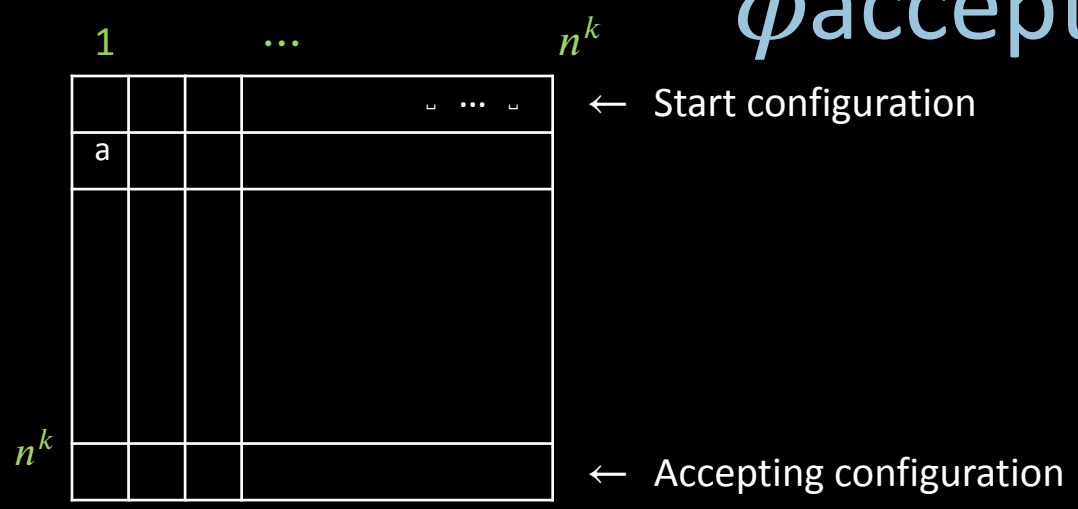
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

$\phi\text{start} = x_{1,1,q_0} \wedge x_{1,2,w_1} \wedge x_{1,3,w_2} \wedge \cdots \wedge x_{1,n^k,␣}$

$\phi\text{accept} = \bigvee_{1 \le j \le n^k} x_{n^k,j,q\text{accept}}$

Constructing $\phi_{M,w}$: ϕstart and ϕaccept



← Start configuration

← Accepting configuration

$\phi_{M,w}$ "says" a tableau for M on w exists.

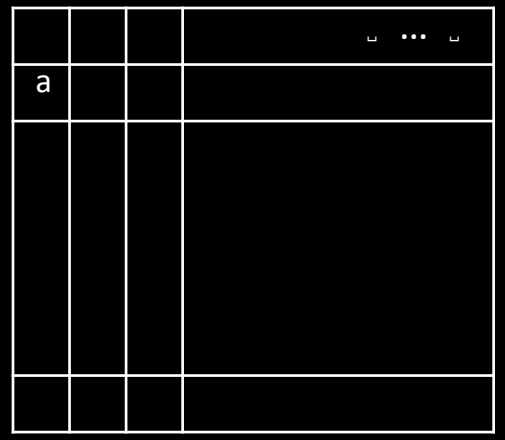
$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

ϕcell done ✓

$\phi\text{start} = x_{1,1,q_0} \wedge x_{1,2,w_1} \wedge x_{1,3,w_2} \wedge \cdots \wedge x_{1,n^k,\sqcup}$

$\phi\text{accept} = \bigvee_{1 \le j \le n^k} x_{n^k,j,q\text{accept}}$

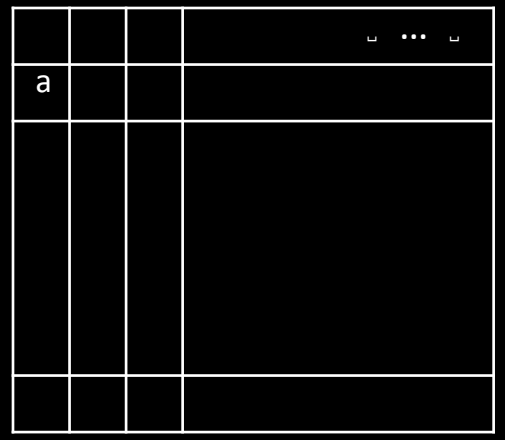
Constructing $\phi_{M,w}$: ϕmove



$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

Constructing $\phi_{M,w}$: ϕmove

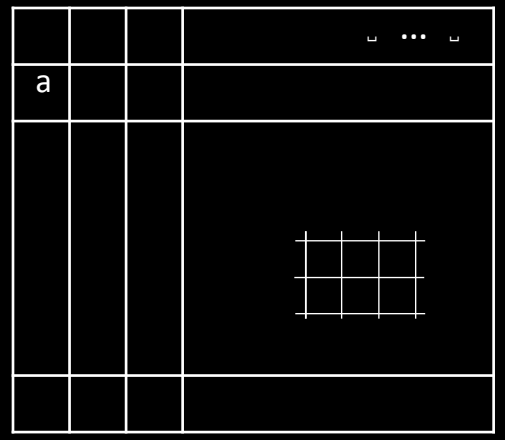


$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove

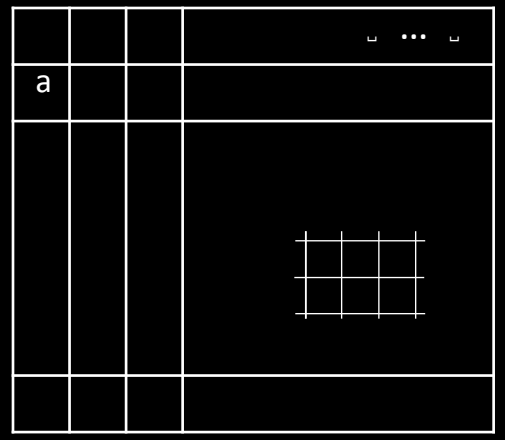


$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



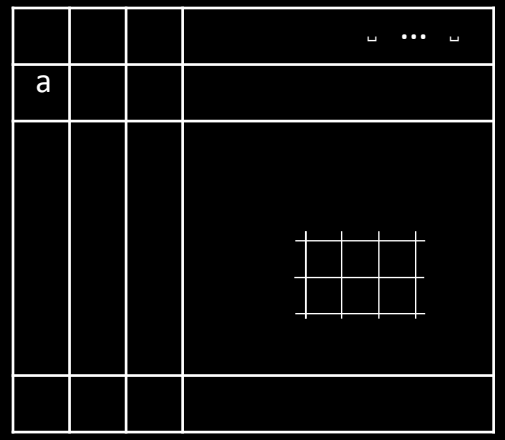
2×3 neighborhood

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

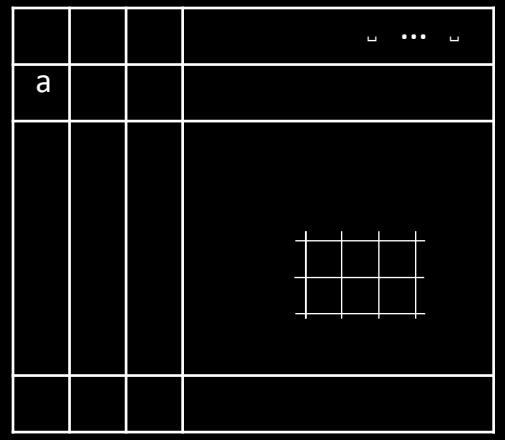
Legal neighborhoods: consistent with M's transition function

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

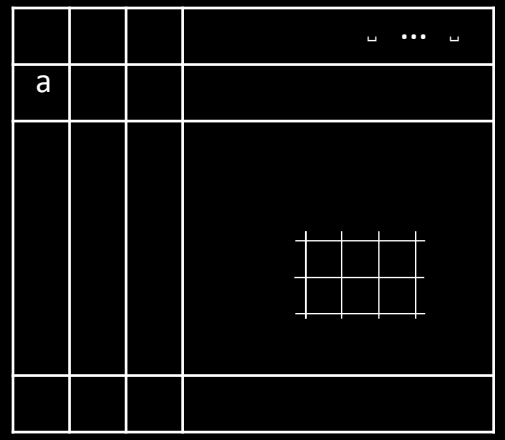
a		b
	a	c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

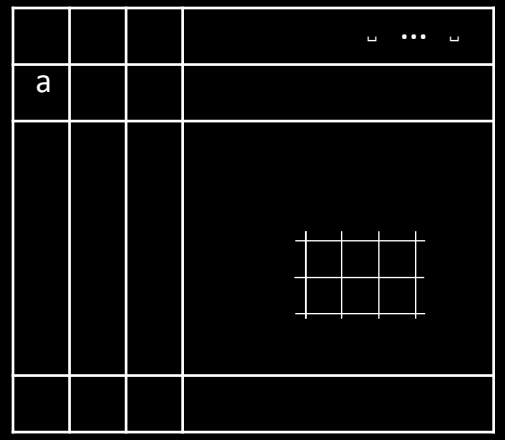
a	b	c
a	b	c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

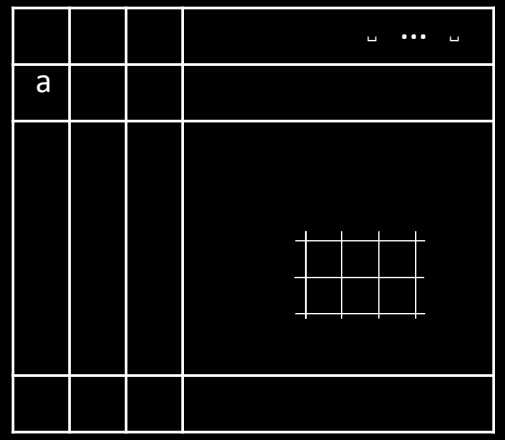
a	b	c
a	b	

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

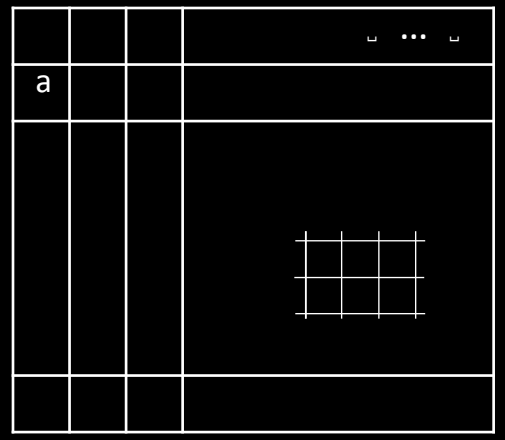
a	b	c
d	b	c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

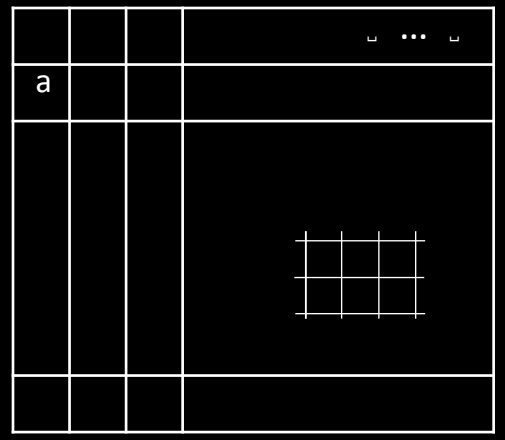
Illegal neighborhoods: not consistent with M's transition function

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

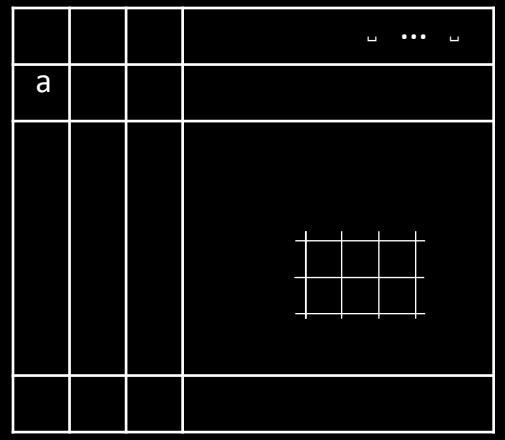
a	b	c
a	d	c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

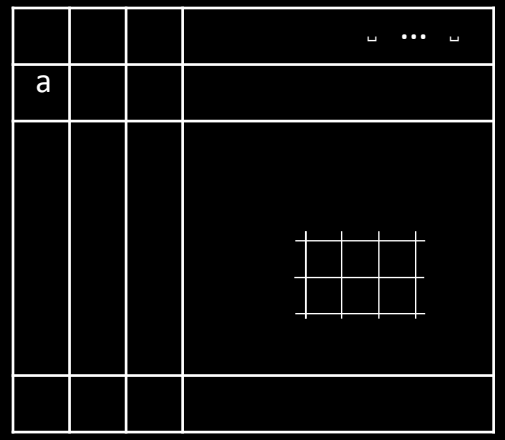
a	b	c
a		c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

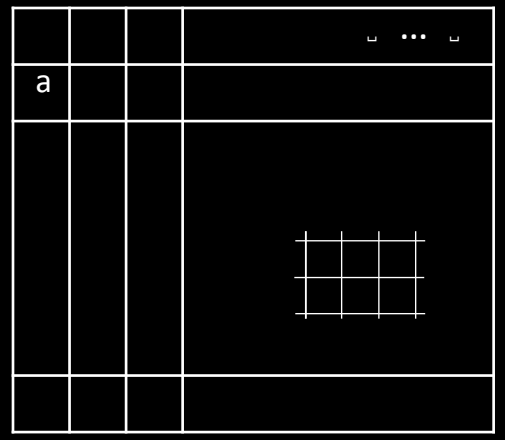
a		c
a	b	c

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

a		c
a	b	c

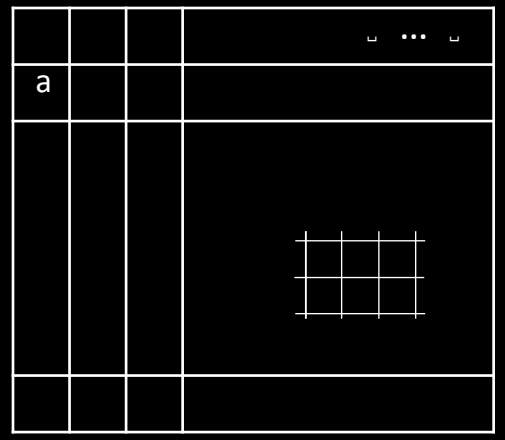
a		c
	d	

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

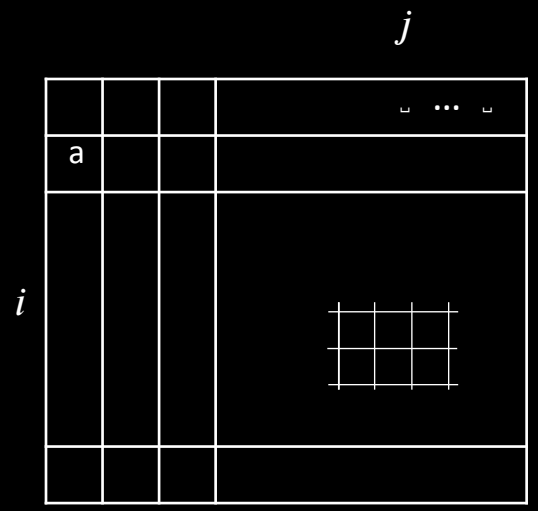
a	b	c
a		c

a		c
a	b	c

a		c
	d	

Claim: If every 2×3 neighborhood is legal then tableau corresponds to a computation history.

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

a		c
a	b	c

a		c
	d	

$\phi_{M,w}$ "says" a tableau for M on w exists.

Claim: If every 2×3 neighborhood is legal then tableau corresponds to a computation history.

$$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$$

✓ ✓ ✓

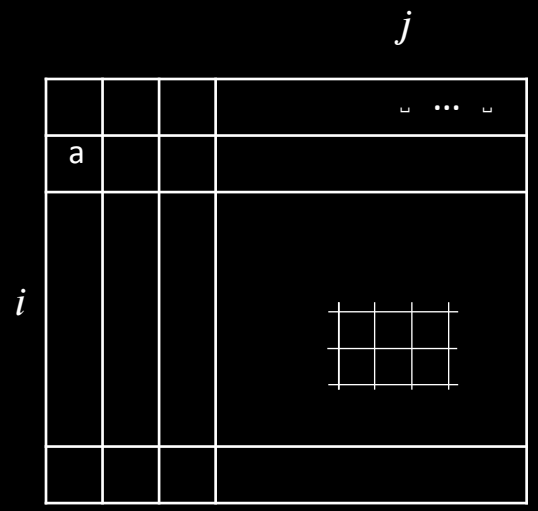
$$\bigvee \left(x_{i,j-1},\mathsf{r} \wedge x_{i,j},\mathsf{s} \wedge x_{i,j+1},\mathsf{t} \wedge x_{i+1,j-1},\mathsf{v} \wedge x_{i+1,j},\mathsf{y} \wedge x_{i+1,j+1},\mathsf{z} \right)$$

Legal

r	s	t
v	y	z

Says that the neighborhood at i, j is legal

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

a		c
a	b	c

a		c
	d	

$\phi_{M,w}$ "says" a tableau for M on w exists.

Claim: If every 2×3 neighborhood is legal then tableau corresponds to a computation history.

$$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$$

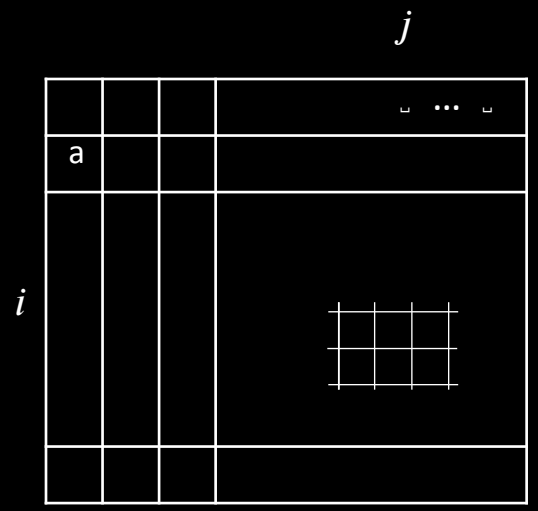
✓ ✓ ✓

$$\bigwedge_{1<i,j<n^k} \left(\bigvee_{\text{Legal}} \left(x_{i,j-1},\mathsf{r} \wedge x_{i,j},\mathsf{s} \wedge x_{i,j+1},\mathsf{t} \wedge x_{i+1,j-1},\mathsf{v} \wedge x_{i+1,j},\mathsf{y} \wedge x_{i+1,j+1},\mathsf{z} \right) \right)$$

r	s	t
v	y	z

Says that the neighborhood at i, j is legal

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

a		c
a	b	c

a		c
	d	

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

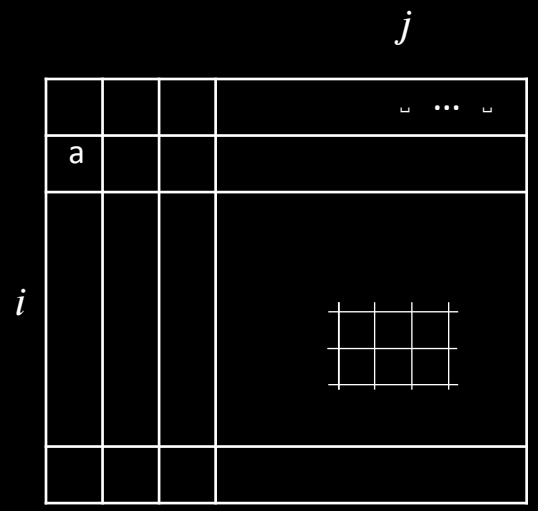
Claim: If every 2×3 neighborhood is legal then tableau corresponds to a computation history.

$$\phi\text{move} = \bigwedge_{1<i,j<n^k} \left(\bigvee_{\text{Legal}} \left(x_{i,j-1},\text{r} \wedge x_{i,j},\text{s} \wedge x_{i,j+1},\text{t} \wedge x_{i+1,j-1},\text{v} \wedge x_{i+1,j},\text{y} \wedge x_{i+1,j+1},\text{z} \right) \right)$$

r	s	t
v	y	z

Says that the neighborhood at i, j is legal

Constructing $\phi_{M,w}$: ϕmove



2×3 neighborhood

Legal neighborhoods: consistent with M's transition function

potential examples:

a		b
	a	c

a	b	c
a	b	c

a	b	c
a	b	

a	b	c
d	b	c

Illegal neighborhoods: not consistent with M's transition function

examples:

a	b	c
a	d	c

a	b	c
a		c

a		c
a	b	c

a		c
	d	

$\phi_{M,w}$ "says" a tableau for M on w exists.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

✓ ✓ ✓

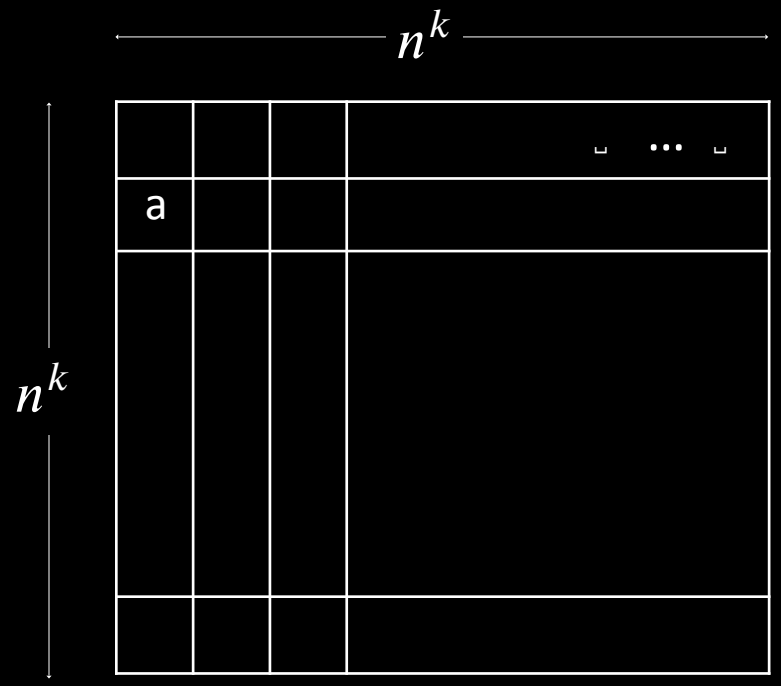
Claim: If every 2×3 neighborhood is legal then tableau corresponds to a computation history.

$$\phi\text{move} = \bigwedge_{1<i,j<n^k} \left(\bigvee_{\text{Legal}} \left(x_{i,j-1},\text{r} \wedge x_{i,j},\text{s} \wedge x_{i,j+1},\text{t} \wedge x_{i+1,j-1},\text{v} \wedge x_{i+1,j},\text{y} \wedge x_{i+1,j+1},\text{z} \right) \right)$$

r	s	t
v	y	z

Says that the neighborhood at i, j is legal

Conclusion: SAT is NP-complete



Summary:

For $A \in$ NP, decided by NTM M,

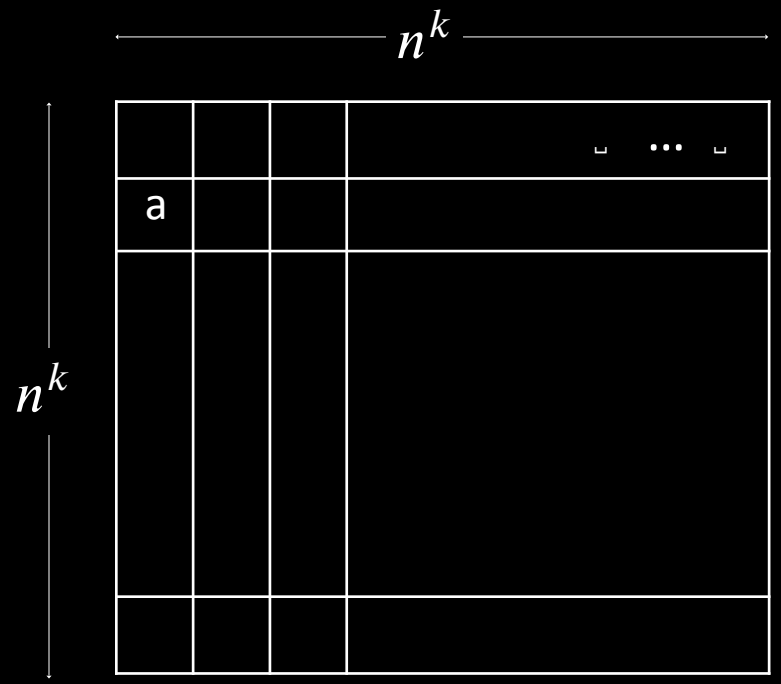
we gave a reduction f from A to SAT:

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable.

Conclusion: SAT is NP-complete



Summary:

For $A \in$ NP, decided by NTM M,

we gave a reduction f from A to SAT:

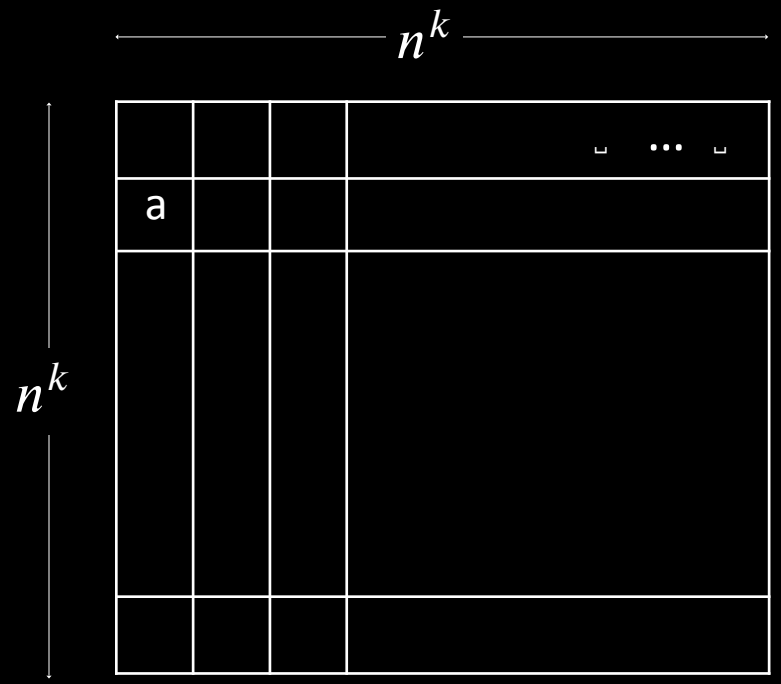
$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

Conclusion: SAT is NP-complete



Summary:

For $A \in$ NP, decided by NTM M,

we gave a reduction f from A to SAT:

$f: \Sigma^* \rightarrow$ formulas

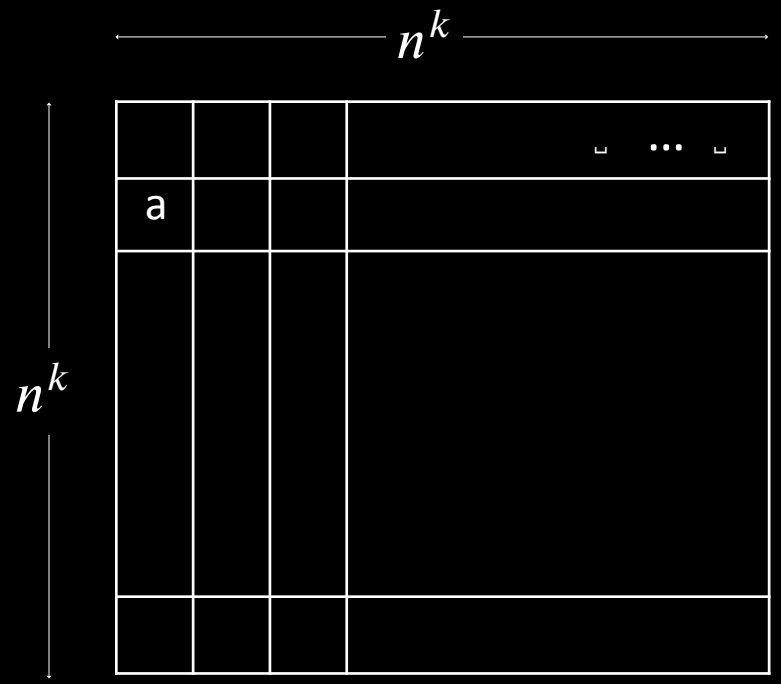
$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

The size of $\phi_{M,w}$ is roughly the size of the tableau for M on w, so size is $O(n^k \times n^k) = O(n^{2k})$.

Conclusion: SAT is NP-complete



Summary:

For $A \in$ NP, decided by NTM M,

we gave a reduction f from A to SAT:

f: $\Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

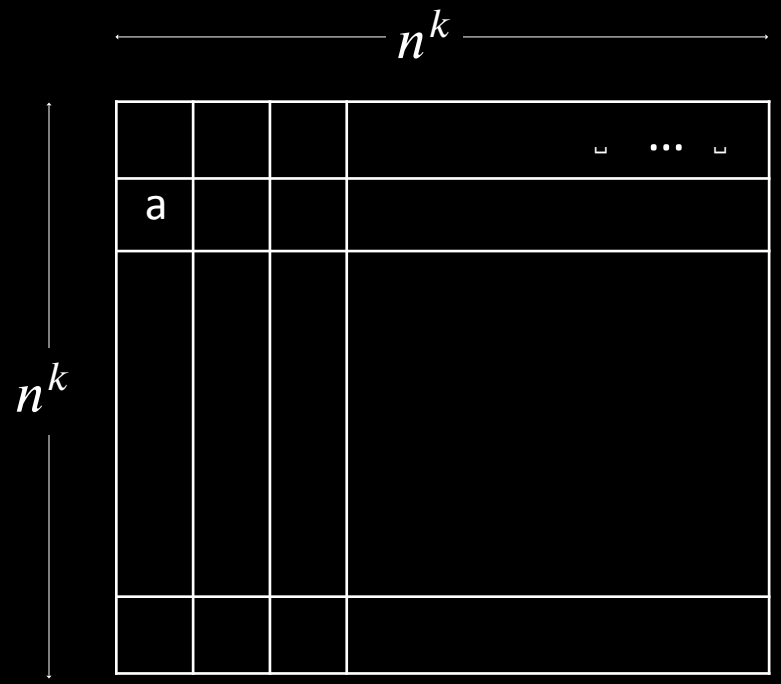
$w \in A$ iff $\phi_{M,w}$ is satisfiable.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

The size of $\phi_{M,w}$ is roughly the size of the tableau for M on w, so size is $O(n^k \times n^k) = O(n^{2k})$.

Therefore f is computable in polynomial time.

Conclusion: SAT is NP-complete



Summary:

For $A \in$ NP, decided by NTM M,

we gave a reduction f from A to SAT:

$f: \Sigma^* \rightarrow$ formulas

$f(w) = \langle \phi_{M,w} \rangle$

$w \in A$ iff $\phi_{M,w}$ is satisfiable.

$\phi_{M,w} = \phi\text{cell} \wedge \phi\text{start} \wedge \phi\text{move} \wedge \phi\text{accept}$

The size of $\phi_{M,w}$ is roughly the size of the tableau for M on w, so size is $O(n^k \times n^k) = O(n^{2k})$.

Therefore f is computable in polynomial time.

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

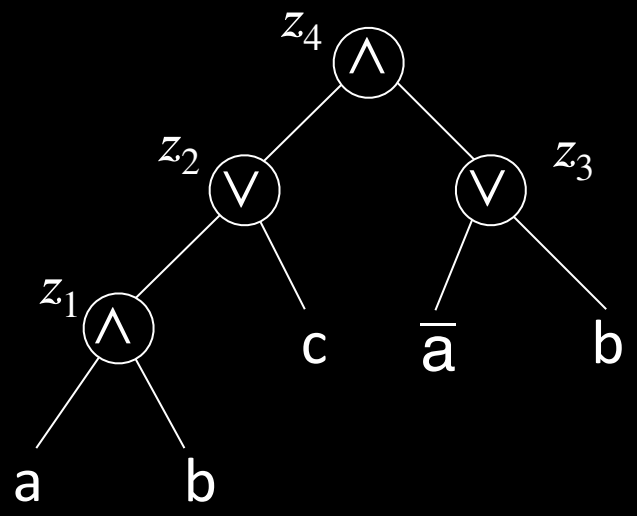
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

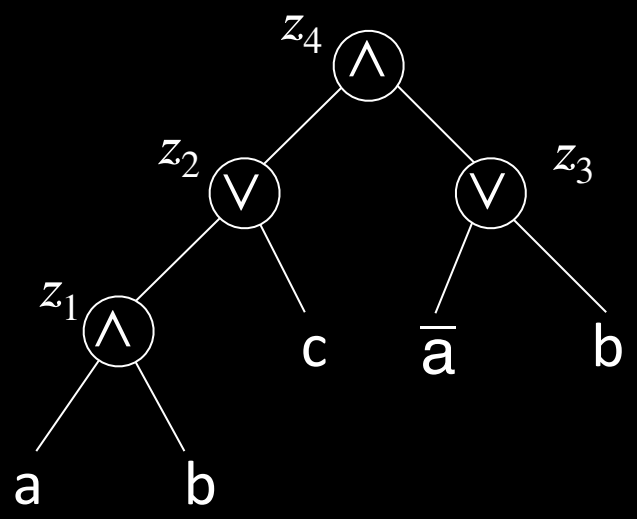
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $\big(\overline{A} \vee B\big)$ $\overline{(A \wedge B)}$ and $\big(\overline{A} \vee \overline{B}\big)$

$3SAT$ is NP-complete

Theorem: $3SAT$ is NP-complete

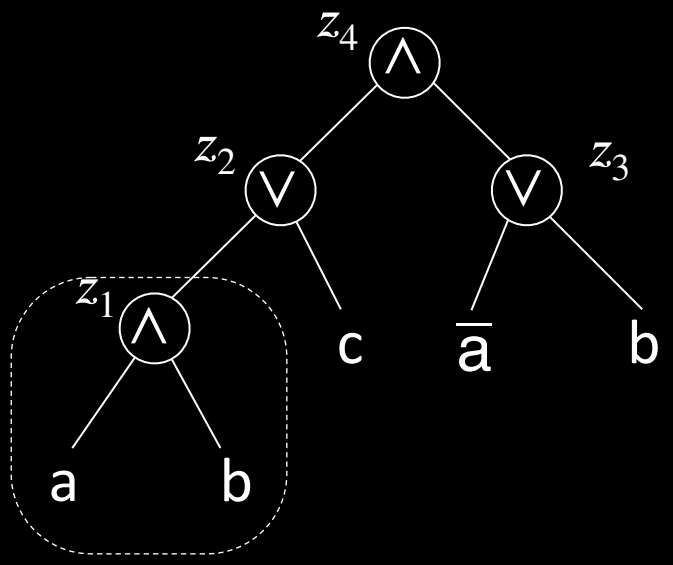
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $\big(\overline{A} \vee B\big)$ $\overline{(A \wedge B)}$ and $\big(\overline{A} \vee \overline{B}\big)$

$3SAT$ is NP-complete

a	b	a ∧ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	0	$(\bar{a} \wedge b) \rightarrow \bar{c}$
1	0	0	$(a \wedge \bar{b}) \rightarrow \bar{c}$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

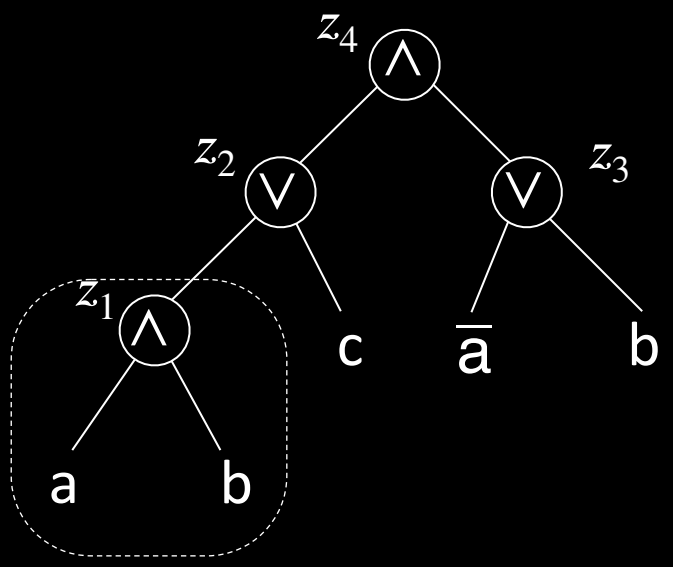
Theorem: $3SAT$ is NP-complete

Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.
(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $\big(\bar{A} \vee B\big)$ $\overline{(A \wedge B)}$ and $\big(\bar{A} \vee \bar{B}\big)$

$3SAT$ is NP-complete

a	b	a ∧ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	0	$(\bar{a} \wedge b) \rightarrow \bar{c}$
1	0	0	$(a \wedge \bar{b}) \rightarrow \bar{c}$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

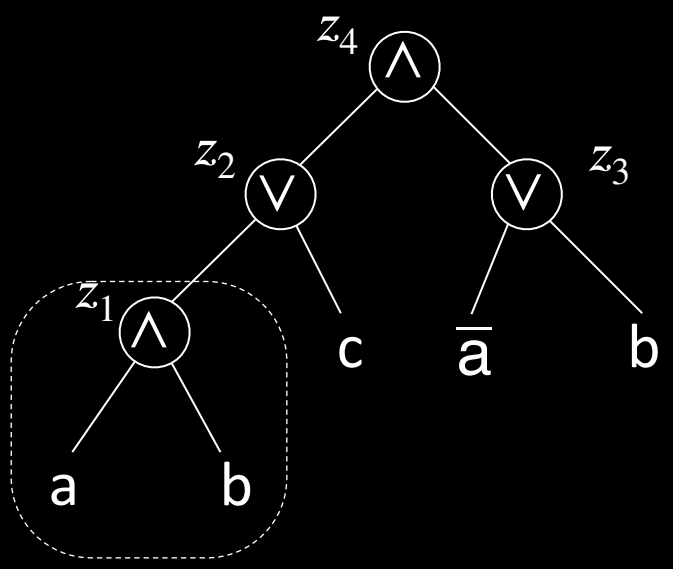
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$

$3SAT$ is NP-complete

a	b	a ∧ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	0	$(\bar{a} \wedge b) \rightarrow \bar{c}$
1	0	0	$(a \wedge \bar{b}) \rightarrow \bar{c}$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

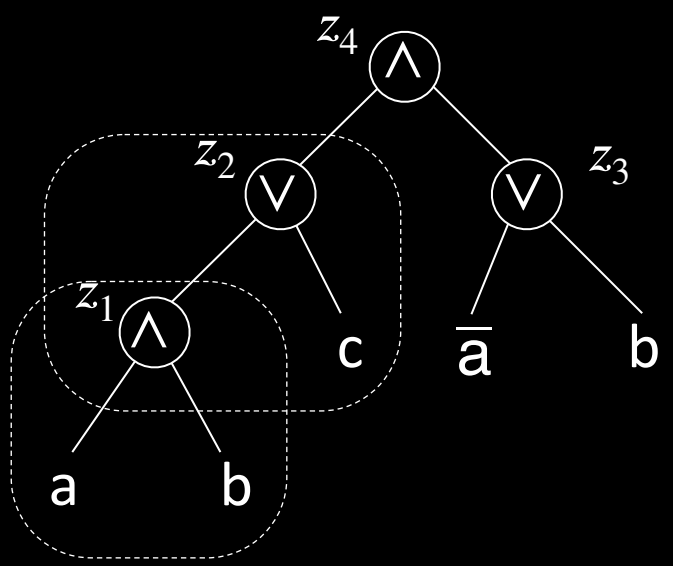
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

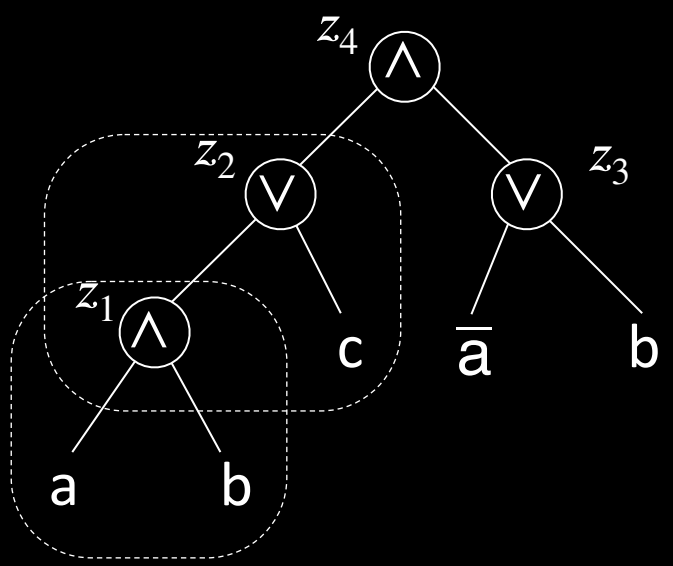
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

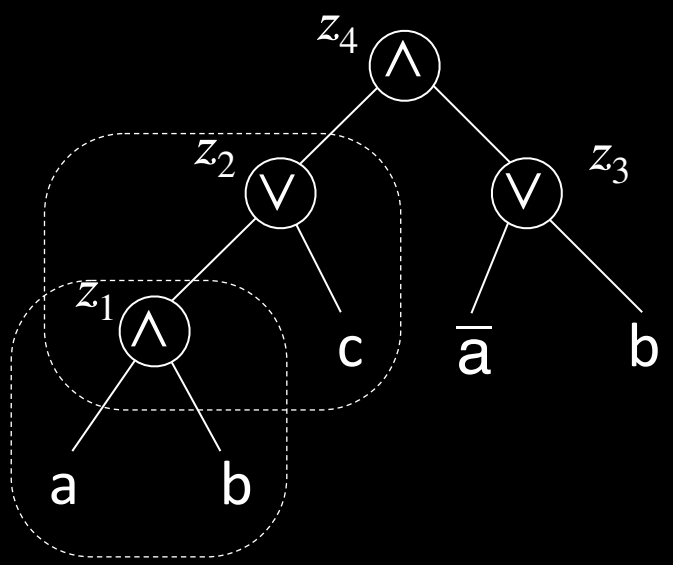
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\bar{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\bar{A} \vee \bar{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \bar{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge \bar{c}) \rightarrow \bar{z_2}\big)$$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

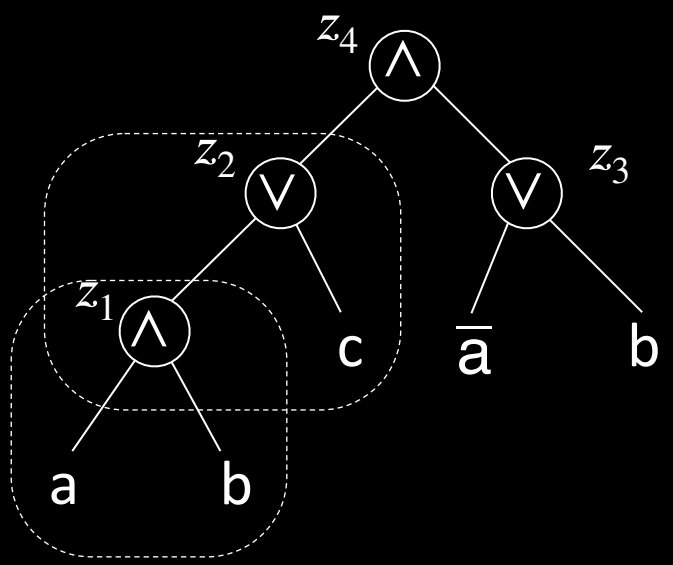
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\overline{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\overline{z_1} \wedge \bar{c}) \rightarrow \overline{z_2}\big)$$

⋮ repeat for each z_i

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

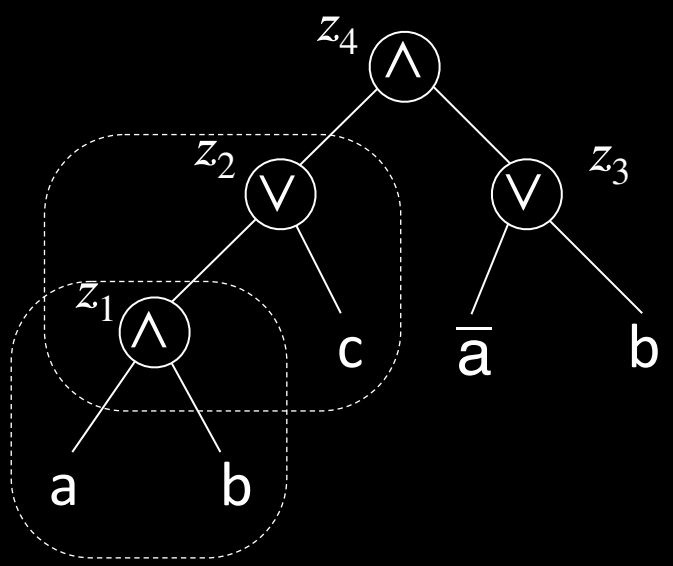
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big(\overline{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big(\overline{z_1} \wedge \bar{c}) \rightarrow \overline{z_2}\big)$$
$$\vdots \text{ repeat for each } z_i$$
$$\wedge \ (z_4)$$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

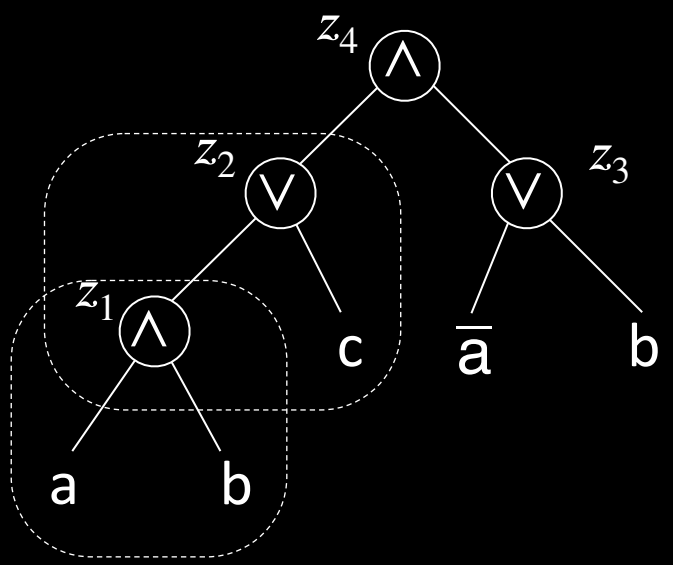
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\overline{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\overline{A} \vee \overline{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \overline{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \overline{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\overline{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\overline{z_1} \wedge \bar{c}) \rightarrow \overline{z_2}\big)$$
$$\vdots \text{ repeat for each } z_i$$
$$\wedge \ (z_4)$$

Observe that $\big((a \wedge b) \rightarrow c\big)$ is logically equivalent to $\big(\bar{a} \vee \bar{b} \vee c\big)$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

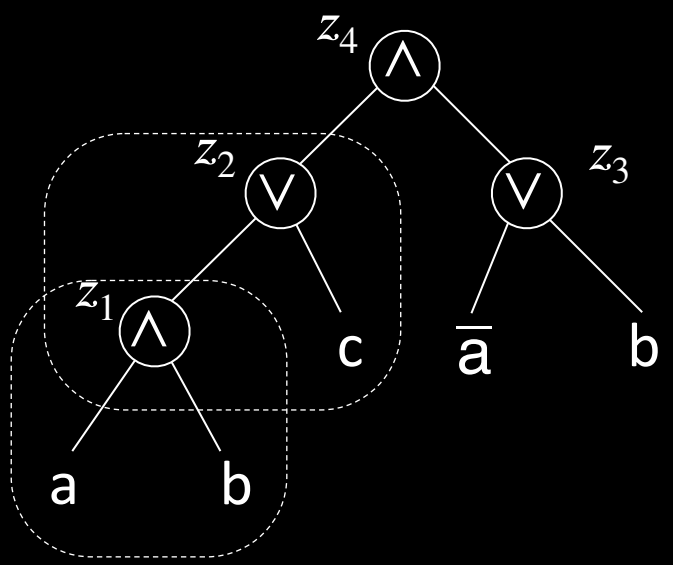
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\bar{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\bar{A} \vee \bar{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \bar{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge \bar{c}) \rightarrow \bar{z_2}\big)$$
$$\vdots \text{ repeat for each } z_i$$
$$\wedge \ (z_4)$$

Observe that $\big((a \wedge b) \rightarrow c\big)$ is logically equivalent to $(\bar{a} \vee \bar{b} \vee c)$

$$\Big[\big((a \wedge b) \rightarrow c\big) \leftrightarrow \big(\overline{(a \wedge b)} \vee c\big) \leftrightarrow \Big(\big(\bar{a} \vee \bar{b}\big) \vee c\Big) \leftrightarrow \big(\bar{a} \vee \bar{b} \vee c\big)\Big]$$

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

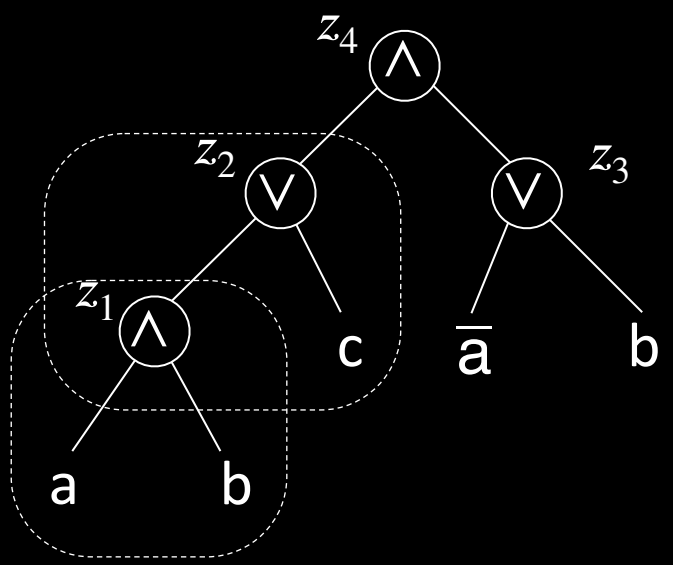
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\bar{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\bar{A} \vee \bar{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \bar{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge \bar{c}) \rightarrow \bar{z_2}\big)$$
$$\vdots \text{ repeat for each } z_i$$
$$\wedge \ (z_4)$$

Observe that $\big((a \wedge b) \rightarrow c\big)$ is logically equivalent to $\big(\bar{a} \vee \bar{b} \vee c\big)$

$$\Big[\big((a \wedge b) \rightarrow c\big) \leftrightarrow \big(\overline{(a \wedge b)} \vee c\big) \leftrightarrow \Big(\big(\bar{a} \vee \bar{b}\big) \vee c\Big) \leftrightarrow \big(\bar{a} \vee \bar{b} \vee c\big)\Big]$$

Check-in 16.3

$3SAT$ is NP-complete

a	b	a ∨ b = c	
1	1	1	$(a \wedge b) \rightarrow c$
0	1	1	$(\bar{a} \wedge b) \rightarrow c$
1	0	1	$(a \wedge \bar{b}) \rightarrow c$
0	0	0	$(\bar{a} \wedge \bar{b}) \rightarrow \bar{c}$

Theorem: $3SAT$ is NP-complete

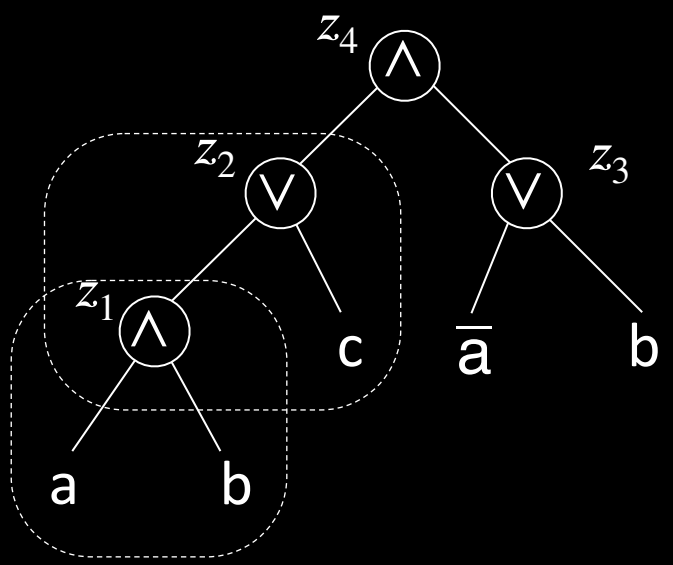
Proof: Show $SAT \leq_P 3SAT$

Give reduction f converting formula ϕ to 3CNF formula ϕ', preserving satisfiability.

(Note: ϕ and ϕ' are not logically equivalent)

Example: Say $\phi = \big((a \wedge b) \vee c\big) \wedge (\bar{a} \vee b)$

Tree structure for ϕ:



Logical equivalence: $(A \rightarrow B)$ and $(\bar{A} \vee B)$ $\quad \overline{(A \wedge B)}$ and $(\bar{A} \vee \bar{B})$

$$\phi' = \big((a \wedge b) \rightarrow z_1\big) \wedge \big((\bar{a} \wedge b) \rightarrow \bar{z_1}\big) \wedge \Big(\big(a \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big) \wedge \Big(\big(\bar{a} \wedge \bar{b}\big) \rightarrow \bar{z_1}\Big)$$
$$\wedge \big((z_1 \wedge c) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge c) \rightarrow z_2\big) \wedge \big((z_1 \wedge \bar{c}) \rightarrow z_2\big) \wedge \big((\bar{z_1} \wedge \bar{c}) \rightarrow \bar{z_2}\big)$$
$$\vdots \quad \text{repeat for each } z_i$$

Check-in 16.3

If ϕ has k operations ($\wedge$ and $\vee$), how many clauses has ϕ'?

(a) $k+1$

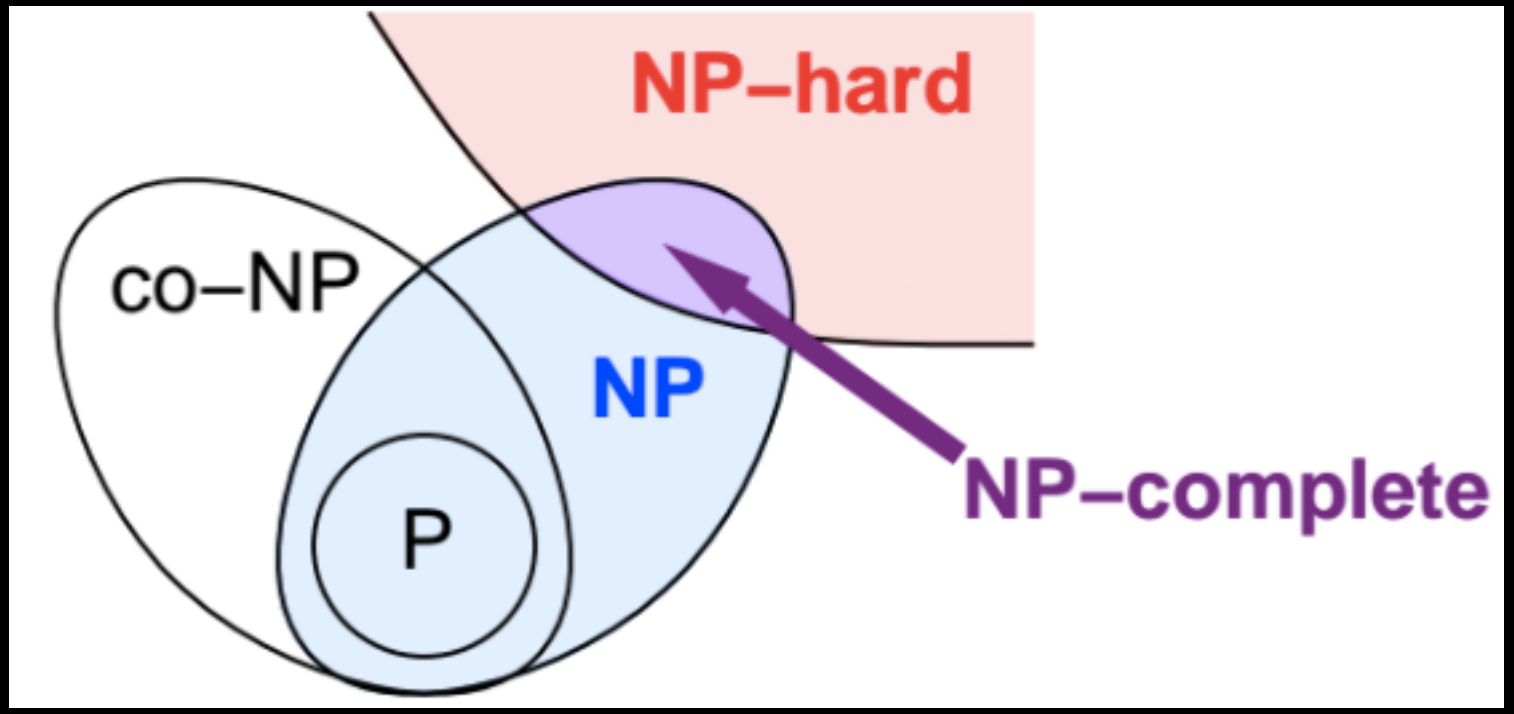
(c) k^2

(b) $4k+1$

(d) $2k^2$

coNP

- Dfn: coNP = { L | $\bar{L} \in$ NP }
- NOT-HAM



NP and certificates

- HAM is NP
- A P-time checkable certificate system:
  - Example HAM:
    - Certificate: permutation
    - Check: if each vertex <-> next vertex
- Theorem: A has P-time checkable certificate system iff A is NP
- Proof:
  - 1) P-time checkable certificate system -> NP
  - 2) NP -> P-time checkable certificate system

P vs NP

- Power of non-determinism
- Checking is easier than finding
  - معما چو حل گشت آسان شود
  - Proofing vs teaching
  - A good music
  - A good idea
  - A good business

What if P = NP

- Mathematicians could be replaced by efficient theorem-discovering programs
- Powerful AIs
- Inventors and engineers would be replaced
- Theoretiacl physics would be easy
- Randomness
- No priavcy, no SSL, no RSA, no PGP, no Bitcoin, …
- Innovation

Quick review of today

1. SAT and 3SAT
2. NP-Completeness
3. SAT is NP-complete
4. $3SAT$ is NP-complete

Quick review of today

1. SAT and 3SAT
2. NP-Completeness
3. SAT is NP-complete
4. $3SAT$ is NP-complete