Minimum Vertex Cover, Matching and Graph 3-Coloring

CSCI3360 Introduction to Computational Complexity
Tutorial (Week 8)

Chak Fai Yung

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Today's tutorial

1. **Section 5.2.2: Matching and vertex cover.** Prove Kőnig's theorem for **bipartite graphs** and construct a minimum cover.
2. **Section 5.3: Graph 3-coloring.** Build and verify the reduction from 3SAT using not-all-equal constraints.

Throughout: prove optimality with matching bounds and verify both directions of each reduction.

Minimum vertex cover: the quantity we want to minimize

Recall: a vertex cover $C \subseteq V$ contains at least one endpoint of every edge.

Word notation

$$\text{MinVertexCoverSize}(G)$$

is the smallest number of vertices in any vertex cover of G.

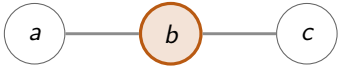
- An actual cover C gives an **upper bound**:

$$\text{MinVertexCoverSize}(G) \leq |C|.$$

- To prove that C is minimum, we also need a **lower bound** of $|C|$.

A matching will provide that lower bound.

A cover and a proof that it is smallest



$C = \{b\}$

Orange fill means selected for the cover.

1. $C = \{b\}$ covers both edges, so

$$\text{MinVertexCoverSize}(G) \leq 1.$$

2. The graph has an edge. A cover with zero vertices leaves it uncovered, so

$$\text{MinVertexCoverSize}(G) \geq 1.$$

Both bounds give $\text{MinVertexCoverSize}(G) = 1$.

Matching and maximum matching

A **matching** $M \subseteq E$ consists of edges with no shared endpoints.

Definition 5.2.3

The decision version of Maximum Matching asks whether G has a matching with at least k edges.

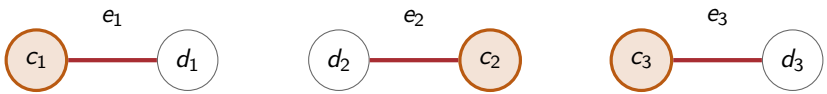
Write MaxMatchingSize(G) for the largest number of edges in any matching of G.

- A vertex is **matched** if it is an endpoint of an edge in M.
- A matched vertex has exactly one **matching partner**.
- A **maximum** matching has largest size. A **maximal** matching merely admits no extra edge.

Maximum Matching is in P, in both bipartite and general graphs.

Every matching forces distinct choices in a cover

Let $M = \{e_1, \ldots, e_r\}$ be a matching and let C be any vertex cover.



1. Every matching edge e_i needs an endpoint in C. Choose one such endpoint c_i.
2. Matching edges have no endpoints in common. Thus $c_1, \ldots, c_r$ are distinct vertices of C.
3. Hence every cover has at least $r = |M|$ vertices:

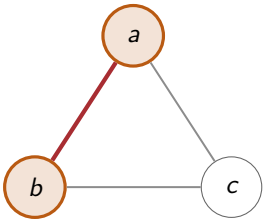
$$\boxed{|M| \leq |C|.}$$

Why the bipartite assumption matters

Apply $|M| \leq |C|$ to a maximum matching and a minimum cover:

$$\text{MaxMatchingSize}(G) \leq \text{MinVertexCoverSize}(G).$$

Equality is not guaranteed for general graphs.



The nonbipartite triangle K_3:

- Every two edges share a vertex, so the largest matching has one edge.
- One selected vertex leaves its opposite edge uncovered; two selected vertices cover all edges.

$$\text{MaxMatchingSize}(K_3) = 1,$$
$$\text{MinVertexCoverSize}(K_3) = 2.$$

Kőnig's theorem for bipartite graphs

A graph is **bipartite** if $V = L \dot{\cup} R$ and every edge has one endpoint in each part.

Theorem 5.2.2: bipartite graphs

Assume G is bipartite. Then

$$\boxed{\text{MaxMatchingSize}(G) = \text{MinVertexCoverSize}(G).}$$

The lower bound holds for every graph. For a **bipartite** graph, we prove the reverse inequality as follows:

1. Fix a maximum matching M, so $|M| = \text{MaxMatchingSize}(G)$.
2. Construct a vertex cover C with $|C| = |M|$.
3. Since C is a cover, $\text{MinVertexCoverSize}(G) \leq |C| = |M|$. This proves the reverse inequality and hence equality.

Bipartite matching: recall from Week 3

Results from Week 3, Section 2.2.3

- A maximum matching in a bipartite graph can be found in polynomial time.
- **Berge's theorem:** a matching is maximum if and only if it has no augmenting path.

An augmenting path alternates between edges outside and inside the matching and has unmatched endpoints.

Refer to the Week 3 slides for the matching algorithm, its correctness proof, and running-time analysis.

For Kőnig's theorem, fix a maximum matching M. We use the fact that M has no augmenting path to construct a minimum vertex cover.

Marked sets for the minimum-cover construction

Fix a maximum matching M in a **bipartite graph** $G = (L \dot\cup R, E)$.

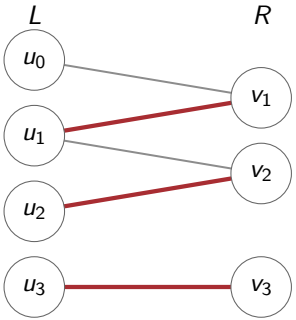
1. Initially mark **every unmatched vertex in** L.
2. From a marked $u \in L$, follow every edge $uv \notin M$ to mark $v \in R$.
3. From a marked $v \in R$, follow its matching edge, if any, to mark its partner in L.
4. Continue until no new vertex can be marked. Call the marked sets Z_L, Z_R.

Paths witnessing reachability

Record a predecessor when a vertex is first marked. Following predecessors backward gives a simple alternating path from an unmatched left vertex.

Equivalently, orient nonmatching edges $L \to R$ and matching edges $R \to L$, then run a directed search from all unmatched left vertices.

Running example: begin with a maximum matching



The red edges form

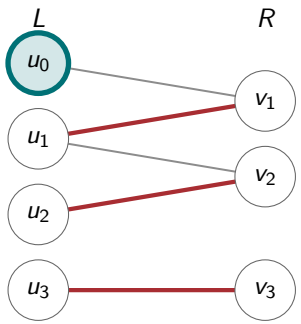
$$M = \{u_1v_1, u_2v_2, u_3v_3\}.$$

They have distinct endpoints, so M is a matching of size three.

Every matching uses distinct vertices of R. Since $|R| = 3$, no matching can be larger. Thus M is maximum.

We now use this given M to construct a minimum cover.

Marking step 1: start from unmatched left vertices



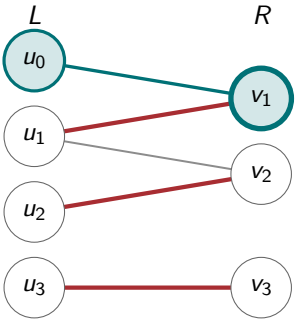
The initial roots are the unmatched vertices in L.

Here u_0 is unmatched. Each of u_1, u_2, u_3 belongs to a red matching edge, so none is an initial root.

$$Z_L = \{u_0\}, \qquad Z_R = \varnothing.$$

Teal circles denote vertices reached so far. A thicker border indicates the vertex newly marked on this frame.

Marking step 2: follow an edge outside the matching



From a marked left vertex, follow edges **outside** M to the right.

Since $u_0v_1 \notin M$, we may follow

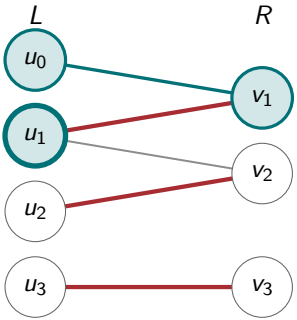
$$u_0 \longrightarrow v_1.$$

Therefore mark v_1:

$$Z_L = \{u_0\}, \qquad Z_R = \{v_1\}.$$

The teal arrow records this permitted exploration move.

Marking step 3: follow the unique matching partner



The marked right vertex v_1 has matching partner u_1.

Follow the red matching edge **from right to left**:

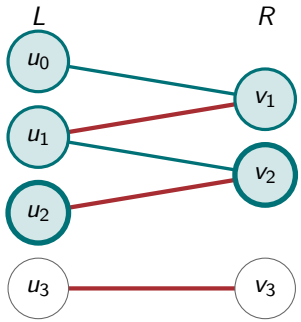
$$v_1 \longrightarrow u_1.$$

Now

$$Z_L = \{u_0, u_1\}, \qquad Z_R = \{v_1\}.$$

The witness path $u_0 \to v_1 \to u_1$ alternates between a nonmatching edge and a matching edge.

Marking step 4: continue until no vertex can be added



From u_1, follow $u_1v_2 \notin M$ to mark v_2. Then follow its matching edge to mark u_2.

$$u_0 \to v_1 \to u_1 \to v_2 \to u_2.$$

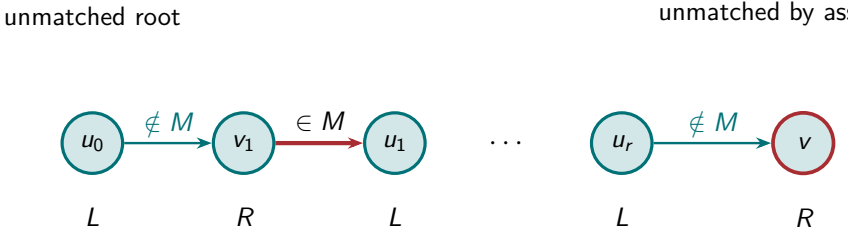
At u_2 no eligible move remains. The component u_3v_3 is unreachable from the root.

$$Z_L = \{u_0, u_1, u_2\},$$
$$Z_R = \{v_1, v_2\}.$$

These are the final reached sets.

A marked unmatched right vertex would be impossible

Suppose an unmatched $v \in R$ were marked. Its predecessor chain would give a path of the following form:

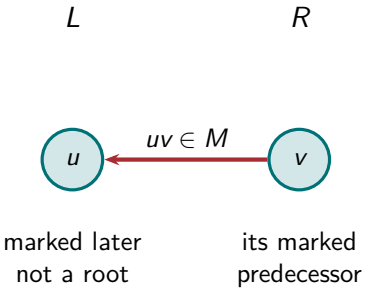


1. The path starts at an unmatched left root and ends at the unmatched right vertex v.
2. Its edges alternate outside and inside M, starting and ending outside M.
3. It is an augmenting path, contradicting Berge's theorem from Week 3 because M is maximum.

Therefore **every marked right vertex is matched**.

Matching partners ($\Rightarrow$): a marked left endpoint

Fix $uv \in M$, with $u \in L$ and $v \in R$. Assume $u \in Z_L$.



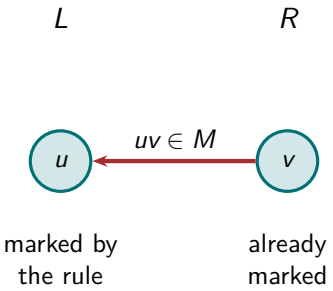
1. Since $uv \in M$, u is matched. Initial roots are unmatched, so u was not marked initially.
2. Every later discovery of a left vertex follows a matching edge from a marked right vertex.
3. The matching partner is unique. Thus the edge used to discover u must be $v \to u$.

Therefore v was already marked, and $v \in Z_R$.

$$\boxed{u \in Z_L \implies v \in Z_R.}$$

Matching partners ($\Leftarrow$): a marked right endpoint

Again fix $uv \in M$. Now assume $v \in Z_R$.



1. The marked vertex v has matching partner u.
2. The right-to-left rule follows $v \to u$.
3. The exploration stops only when no new mark is possible, so u must belong to the final set Z_L.

This proves the reverse implication.

Together, for every $uv \in M$,

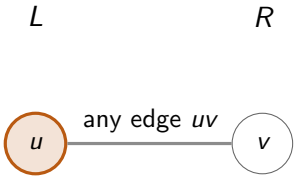
$$\boxed{u \in Z_L \iff v \in Z_R.}$$

The cover rule: an unmarked left endpoint covers its edge

Define

$$\boxed{C = (L \setminus Z_L) \cup Z_R.}$$

Select **unmarked left vertices** and **marked right vertices**.



Take any edge uv, with $u \in L$, $v \in R$.

Case 1: $u \notin Z_L$

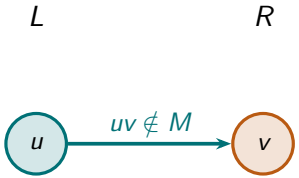
Since u is an unmarked left vertex,

$$u \in L \setminus Z_L \subseteq C.$$

So uv is covered by u, regardless of whether v is marked.

A marked left endpoint: the nonmatching-edge case

Take $uv \in E$, where $u \in Z_L$ and $uv \notin M$.



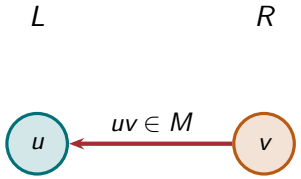
1. The left endpoint u is marked.
2. Since uv is outside M, the left-to-right rule must follow it and mark v.
3. Therefore $v \in Z_R \subseteq C$.

The right endpoint covers the edge, even though the left endpoint is not selected.

Here the nonmatching edge itself explains why the right endpoint is marked.

A marked left endpoint: the matching-edge case

Take $uv \in M$, where $u \in Z_L$.



This edge is oriented **right to left**, so we do not use the preceding left-to-right rule.

1. The matching-partner implication gives $u \in Z_L \Rightarrow v \in Z_R$.
2. Thus $v \in Z_R \subseteq C$, and the right endpoint covers uv.

Every edge is in Case 1 or one of these two marked-left cases. Therefore C covers every edge.

Counting: exactly one endpoint from each matching edge

For $uv \in M$, the partner equivalence says both endpoints are marked or both are unmarked.

Both marked

$u \in Z_L$ $\quad$ $v \in Z_R$



not selected $\quad$ selected

Only v belongs to C: marked right vertices are selected, marked left vertices are excluded.

Both unmarked

$u \notin Z_L$ $\quad$ $v \notin Z_R$



selected $\quad$ not selected

Only u belongs to C: unmarked left vertices are selected, unmarked right vertices are excluded.

Distinct matching edges share no endpoints. These choices therefore give exactly $|M|$ distinct selected **matched vertices**.

Counting: unmatched vertices never enter the cover

Consider any unmatched vertex; other graph edges are omitted in these schematic fragments.

unmatched in L



initial root
marked, not selected

unmatched in R



cannot be reached
unmarked, not selected

Unmatched left vertices are roots, so they are marked. Thus $u \in Z_L$ and $u \notin C$.

By Property 1, an unmatched right vertex cannot be marked. Thus $v \notin Z_R$ and $v \notin C$.

There are no selected unmatched vertices. Together with one selected endpoint per matching edge,

$$\boxed{|C| = |M|.}$$

Kőnig's equality for bipartite graphs

For our **bipartite graph** G, we constructed a cover C from a maximum matching M and proved

$$|C| = |M| = \text{MaxMatchingSize}(G).$$

1. The general matching lower bound says that every cover is at least this large:

$$\text{MaxMatchingSize}(G) \leq \text{MinVertexCoverSize}(G).$$

2. Our actual cover supplies the opposite bound:

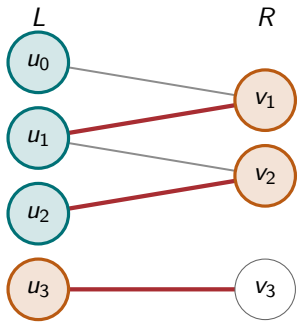
$$\text{MinVertexCoverSize}(G) \leq |C| = \text{MaxMatchingSize}(G).$$

Together they give Kőnig's equality for **bipartite graphs**:

$$\boxed{\text{MaxMatchingSize}(G) = \text{MinVertexCoverSize}(G).}$$

Since C attains this value, C is a minimum vertex cover.

From the reached sets to the selected cover



Select unmarked vertices on the left and marked vertices on the right:

$$C = (L \setminus Z_L) \cup Z_R = \{u_3, v_1, v_2\}.$$

Orange circles are the vertices selected for C.

Matching edge	Its selected endpoint
u_1v_1	v_1
u_2v_2	v_2
u_3v_3	u_3

Every edge is covered, and $|C| = 3 = |M|$.

A polynomial-time algorithm for bipartite vertex cover

For a bipartite graph $G = (L \dot{\cup} R, E)$:

1. Compute a maximum matching M.
   Refer to Week 3 slides, Section 2.2.3, for the matching algorithm and its proof.
2. Compute Z_L, Z_R using the alternating exploration in Kőnig's proof.
3. Return $C = (L \setminus Z_L) \cup Z_R$.

Once M is known, computing the marked sets and extracting C take $O(|V| + |E|)$ time.

Complexity consequence

The matching algorithm from Week 3 runs in polynomial time, and the cover extraction is linear. Thus Bipartite Vertex Cover is in P.

The returned cover is minimum by the coverage and counting arguments above.

Graph 3-coloring

A **proper 3-coloring** assigns each vertex one of three colors so that adjacent vertices receive different colors.

$$\mathrm{G3C} = \{\langle G \rangle : G \text{ admits a proper coloring with at most three colors}\}.$$

Theorem 5.3.1

$\mathrm{G3C}$ is NP-complete.

We prove two claims:

1. $\mathrm{G3C} \in \mathrm{NP}$: a coloring has a polynomial-time verifier.
2. $\mathrm{G3C}$ is NP-hard: construct $3\mathrm{SAT} \leq_P \mathrm{G3C}$.

Two-colorability is bipartiteness and has a linear-time test.

Graph 3-coloring belongs to NP

For input $G = (V, E)$, the certificate is a function

$$c : V \longrightarrow \{1, 2, 3\}.$$

The verifier performs two checks:

1. Every vertex has exactly one listed color in $\{1, 2, 3\}$.
2. For every edge $uv \in E$, verify that $c(u) \neq c(v)$.

If G is 3-colorable, a proper coloring passes all checks.

If the verifier accepts, its checked function c is a proper 3-coloring by definition.

The certificate has $O(|V|)$ color entries, and verification takes $O(|V| + |E|)$ time.

The structure of the reduction

Given a 3-CNF formula φ, build a graph G_φ in three layers.

1. **Palette triangle:** establish color names $\mathsf{T}, \mathsf{F}, \mathsf{B}$.
2. **Variable gadgets:** encode complementary Boolean values for each variable and its negation.
3. **Constraint gadgets:** enforce not-all-equal relations that encode each original OR clause.

The required equivalence is

$$\boxed{\varphi \text{ is satisfiable} \iff G_\varphi \text{ is 3-colorable.}}$$

Not-all-equal constraints

For Boolean inputs, $\mathrm{NAE}(a, b, c)$ means that their values are **not all equal**.

a	b	c	$\mathrm{NAE}(a, b, c)$
F	F	F	false
T	T	T	false
Any mixture of T and F			true

- A triple must contain at least one true and at least one false input.
- OR and NAE are different: $T \vee T \vee T$ is true, but $\mathrm{NAE}(T, T, T)$ is false.

Replace one OR clause by two NAE constraints

For each clause $(\ell_1 \vee \ell_2 \vee \ell_3)$, introduce a **fresh** variable y and require

$$\mathrm{NAE}(\ell_1, \ell_2, y), \qquad \mathrm{NAE}(\neg y, \ell_3, \mathsf{F}).$$

Here F is a fixed false constant.

Clause equivalence

$$\ell_1 \vee \ell_2 \vee \ell_3 \quad \Longleftrightarrow \quad \exists y \, [\mathrm{NAE}(\ell_1, \ell_2, y) \wedge \mathrm{NAE}(\neg y, \ell_3, \mathsf{F})].$$

The value of y may depend on the clause. Different clauses receive different auxiliary variables.

Clause equivalence ($\Rightarrow$): first two literals

Assume $\ell_1 \vee \ell_2 \vee \ell_3$ is true. We must **construct a value of** y satisfying both NAE constraints.

Case 1: at least one of ℓ_1, ℓ_2 is true

Choose $y = \mathsf{F}$, so $\neg y = \mathsf{T}$.

1. The triple (ℓ_1, ℓ_2, y) contains a true input by the case assumption and a false input y. Hence $\mathrm{NAE}(\ell_1, \ell_2, y)$ holds.
2. The triple $(\neg y, \ell_3, \mathsf{F})$ contains $\neg y = \mathsf{T}$ and the fixed false input. Hence $\mathrm{NAE}(\neg y, \ell_3, \mathsf{F})$ holds, whatever ℓ_3 is.

Thus this choice of y satisfies both constraints.

Clause equivalence ($\Rightarrow$): only the third literal

Continue to assume $\ell_1 \vee \ell_2 \vee \ell_3$ is true.

Case 2: $\ell_1 = \ell_2 = \mathsf{F}$

The true OR clause forces $\ell_3 = \mathsf{T}$. Choose $y = \mathsf{T}$, so $\neg y = \mathsf{F}$.

The two input triples are now

$$(\ell_1, \ell_2, y) = (\mathsf{F}, \mathsf{F}, \mathsf{T}), \qquad (\neg y, \ell_3, \mathsf{F}) = (\mathsf{F}, \mathsf{T}, \mathsf{F}).$$

Each triple contains both Boolean values, so both NAE constraints hold.

Cases 1 and 2 cover every satisfying assignment to the original clause. Therefore

$$\ell_1 \vee \ell_2 \vee \ell_3 \implies \exists y \, [\mathrm{NAE}(\ell_1, \ell_2, y) \wedge \mathrm{NAE}(\neg y, \ell_3, \mathsf{F})].$$

Clause equivalence ($\Leftarrow$): the constraints force OR

Assume some value of y satisfies

$$\mathrm{NAE}(\ell_1, \ell_2, y) \quad \text{and} \quad \mathrm{NAE}(\neg y, \ell_3, \mathsf{F}).$$

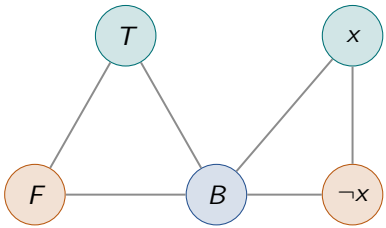
1. If ℓ_1 or ℓ_2 is true, then the original OR clause is already true.
2. Otherwise $\ell_1 = \ell_2 = \mathsf{F}$. The first constraint forbids $(\mathsf{F}, \mathsf{F}, \mathsf{F})$, so $y = \mathsf{T}$.
3. Hence $\neg y = \mathsf{F}$. The second constraint becomes

$$\mathrm{NAE}(\mathsf{F}, \ell_3, \mathsf{F}),$$

which forces $\ell_3 = \mathsf{T}$.

In every case, $\ell_1 \vee \ell_2 \vee \ell_3$ is true. This proves the reverse implication.

The palette triangle and variable gadget



- The triangle T, F, B uses all three colors. Name the colors after these vertices.
- Join x and $\neg x$ to each other and both to B.
- Neither can use color B. They must receive opposite colors T, F.

Do this for every original variable and every auxiliary y.

Variable gadget ($\Rightarrow$): a coloring gives truth values

Assume the palette and variable gadget have a proper 3-coloring.

1. T, F, B are pairwise adjacent. Their colors are distinct, so name these three colors $\mathrm{T}, \mathrm{F}, \mathrm{B}$.
2. Both x and $\neg x$ are adjacent to B. Therefore neither uses B.
3. The edge $x\neg x$ forces distinct colors. Only T, F remain, so one literal receives each color.

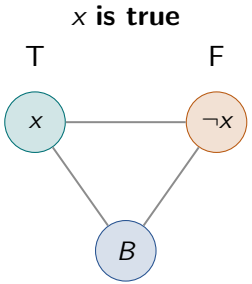
Define the Boolean value of x by

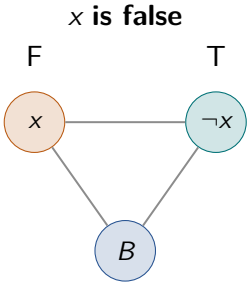
$$x = \text{true} \quad \Longleftrightarrow \quad c(x) = \mathrm{T}.$$

Then the color of $\neg x$ agrees with the Boolean negation of this value.

Variable gadget ($\Leftarrow$): both truth values extend

Fix the three palette color names T, F, B.

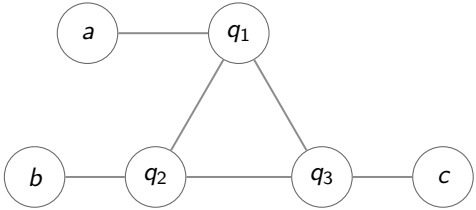




- The two edges to B forbid color B at both literal vertices.
- The edge $x\neg x$ forces the remaining two colors to be different.

These two possibilities are precisely the two truth values of x.

The gadget for $\mathrm{NAE}(a, b, c)$



1. Add three fresh vertices q_1, q_2, q_3 forming a triangle.
2. Add the edges q_1a, q_2b, q_3c.

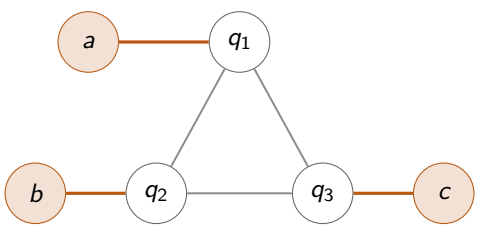
The inputs already have colors in $\{\mathrm{T}, \mathrm{F}\}$.

Claim: the input coloring extends to this triangle exactly when $\mathrm{NAE}(a, b, c)$ holds.

NAE gadget ($\Rightarrow$): all-false inputs forbid one color

Assume the input coloring extends to a proper coloring of the gadget.

$q_1, q_2, q_3 \in \{\mathrm{T}, \mathrm{B}\}$



Inputs: $(\mathrm{F}, \mathrm{F}, \mathrm{F})$

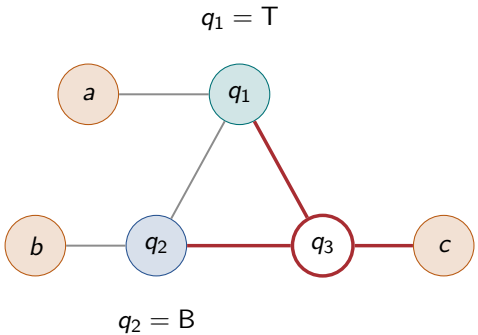
Suppose a, b, c all have color F.

1. Each q_i is adjacent to a false-colored input, so q_i cannot use F.
2. The helper vertices form a triangle, so all three must have different colors.
3. Only two colors remain: T and B.

This is impossible. Thus all-false inputs cannot admit a proper extension.

The all-false case: the last helper has no legal color

The first two helpers are adjacent and can use only T, B. Up to exchanging these two colors, let $q_1 = \mathsf{T}$ and $q_2 = \mathsf{B}$.



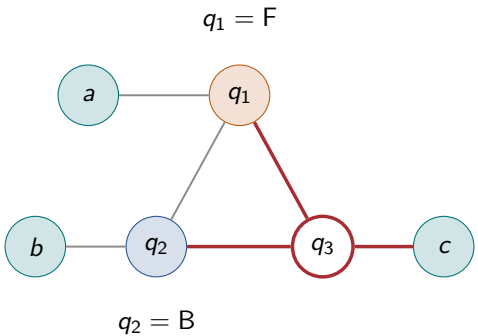
Every possible color for q_3 causes a conflict:

Color of q_3	Conflicting neighbor
T	$q_1 = \mathsf{T}$
B	$q_2 = \mathsf{B}$
F	$c = \mathsf{F}$

The three red edges exclude all three colors at q_3.

The all-true case gives the same obstruction

If a, b, c all have color T, every helper must avoid T.



Only F, B remain for the helpers. Once q_1, q_2 receive these two colors, q_3 cannot use:

- F, because of q_1;
- B, because of q_2;
- T, because of input c.

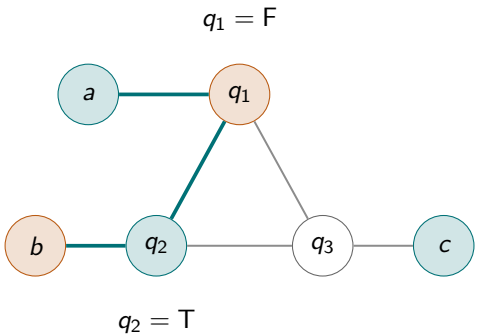
So all-true inputs also have no extension.

A proper extension therefore rules out both all-equal cases:

$$\boxed{\text{proper extension} \implies \mathrm{NAE}(a, b, c).}$$

NAE gadget ($\Leftarrow$): first color two helpers

Assume $\mathrm{NAE}(a, b, c)$ holds. At least one input is true and at least one is false. Choose one of each; the picture illustrates $a = \mathsf{T}$, $b = \mathsf{F}$.



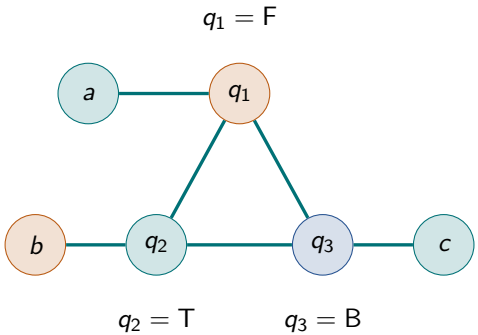
1. Color the helper adjacent to the chosen true input F.
2. Color the helper adjacent to the chosen false input T.

The two attachment edges are proper because their endpoints have opposite Boolean colors.

The edge between these two helpers is also proper: $\mathsf{F} \neq \mathsf{T}$.

NAE gadget ($\Leftarrow$): color the last helper B

Give the remaining helper color B.



1. The helpers use F, T, B, so all three triangle edges are proper.
2. The first two attachment edges join opposite Boolean colors.
3. The remaining input has color T or F, so it differs from its helper's color B.

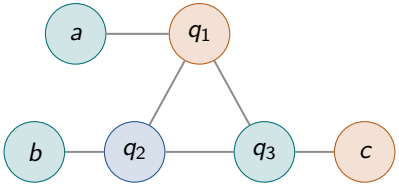
All six gadget edges are therefore properly colored.

$\mathrm{NAE}(a, b, c) \implies$ proper extension exists.

The construction works for both types of mixed inputs

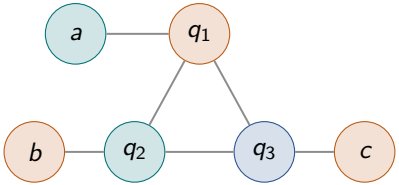
Every mixed Boolean triple has either two true inputs or two false inputs.

Inputs $(\mathsf{T}, \mathsf{T}, \mathsf{F})$



Helpers: $(q_1, q_2, q_3) = (\mathsf{F}, \mathsf{B}, \mathsf{T})$.

Inputs $(\mathsf{T}, \mathsf{F}, \mathsf{F})$



Helpers: $(q_1, q_2, q_3) = (\mathsf{F}, \mathsf{T}, \mathsf{B})$.

Permuting the three input positions and their attached helpers covers all six mixed triples. The same rule always gives a valid extension.

The complete construction for one clause

For clause $C_j = (\ell_{j1} \vee \ell_{j2} \vee \ell_{j3})$, add:

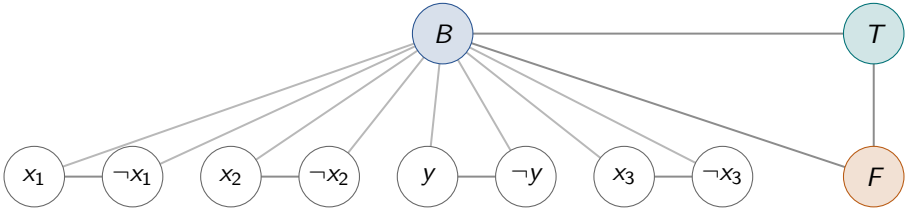
1. A fresh auxiliary variable pair $y_j, \neg y_j$, with its variable gadget.
2. One fresh triangle for $\mathrm{NAE}(\ell_{j1}, \ell_{j2}, y_j)$.
3. One fresh triangle for $\mathrm{NAE}(\neg y_j, \ell_{j3}, \mathsf{F})$.

Triangle	Its three existing input vertices	New vertices
First	$\ell_{j1}, \ell_{j2}, y_j$	q_{j1}, q_{j2}, q_{j3}
Second	$\neg y_j, \ell_{j3}, F$	r_{j1}, r_{j2}, r_{j3}

Different constraints share input vertices when needed, but their triangle vertices are always distinct.

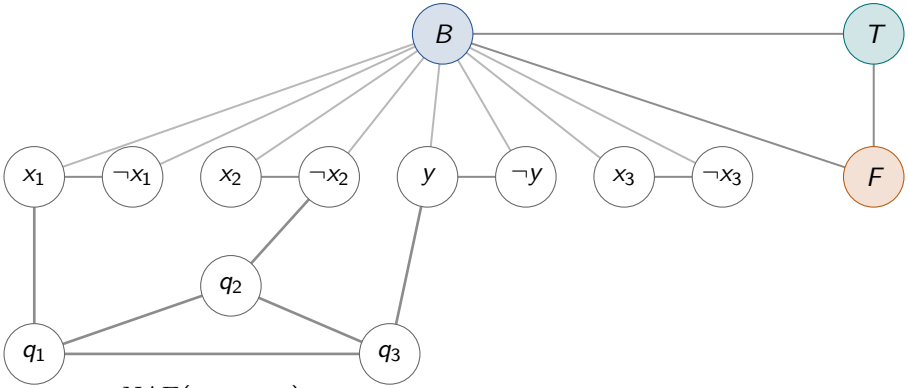
One-clause graph: palette and variable pairs

For $C = (x_1 \vee \neg x_2 \vee x_3)$, create the palette and four literal pairs, including the fresh pair $y, \neg y$.



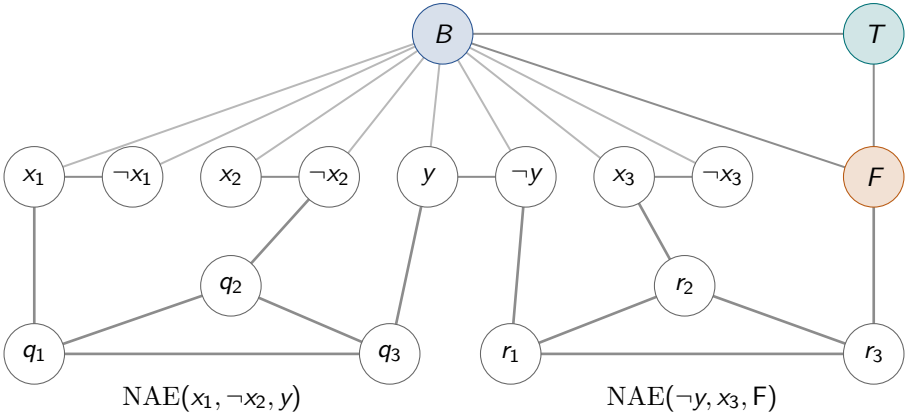
One-clause graph: add the first NAE triangle

Attach q_1, q_2, q_3 to the existing vertices $x_1, \neg x_2, y$, respectively.



One-clause graph: add the second NAE triangle

Attach r_1, r_2, r_3 to $\neg y, x_3, F$. Both constraints share the same auxiliary pair.



Worked example: color a satisfied clause

Consider $(x_1 \vee \neg x_2 \vee x_3)$ with

$$x_1 = \mathsf{F}, \qquad x_2 = \mathsf{T}, \qquad x_3 = \mathsf{T}.$$

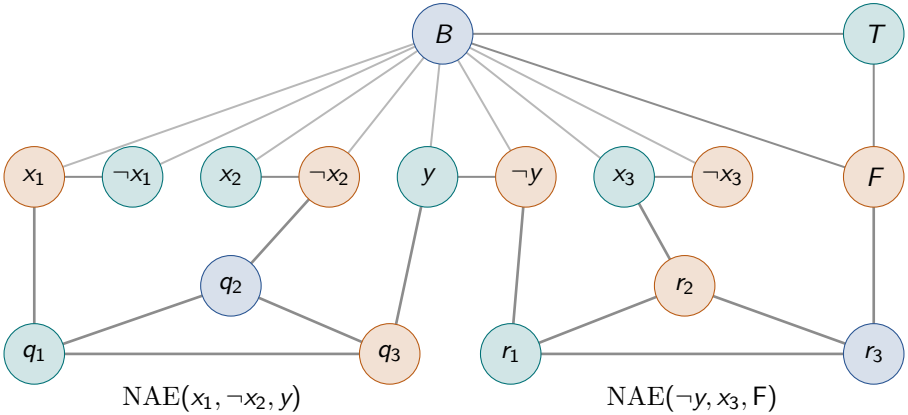
Its literal values are $(\mathsf{F}, \mathsf{F}, \mathsf{T})$, so choose $y = \mathsf{T}$.

Constraint	Input colors	Valid triangle colors
$\mathrm{NAE}(x_1, \neg x_2, y)$	$(\mathsf{F}, \mathsf{F}, \mathsf{T})$	$(\mathsf{T}, \mathsf{B}, \mathsf{F})$
$\mathrm{NAE}(\neg y, x_3, \mathsf{F})$	$(\mathsf{F}, \mathsf{T}, \mathsf{F})$	$(\mathsf{T}, \mathsf{F}, \mathsf{B})$

Each triangle has all three colors. In each position, the triangle color differs from its attached input color.

One-clause graph: a satisfying assignment extends

Take $x_1 = \mathsf{F}$, $x_2 = \mathsf{T}$, $x_3 = \mathsf{T}$, and choose $y = \mathsf{T}$. Every edge now joins different colors.



Global proof ($\Rightarrow$): color the Boolean vertices

Assume φ has a satisfying assignment A. We construct a proper 3-coloring of G_φ.

1. Color T, F, B with the three distinct colors $\mathsf{T}, \mathsf{F}, \mathsf{B}$.
2. For each original variable x_i, color its literal pair according to A: true literals receive T, false literals receive F.
3. For each true clause C_j, choose $y_j = \mathsf{F}$ if ℓ_{j1} or ℓ_{j2} is true. Otherwise choose $y_j = \mathsf{T}$; then ℓ_{j3} must be true. The clause proof shows that these choices satisfy both NAE constraints.
4. Color $y_j, \neg y_j$ by these complementary Boolean values.

All palette and variable-gadget edges are proper by the variable-gadget proof.

Global proof ($\Rightarrow$): extend every NAE triangle

Every NAE constraint now has Boolean input colors that are not all equal.

1. Apply the NAE-gadget construction to color its three internal vertices.
2. Its three triangle edges have distinct endpoint colors.
3. Its three attachment edges also have distinct endpoint colors, by the local proof.

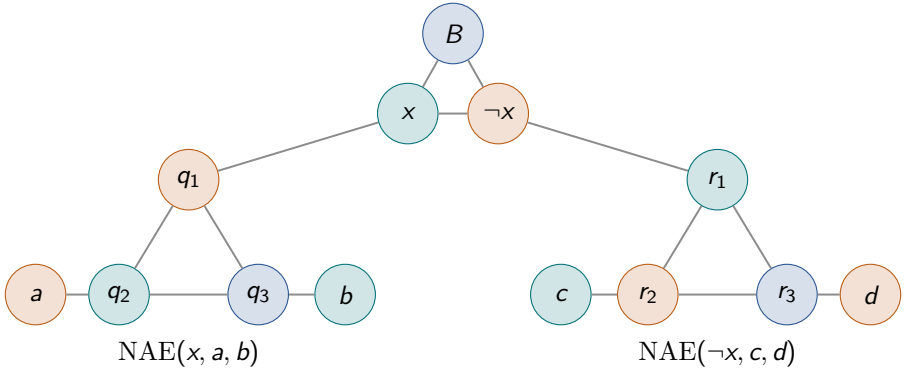
Why can all extensions coexist? Each triangle has fresh internal vertices. Distinct triangles share only already colored inputs, whose colors never change.

The graph contains only palette edges, variable-gadget edges, triangle edges, and attachment edges. We have checked all four types. Hence

$$\boxed{\varphi \text{ satisfiable} \implies G_\varphi \text{ 3-colorable.}}$$

Global proof ($\Rightarrow$): local colorings fit together

Example fragment: fix the Boolean colors first, then color each fresh triangle.



- The two triangles have distinct internal vertices; choosing one triangle's colors changes no vertex in the other.
- Both use the same existing variable pair $x, \neg x$. Its colors remain fixed throughout.

Global proof ($\Leftarrow$): read a consistent assignment

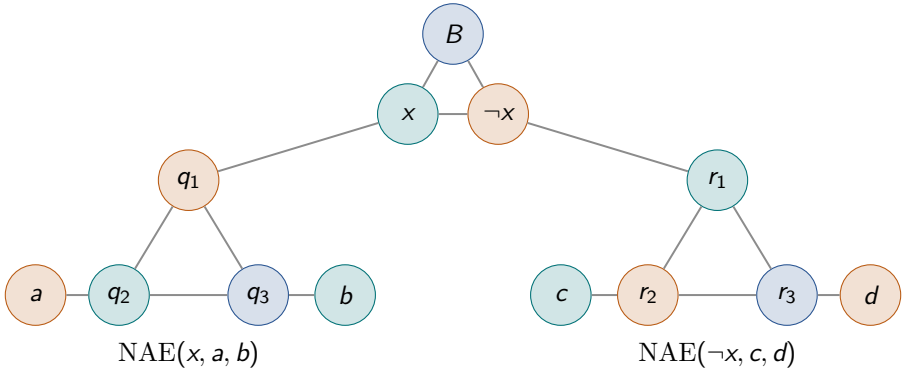
Assume G_φ has a proper 3-coloring c.

1. Its palette triangle has three distinct colors. Rename them $\mathsf{T}, \mathsf{F}, \mathsf{B}$ according to T, F, B.
2. By the variable-gadget proof, every pair $x, \neg x$ has opposite colors from $\{\mathsf{T}, \mathsf{F}\}$.
3. Assign $x = \text{true}$ exactly when $c(x) = \mathsf{T}$. Do this for all original and auxiliary variables.

Every occurrence of a literal uses the same literal vertex. Thus all occurrences receive a consistent truth value, and the vertex $\neg x$ represents its negation.

The shared palette vertex F represents the fixed false constant in every second NAE constraint.

Global proof ($\Leftarrow$): shared literals enforce consistency



1. Read $x = \mathsf{T}$ from the single vertex x; the variable gadget forces $\neg x = \mathsf{F}$.
2. Every constraint using either literal attaches to that **same vertex**. Its truth value cannot change from clause to clause.

The same argument for every variable gives one assignment for the whole formula.

Global proof ($\Leftarrow$): every original clause is true

Fix an arbitrary clause $C_j = (\ell_{j1} \vee \ell_{j2} \vee \ell_{j3})$.

1. Its two NAE triangles have proper colorings, inherited from c.
2. By the necessary direction of the NAE-gadget proof, their Boolean inputs satisfy

$$\mathrm{NAE}(\ell_{j1}, \ell_{j2}, y_j), \qquad \mathrm{NAE}(\neg y_j, \ell_{j3}, \mathsf{F}).$$

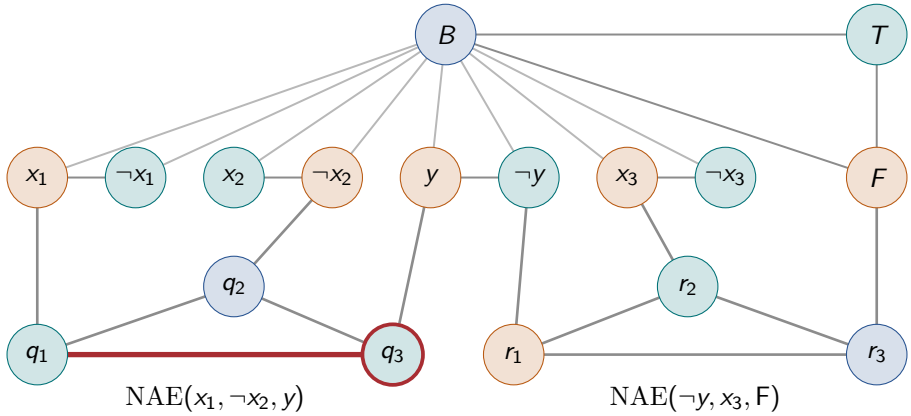
3. If ℓ_{j1} or ℓ_{j2} is true, then C_j is true. Otherwise the first constraint forces $y_j = \mathsf{T}$; the second then forces $\ell_{j3} = \mathsf{T}$.

Since j was arbitrary, every clause is true under the same assignment. Restricting it to the original variables satisfies φ. Therefore

$$\boxed{G_\varphi \text{ 3-colorable} \implies \varphi \text{ satisfiable.}}$$

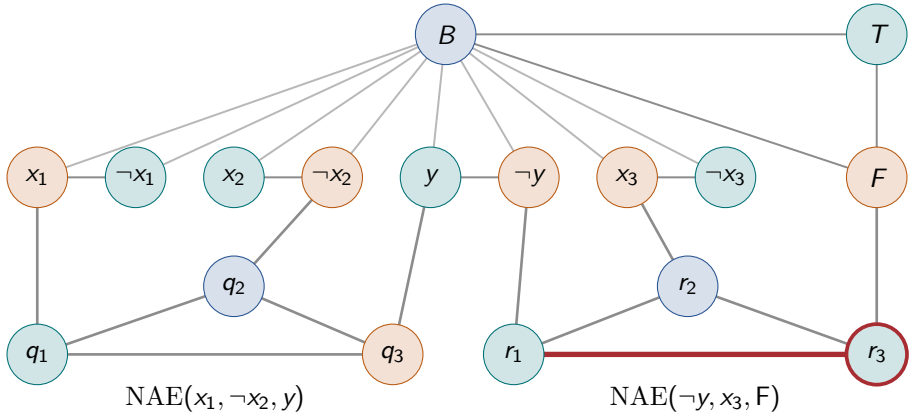
False clause: attempted coloring with $y = \mathrm{F}$

Here $(x_1, \neg x_2, x_3) = (\mathrm{F}, \mathrm{F}, \mathrm{F})$. With $y = \mathrm{F}$, the first inputs are all false, so the NAE lemma rules out every extension. Red shows one failed attempt.



False clause: attempted coloring with $y = \mathrm{T}$

With $y = \mathrm{T}$, the second inputs $(\neg y, x_3, F)$ are all false. The NAE lemma rules out every extension. Red shows one failed attempt.



The construction has polynomial size

Suppose φ has n original variables and m clauses.

Part	Vertices	Edges added
Palette triangle	3	3
$n + m$ variable gadgets	$2(n + m)$	$3(n + m)$
$2m$ NAE gadgets	$6m$	$12m$
Total	$3 + 2n + 8m$	$3 + 3n + 15m$

- Each NAE gadget adds three triangle edges and three attachment edges.
- All vertices and edges can be listed in $O(n + m)$ construction steps.

The encoded graph has size polynomial in the encoded formula size.

The NP-completeness conclusion

We have shown:

$$\varphi \in \mathrm{3SAT} \quad \Longleftrightarrow \quad G_\varphi \in \mathrm{G3C},$$

and $\varphi \mapsto G_\varphi$ is computable in polynomial time.

Theorem 5.3.1

Since $\mathrm{3SAT}$ is NP-complete and

$$\mathrm{3SAT} \leq_P \mathrm{G3C}, \qquad \mathrm{G3C} \in \mathrm{NP},$$

$\mathrm{G3C}$ is NP-complete.

The reduction uses a hypothetical 3-coloring algorithm to decide 3SAT.

Takeaways

- **Kőnig's theorem:** in a **bipartite graph**, the maximum matching size equals the minimum vertex-cover size. General graphs need not satisfy this equality.
- **Graph 3-coloring:** the palette fixes Boolean colors, variable pairs enforce negation, and NAE triangles enforce the transformed clauses.
- **Proof discipline:** show both correctness directions and bound the construction size.

$$\boxed{C = (L \setminus Z_L) \cup Z_R} \qquad \boxed{\mathrm{3SAT} \leq_P \mathrm{G3C}}$$