Lecture 35

More NP-complete Problems

NP-Completeness and NP-Hardness

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $\underbrace{L_1 \leq_p L_2}_{A(x)}$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

NP-Completeness and NP-Hardness

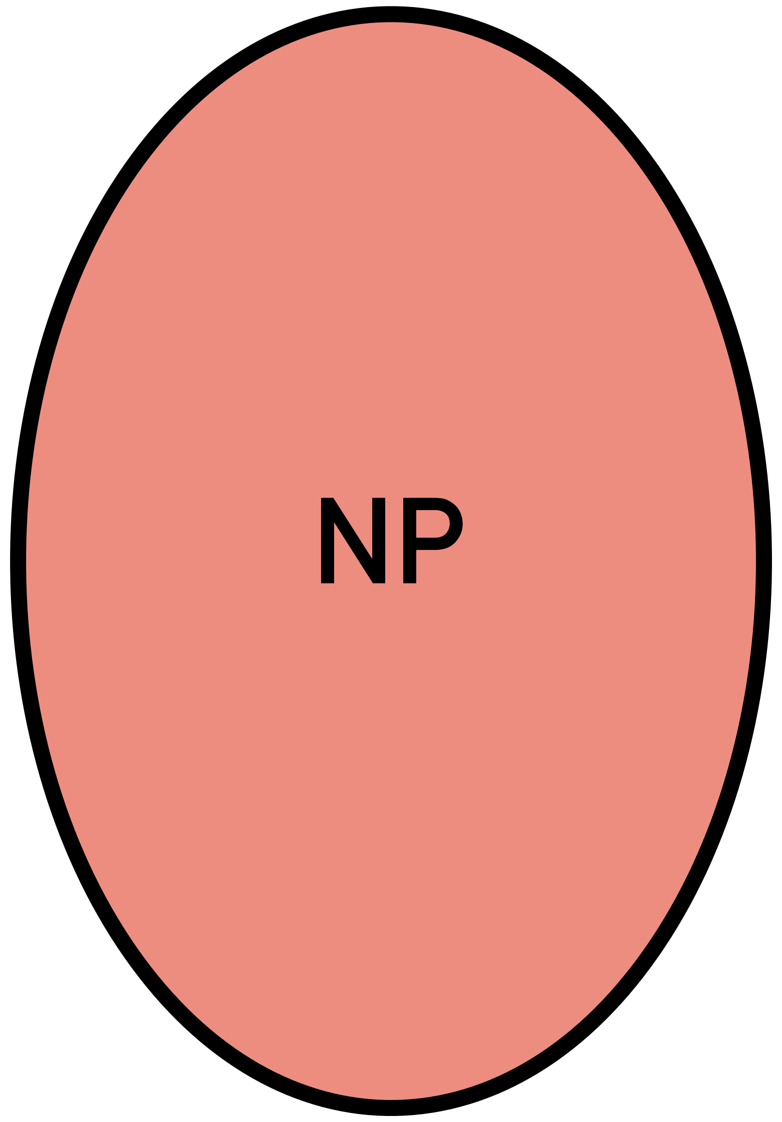
Transitivity in Reduction: If $\underbrace{L_1 \leq_p L_2}_{A(x)}$ and $\underbrace{L_2 \leq_p L_3}_{B(x)}$, then $L_1 \leq_p L_3$.

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $\underbrace{L_1 \leq_p L_2}_{A(x)}$ and $\underbrace{L_2 \leq_p L_3}_{B(x)}$, then $\underbrace{L_1 \leq_p L_3}_{B(A(x))}$.

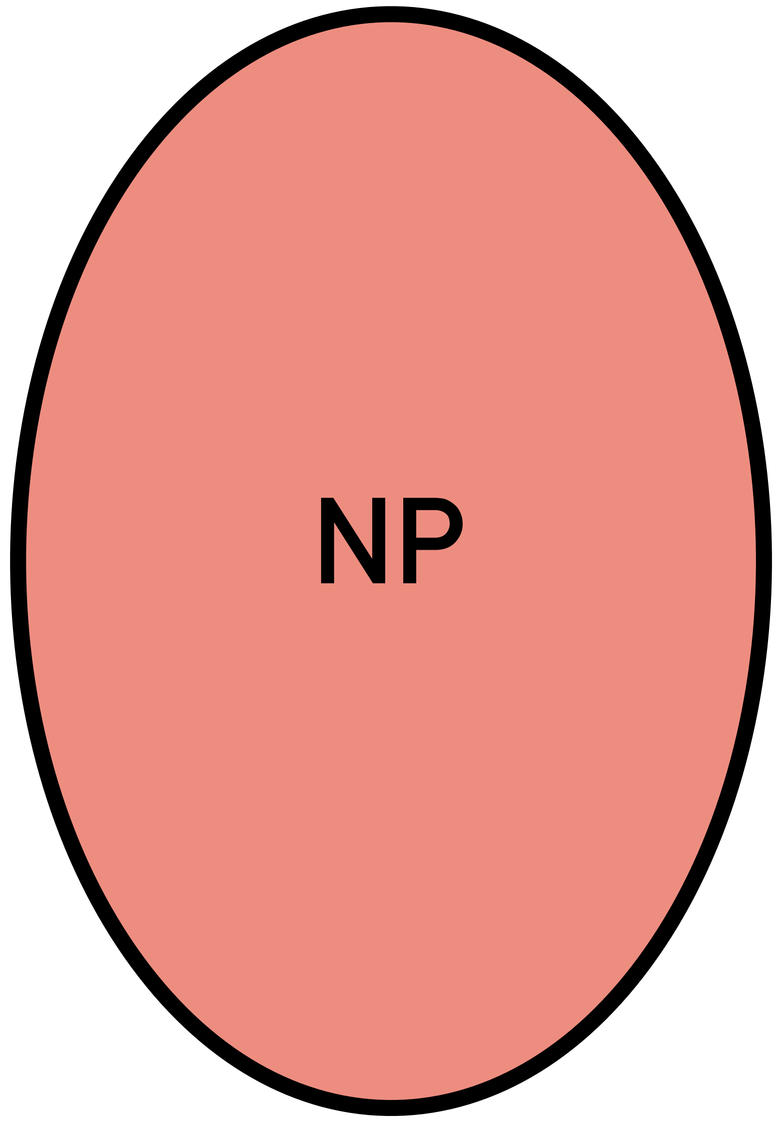
NP-Completeness and NP-Hardness

Transitivity in Reduction: If $\underbrace{L_1 \leq_p L_2}_{A(x)}$ and $\underbrace{L_2 \leq_p L_3}_{B(x)}$, then $\underbrace{L_1 \leq_p L_3}_{B(A(x))}$.



NP-Completeness and NP-Hardness

Transitivity in Reduction: If $\underbrace{L_1 \leq_p L_2}_{A(x)}$ and $\underbrace{L_2 \leq_p L_3}_{B(x)}$, then $\underbrace{L_1 \leq_p L_3}_{B(A(x))}$.

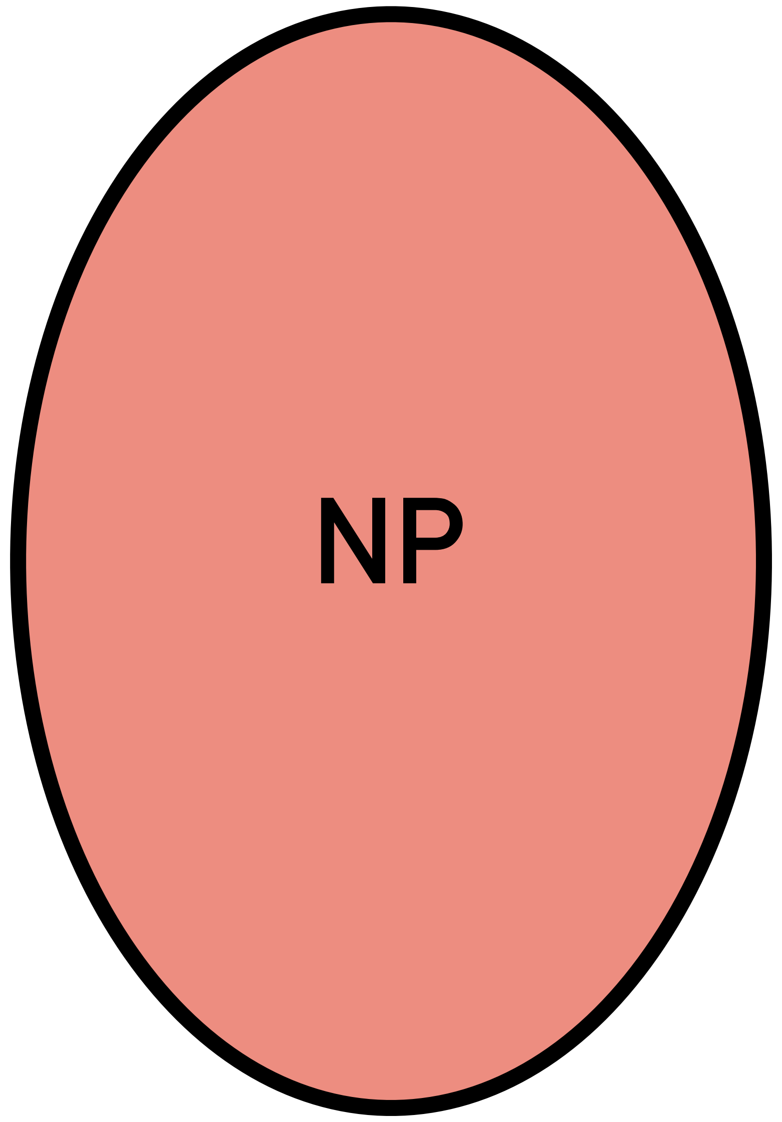


$\leq_p \quad L$

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

$$\underbrace{L_1 \leq_p L_2}_{A(x)} \qquad \underbrace{L_2 \leq_p L_3}_{B(x)} \qquad \underbrace{L_1 \leq_p L_3}_{B(A(x))}$$

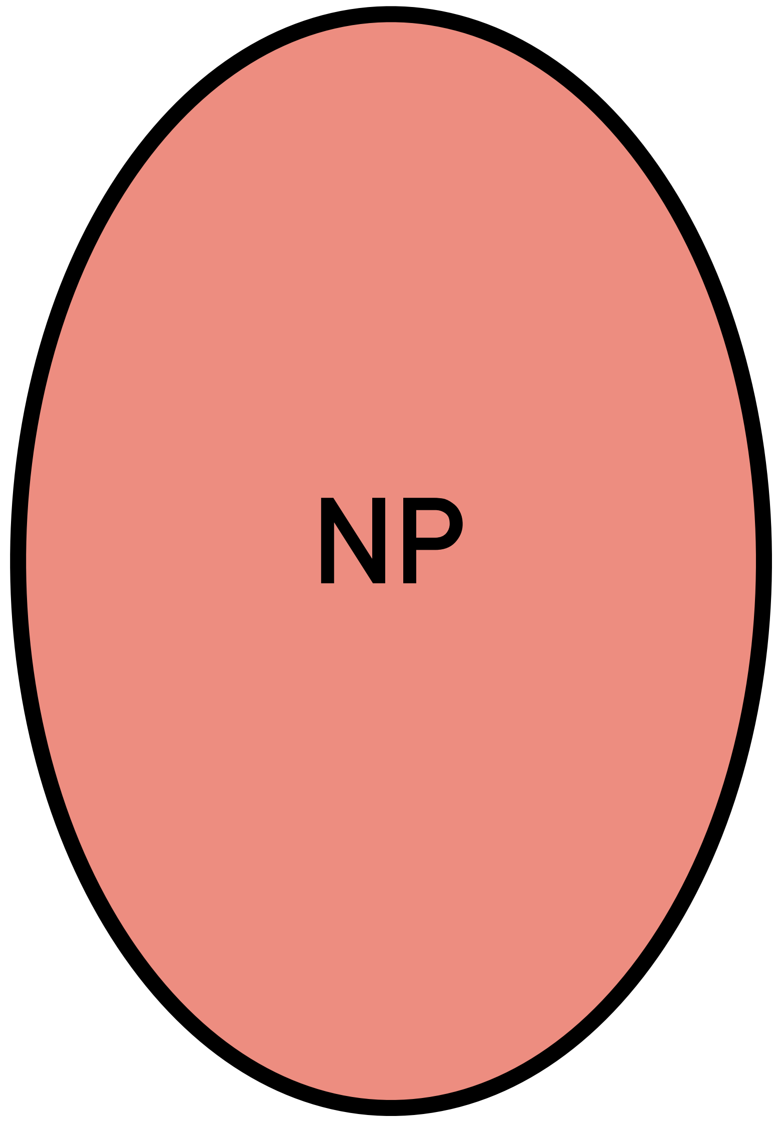


$\leq_p \; L \; \leq_p \; L'$

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

$A(x)$ $B(x)$ $B(A(x))$



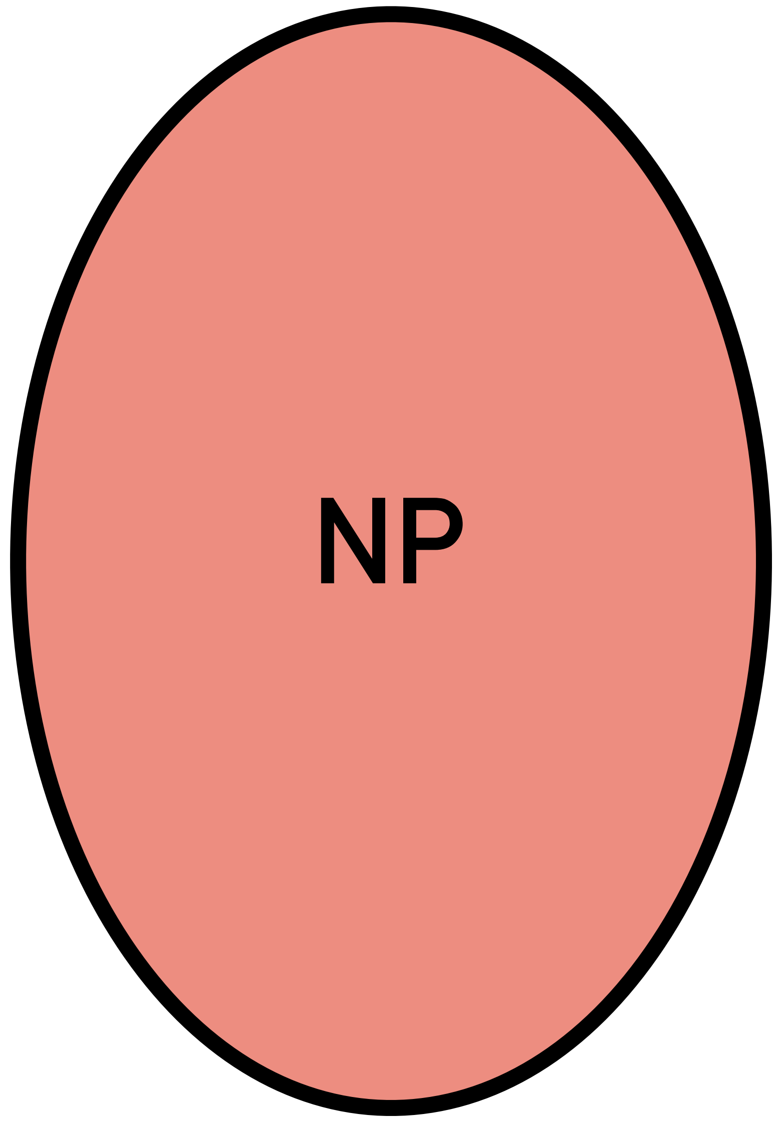
$\leq_p \quad L \quad \leq_p \quad L'$

Observation: If L is **NP-hard** and $L \leq_p L'$,

NP-Completeness and NP-Hardness

Transitivity in Reduction: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$.

$A(x)$ $B(x)$ $B(A(x))$



$\leq_p \; L \; \leq_p \; L'$

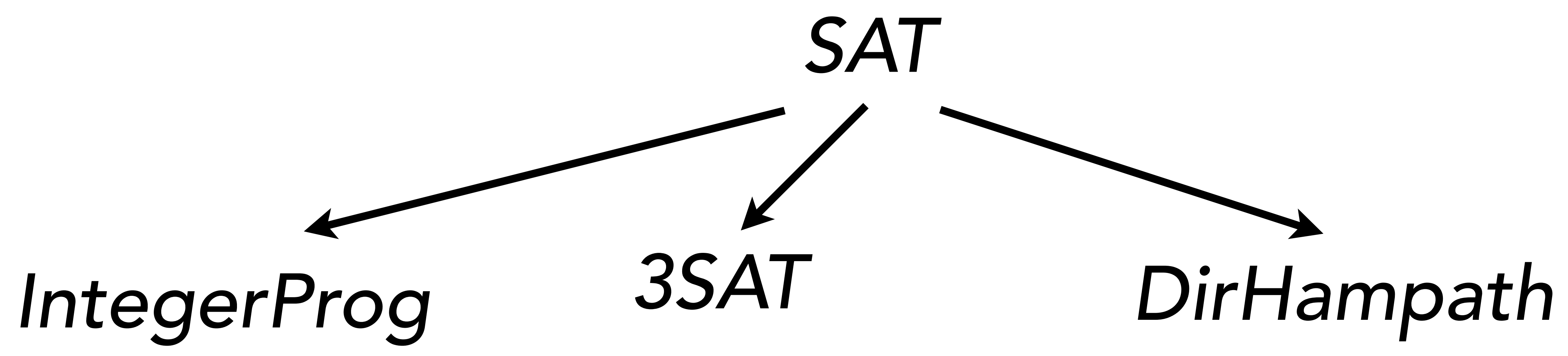
Observation: If L is **NP-hard** and $L \leq_p L'$, then L' is also **NP-hard**.

NP-complete **Network**

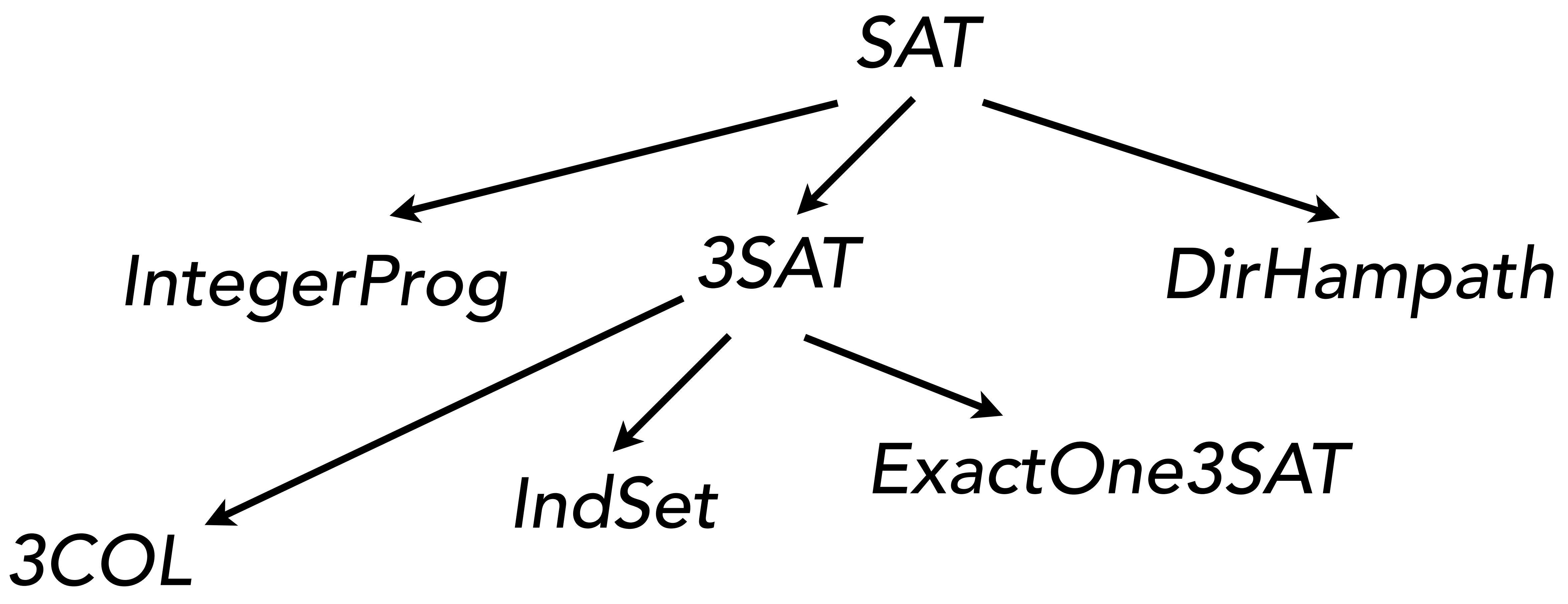
NP-complete **Network**

SAT

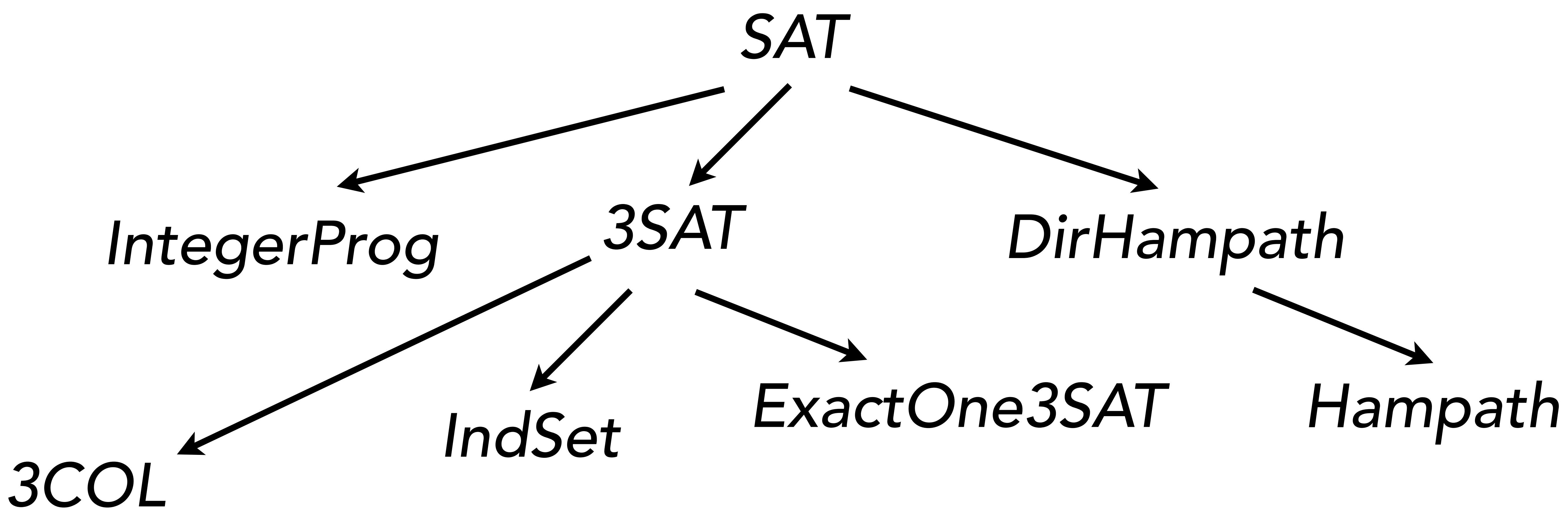
NP-complete **Network**



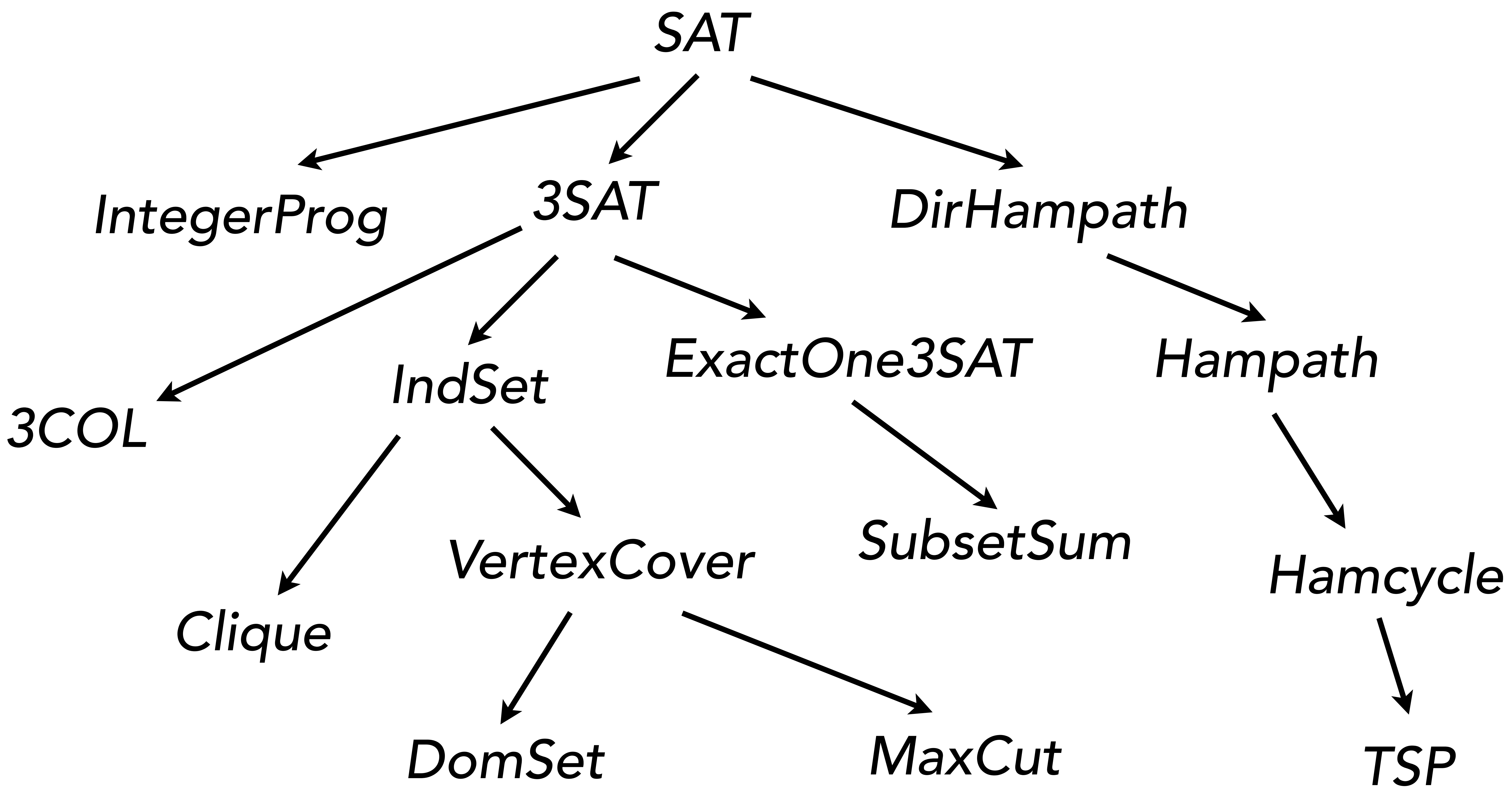
NP-complete **Network**



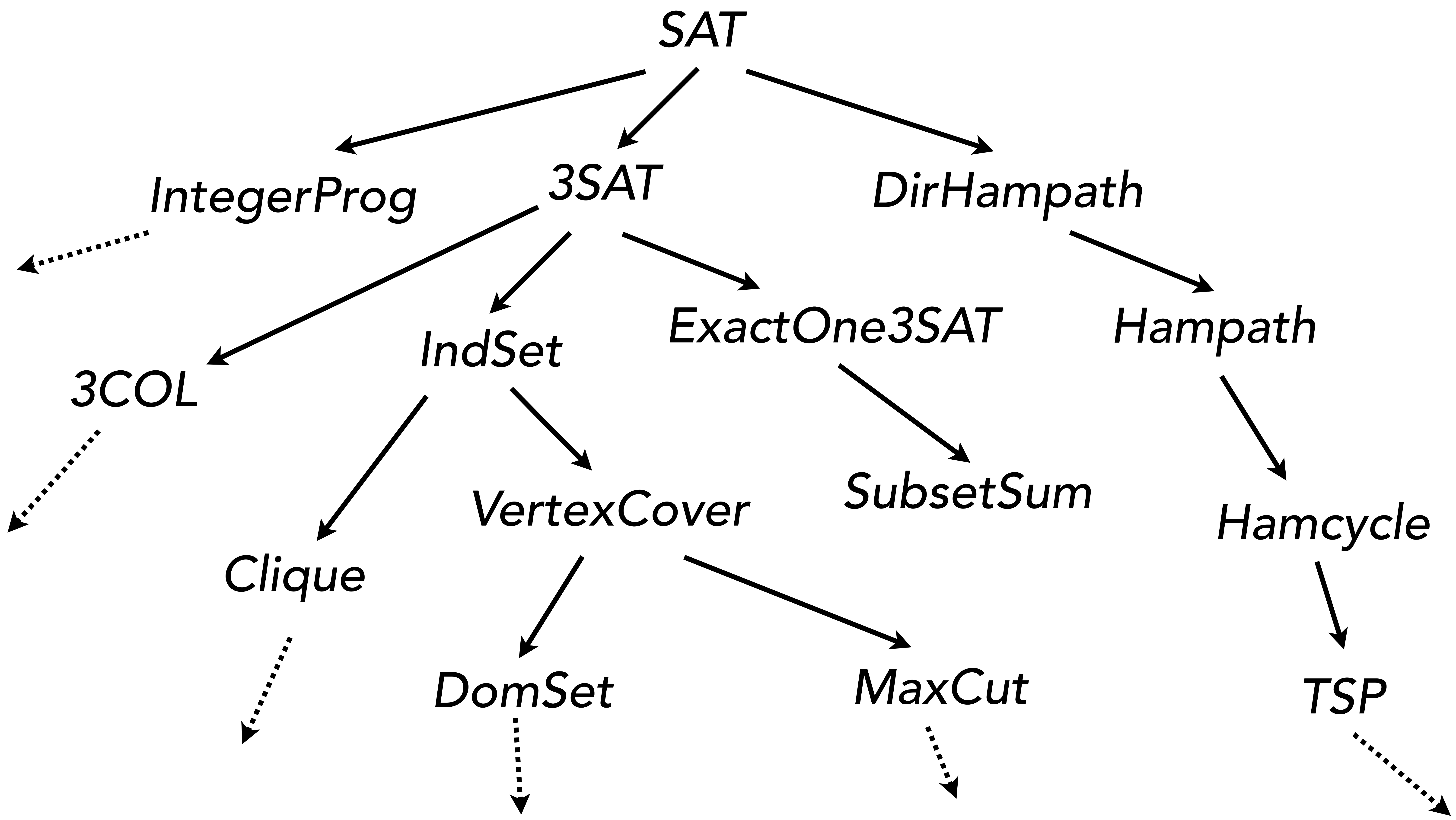
NP-complete **Network**



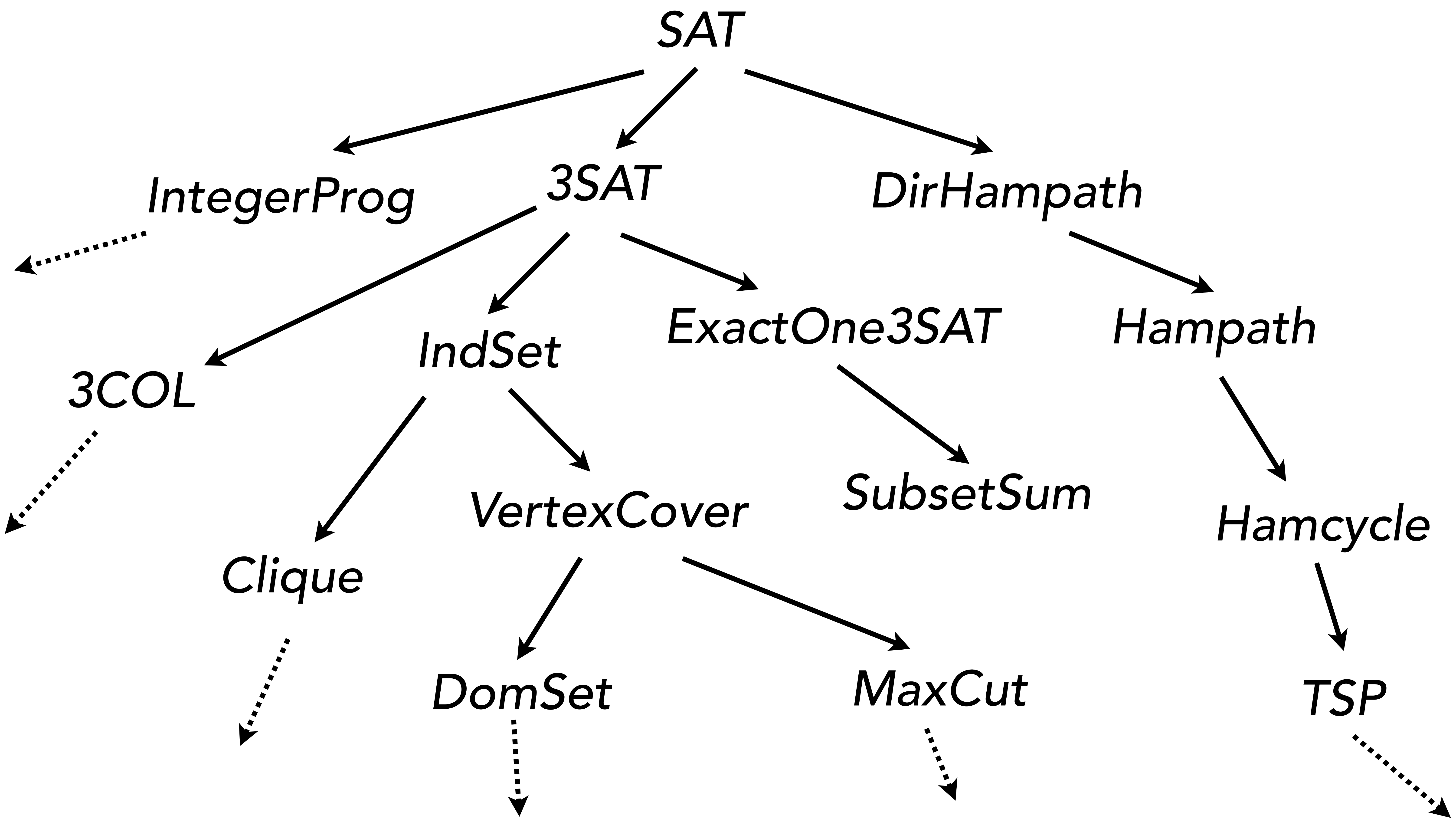
NP-complete **Network**



NP-complete **Network**

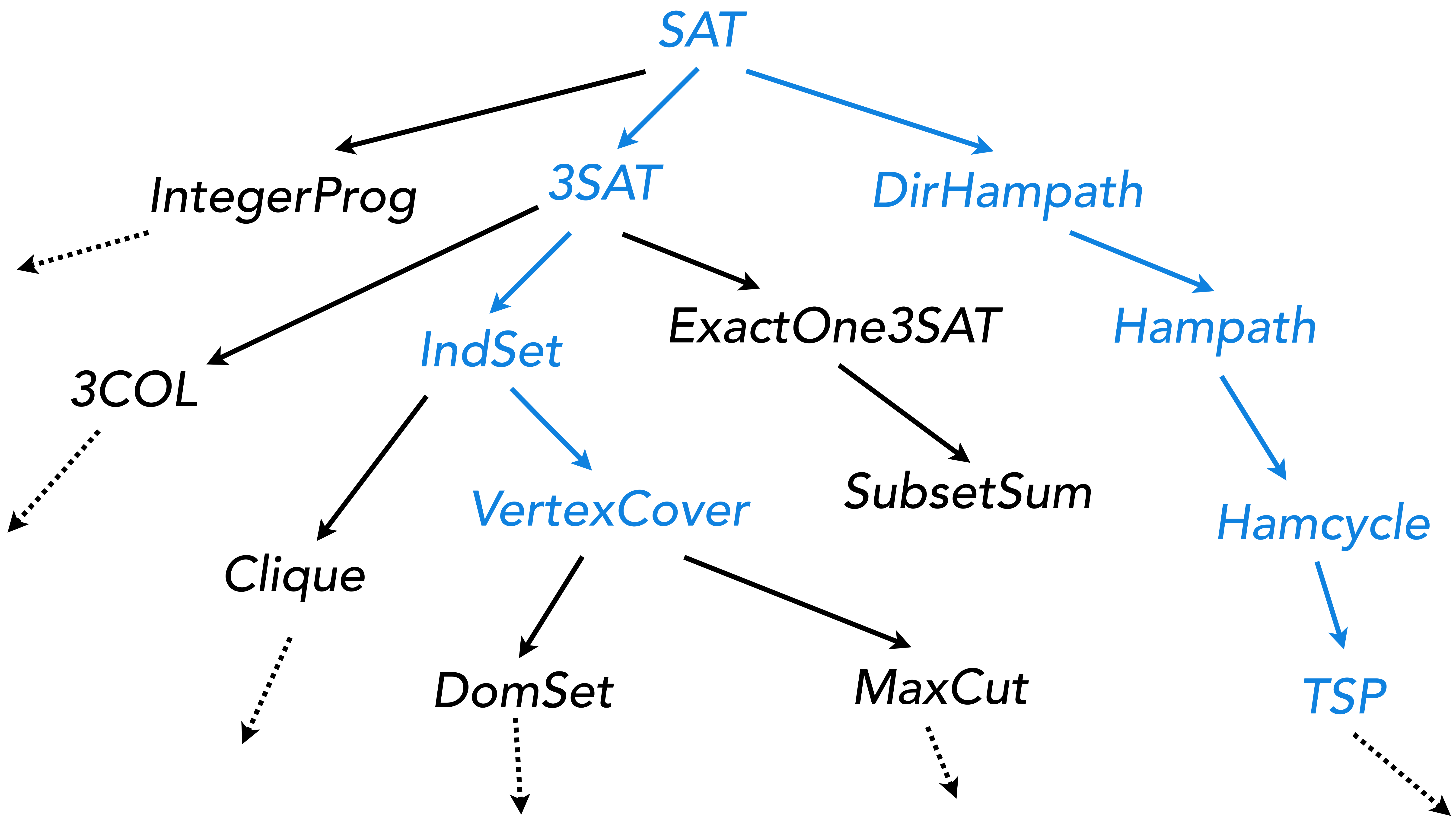


NP-complete **Network**



"In this paper we give theorems that suggest, but do not imply, that these problems, as well as many others, will remain intractable perpetually." – Richard Karp, 1972

NP-complete **Network**



"In this paper we give theorems that suggest, but do not imply, that these problems, as well as many others, will remain intractable perpetually." – Richard Karp, 1972

SAT $\leq_p$ *3SAT*

$SAT \leq_p 3SAT$

Idea:

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \;=$$

$SAT \leq_p$ 3SAT

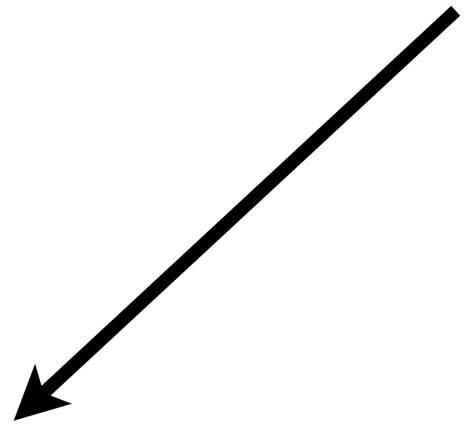
Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$SAT \leq_p 3SAT$

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

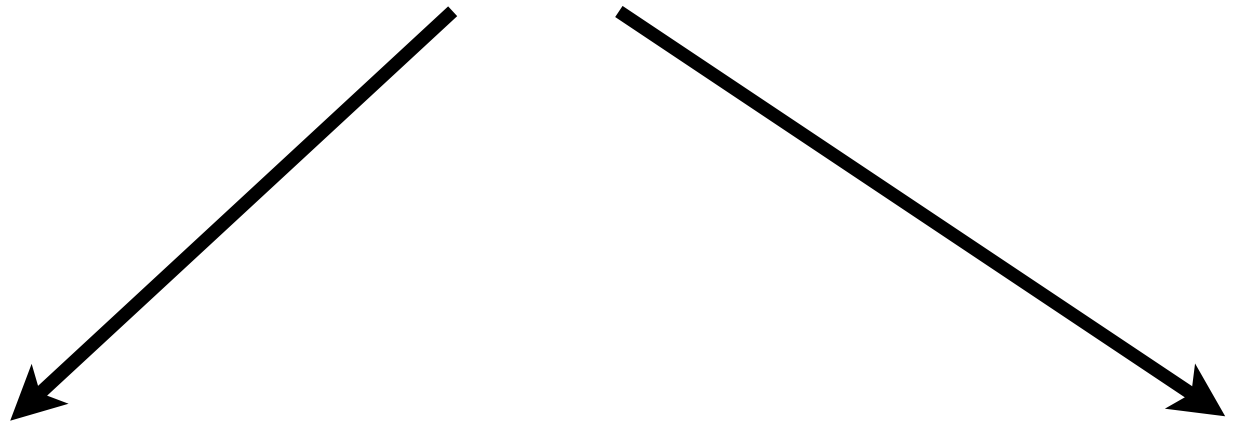
$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$



$SAT \leq_p 3SAT$

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$



$SAT \leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \;=\; (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' \;=\; \qquad \wedge \qquad$$

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \;=\; (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' \;=\; (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge$$

$SAT \leq_p 3SAT$

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' = (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \quad = \quad (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' \quad = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

Time to break a clause of k literals into a 3CNF formula:

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \quad = \quad (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' \quad = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

Time to break a clause of k literals into a 3CNF formula:

- $T(k) =$

$SAT \leq_p 3SAT$

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' = (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

Time to break a clause of k literals into a 3CNF formula:

- $T(k) =$
- $T(3) = c$

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi \;=\; (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' \;=\; (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

Time to break a clause of k literals into a 3CNF formula:

- $T(k) = 2.T(k/2 + 1) + O(k)$
- $T(3) = c$

SAT $\leq_p$ *3SAT*

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into two clauses of almost $k/2$ many literals such that the satisfiability is preserved.

$$\phi = (u_1 \vee u_2 \vee \ldots \vee u_k)$$

$$\phi' = (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \wedge (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

Time to break a clause of k literals into a 3CNF formula:

- $T(k) = 2.T(k/2 + 1) + O(k)$
- $T(3) = c$

Prove that $T(k) = O(k^c)$ *and reduction is polytime.*

Isn't 2*SAT* also NP-Complete?

Isn't 2*SAT* also NP-Complete?

$$(u_1 \vee u_2 \vee u_3)$$

Isn't 2*SAT* also NP-Complete?

$$(u_1 \vee u_2 \vee u_3)$$

$$(u_1 \vee u_2 \vee u) \qquad (u_3 \vee \neg u)$$

Isn't 2*SAT* also NP-Complete?

$$(u_1 \vee u_2 \vee u_3)$$

$$(u_1 \vee u_2 \vee u) \qquad (u_3 \vee \neg u)$$

Further breakdown isn't possible.

$3SAT \leq_p IndSet$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

3SAT $\leq_p$ *IndSet*

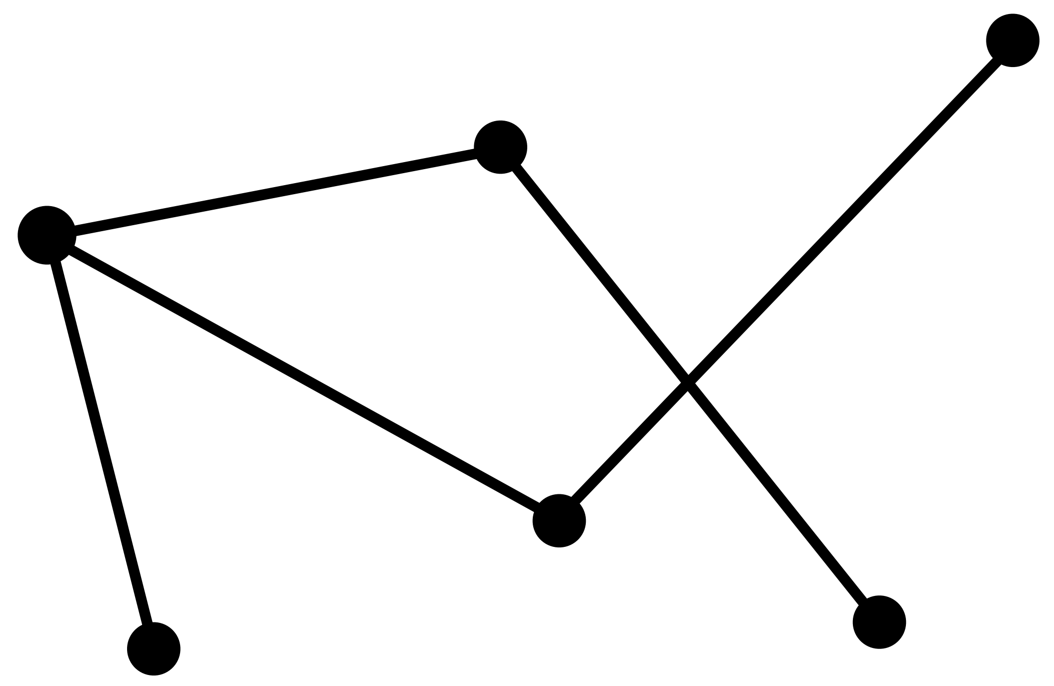
- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

A subset of vertices of G, such that no two of its vertices are adjacent

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

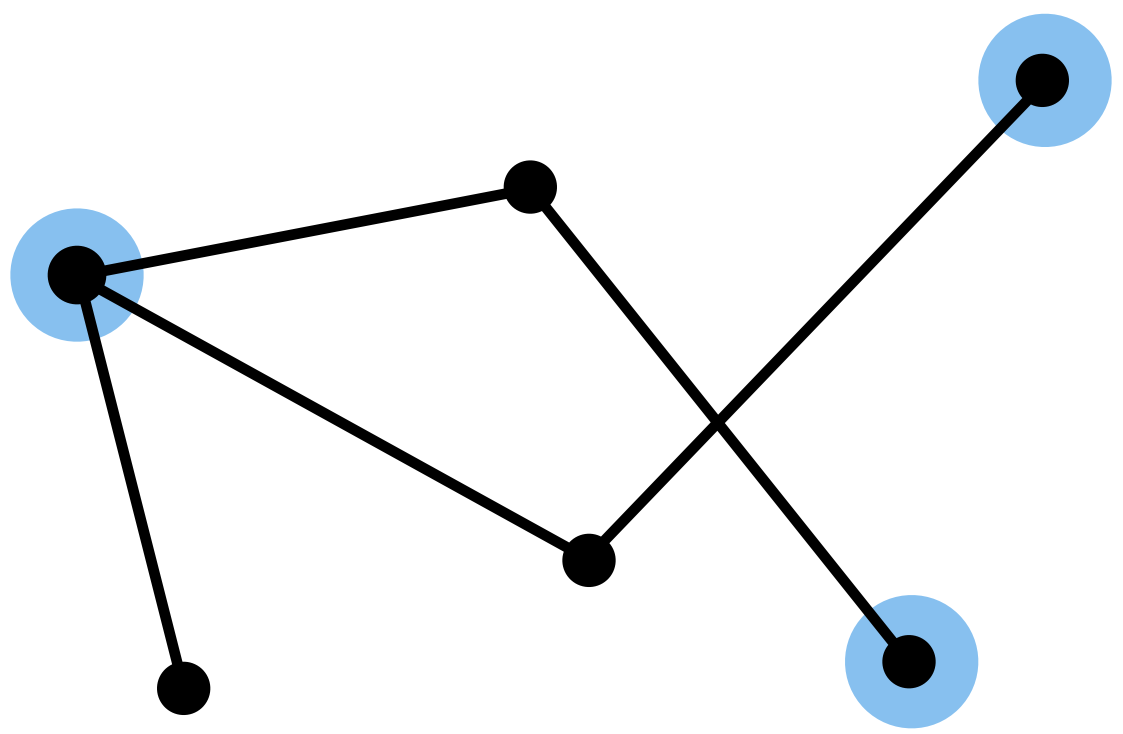
A subset of vertices of G, such that no two of its vertices are adjacent



3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

A subset of vertices of G, such that no two of its vertices are adjacent



Has an independent set of size 3.

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

3SAT $\leq_p$ *IndSet*

- ***3SAT*** = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- ***IndSet*** = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

$$(u_1 \vee u_2 \vee u_3) \implies$$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

v_1: 0 0 1

v_2: 0 1 0

v_3: 0 1 1

$(u_1 \vee u_2 \vee u_3) \implies$ v_4: 1 0 0

v_5: 1 0 1

v_6: 1 1 0

v_7: 1 1 1

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

v_1: 0 0 1

v_2: 0 1 0

v_3: 0 1 1

$(u_1 \vee u_2 \vee u_3) \implies$ v_4: 1 0 0 $(u_1 \vee \overline{u}_3 \vee u_4) \implies$

v_5: 1 0 1

v_6: 1 1 0

v_7: 1 1 1

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

$$(u_1 \vee u_2 \vee u_3) \implies \begin{array}{l} v_1: 0\ 0\ 1 \\ v_2: 0\ 1\ 0 \\ v_3: 0\ 1\ 1 \\ v_4: 1\ 0\ 0 \\ v_5: 1\ 0\ 1 \\ v_6: 1\ 1\ 0 \\ v_7: 1\ 1\ 1 \end{array} \qquad (u_1 \vee \overline{u}_3 \vee u_4) \implies \begin{array}{l} v_8: 0\ 0\ 0 \\ v_9: 0\ 0\ 1 \\ v_{10}: 0\ 1\ 1 \\ v_{11}: 1\ 0\ 0 \\ v_{12}: 1\ 0\ 1 \\ v_{13}: 1\ 1\ 0 \\ v_{14}: 1\ 1\ 1 \end{array}$$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

$(u_1 \vee u_2 \vee u_3) \implies$

v_1: 0 0 1
v_2: 0 1 0
v_3: 0 1 1
v_4: 1 0 0
v_5: 1 0 1
v_6: 1 1 0
v_7: 1 1 1

$(u_1 \vee \overline{u}_3 \vee u_4) \implies$

v_8: 0 0 0
v_9: 0 0 1
v_{10}: 0 1 1
v_{11}: 1 0 0
v_{12}: 1 0 1
v_{13}: 1 1 0
v_{14}: 1 1 1

$\cdots\cdots$

3SAT $\leq_p$ *IndSet*

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.

$(u_1 \vee u_2 \vee u_3) \implies$

v_1: 0 0 1
v_2: 0 1 0
v_3: 0 1 1
v_4: 1 0 0
v_5: 1 0 1
v_6: 1 1 0
v_7: 1 1 1

$(u_1 \vee \overline{u}_3 \vee u_4) \implies$

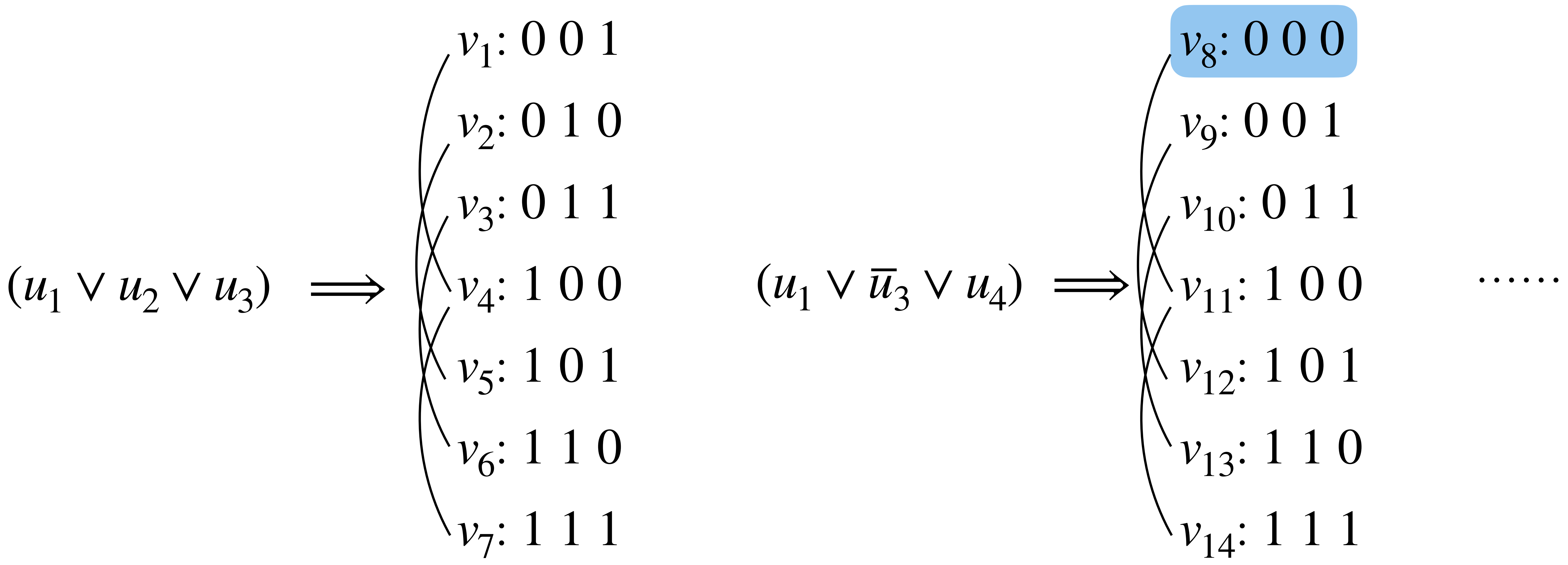
v_8: 0 0 0
v_9: 0 0 1
v_{10}: 0 1 1
v_{11}: 1 0 0
v_{12}: 1 0 1
v_{13}: 1 1 0
v_{14}: 1 1 1

$\cdots\cdots$

$3SAT \leq_p IndSet$

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k \rangle \mid G$ has an independent set of size $k\}$

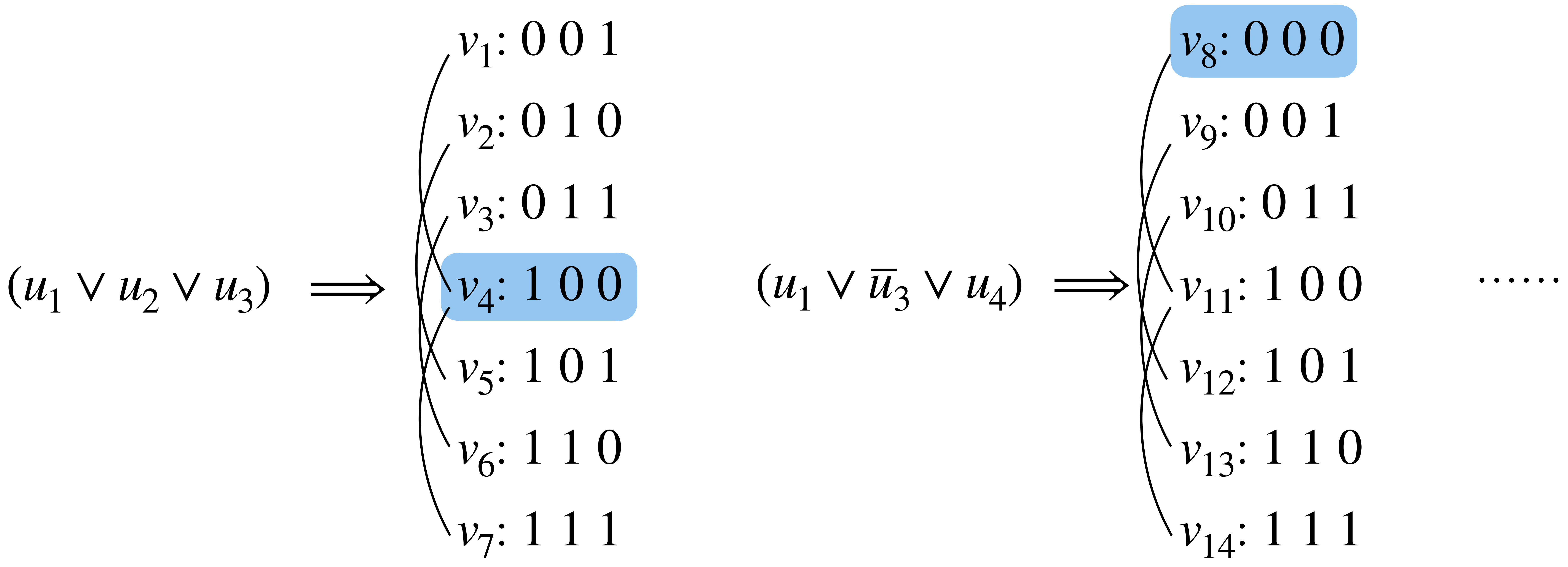
Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.



$3SAT \le_p IndSet$

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

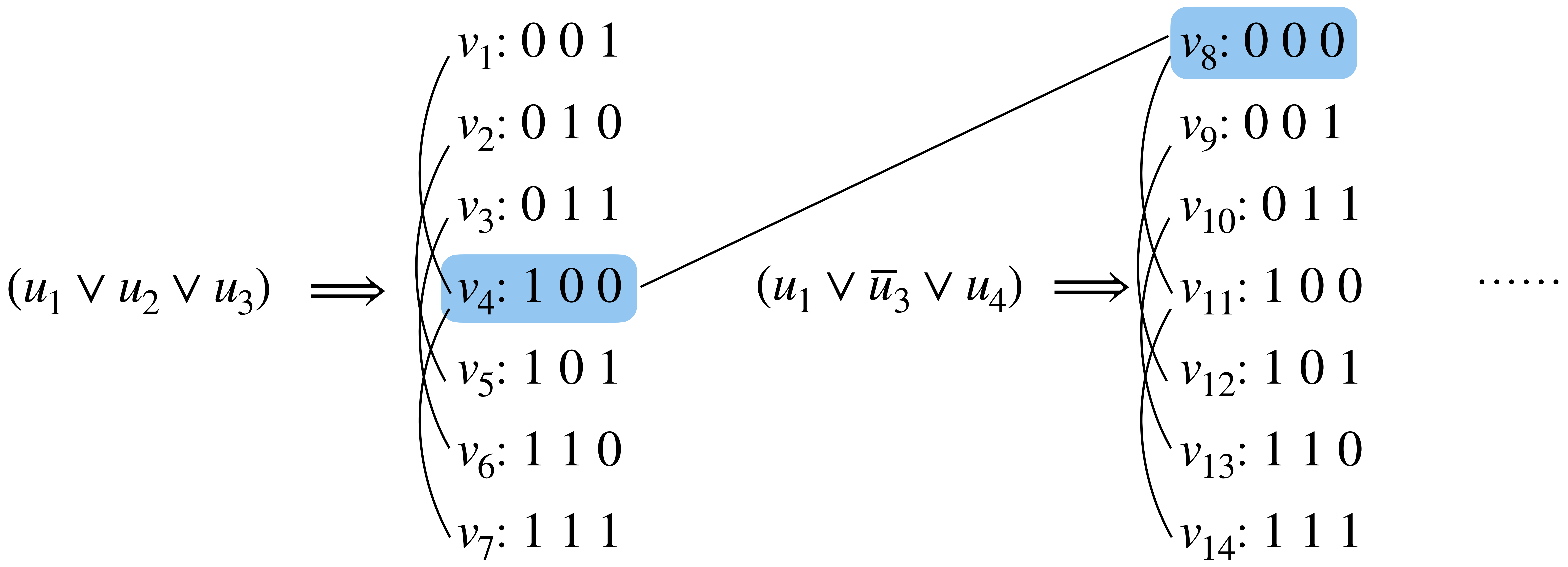
Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.



3SAT $\leq_p$ IndSet

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

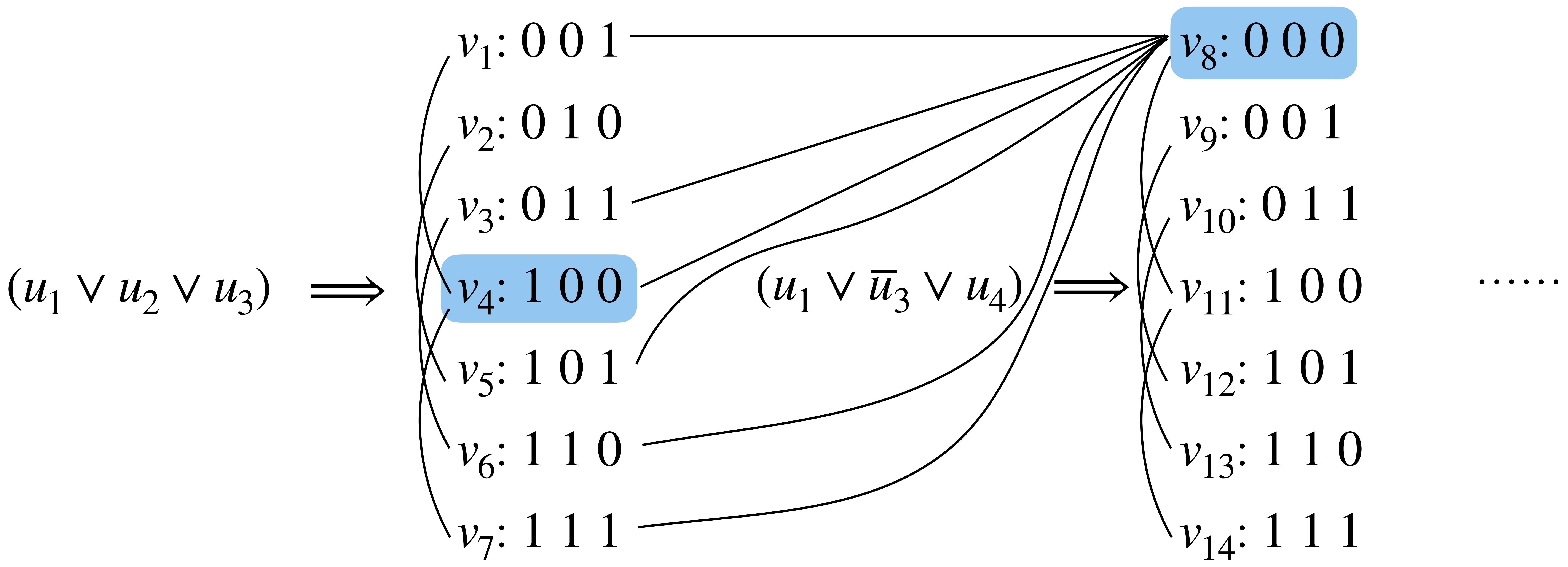
Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.



$3SAT \leq_p IndSet$

- *3SAT* = $\{\phi \mid \phi$ is a satisfiable 3CNF formula$\}$
- *IndSet* = $\{\langle G, k\rangle \mid G$ has an independent set of size $k\}$

Goal: Convert ϕ into (G, k) in polytime, s.t. ϕ is satisfiable iff G has an independent set of size k.



$3SAT \leq_p IndSet$

$3SAT \leq_p IndSet$

$\phi \to \langle G, k \rangle$:

3SAT $\leq_p$ *IndSet*

$\phi \to \langle G, k \rangle$:

- A cluster of 7 vertices ∀ clause of ϕ corresponding to satisfying partial assignments.

$3SAT \leq_p IndSet$

$\phi \rightarrow \langle G, k \rangle$:

- A cluster of 7 vertices ∀ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.

3SAT $\leq_p$ IndSet

$\phi \rightarrow \langle G, k \rangle$:

- A cluster of 7 vertices $\forall$ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.

3SAT $\leq_p$ IndSet

$\phi \rightarrow \langle G, k \rangle$:

- A cluster of 7 vertices ∀ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.
- k = # of clauses in ϕ.

$3SAT \leq_p IndSet$

$\phi \rightarrow \langle G, k \rangle$:

- A cluster of 7 vertices $\forall$ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.
- k = # of clauses in ϕ.

Claim: ϕ is satisfiable iff G has an independent set of size k = # of clauses in ϕ

3SAT $\leq_p$ IndSet

$\phi \to \langle G, k \rangle$:

- A cluster of 7 vertices $\forall$ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.
- k = # of clauses in ϕ.

Claim: ϕ is satisfiable iff G has an independent set of size k = # of clauses in ϕ

Proof: ($\Longrightarrow$) Suppose ϕ has a satisfying assignment u.

$3SAT \leq_p IndSet$

$\phi \to \langle G, k \rangle$:

- A cluster of 7 vertices $\forall$ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.
- k = # of clauses in ϕ.

Claim: ϕ is satisfiable iff G has an independent set of size k = # of clauses in ϕ

Proof: ($\Longrightarrow$) Suppose ϕ has a satisfying assignment u.

Form an independent set S of size k for G:

$3SAT \leq_p IndSet$

$\phi \to \langle G, k \rangle$:

- A cluster of 7 vertices $\forall$ clause of ϕ corresponding to satisfying partial assignments.
- An edge between every pair of vertices in the same cluster.
- An edge between two vertices of different clusters, if they correspond to inconsistent partial assignments.
- k = # of clauses in ϕ.

Claim: ϕ is satisfiable iff G has an independent set of size k = # of clauses in ϕ

Proof: ($\Longrightarrow$) Suppose ϕ has a satisfying assignment u.

Form an independent set S of size k for G:

By picking a vertex from every cluster whose values matches to that of u.