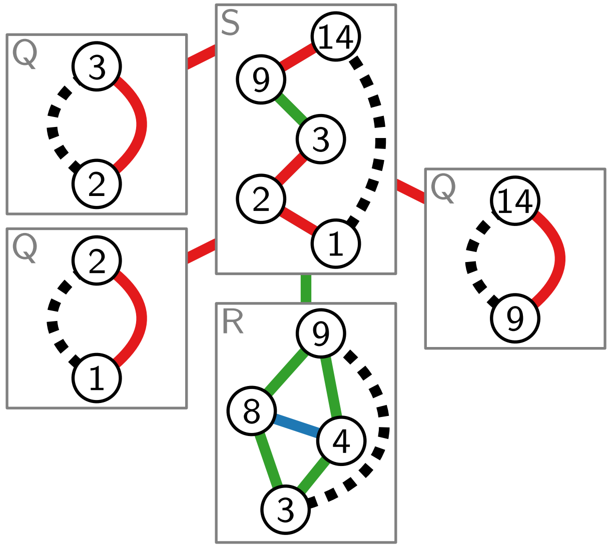
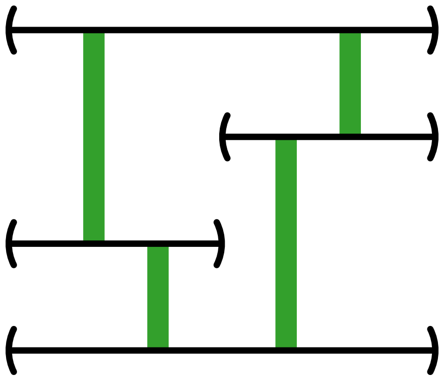
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Visualization of Graphs

Lecture 10:
Partial Visibility Representation Extension



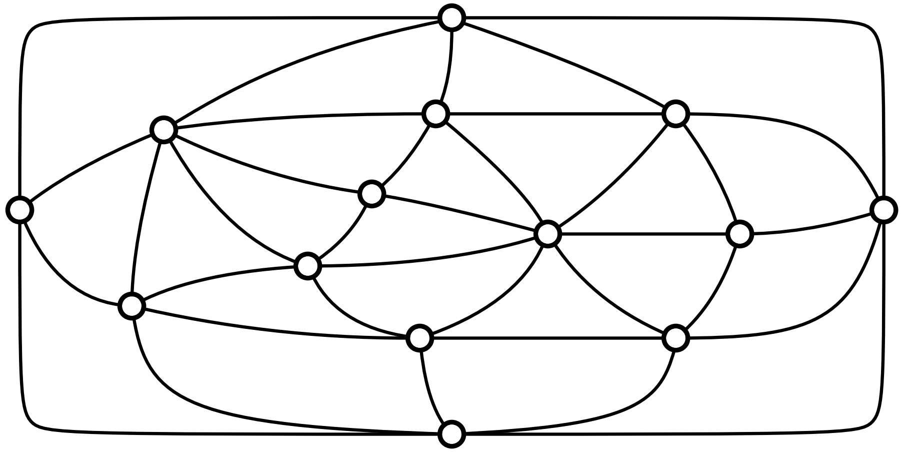


Alexander Wolff

Summer term 2026

Partial Representation Extension Problem

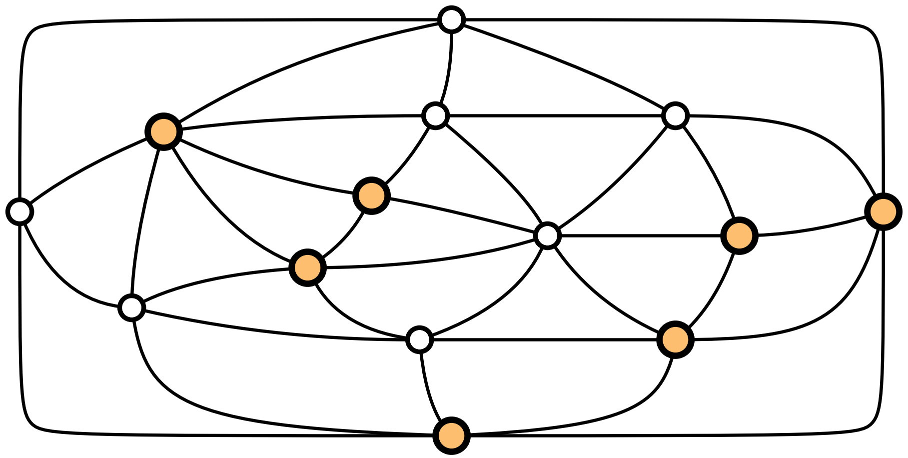
Let G be a graph.



Partial Representation Extension Problem

Let G be a graph.

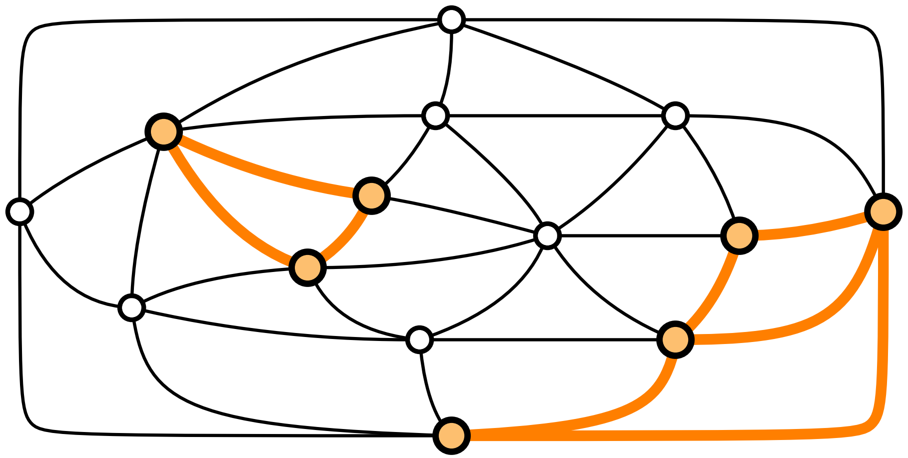
Let $V' \subseteq V(G)$



Partial Representation Extension Problem

Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

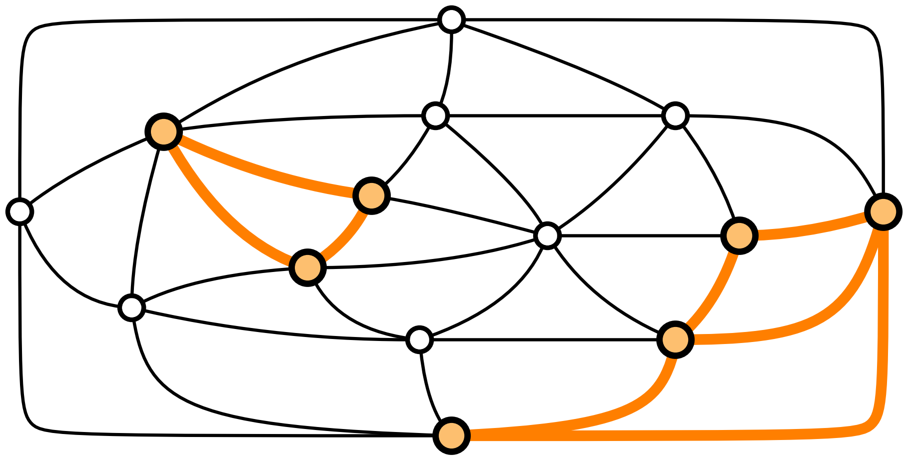


Partial Representation Extension Problem

Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

induced subgraph of G w.r.t. V':
V' and all edges among V'



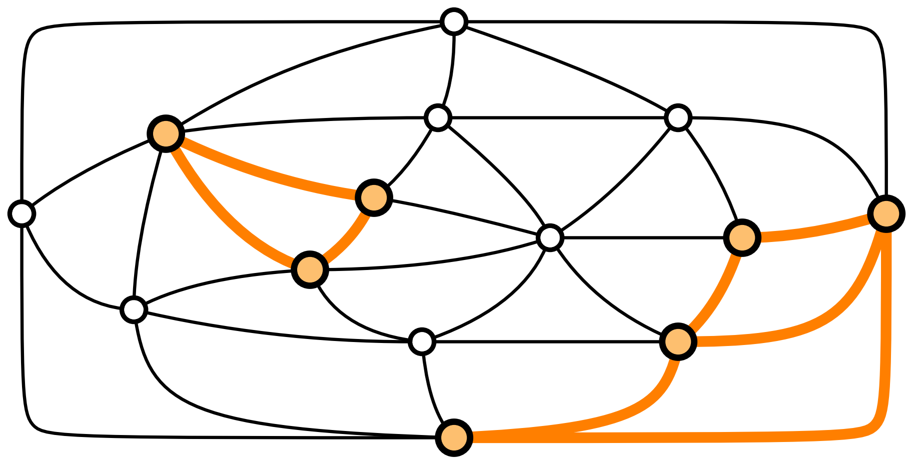
Partial Representation Extension Problem

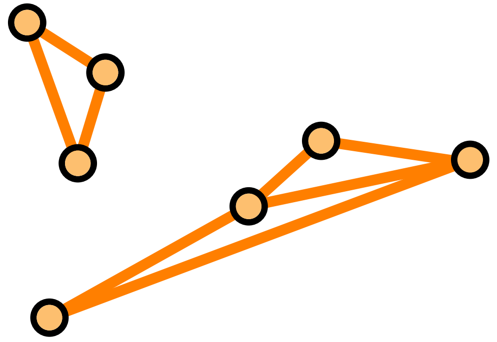
Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

induced subgraph of G w.r.t. V':
V' and all edges among V'

Let Γ_H be a representation of H.





Partial Representation Extension Problem

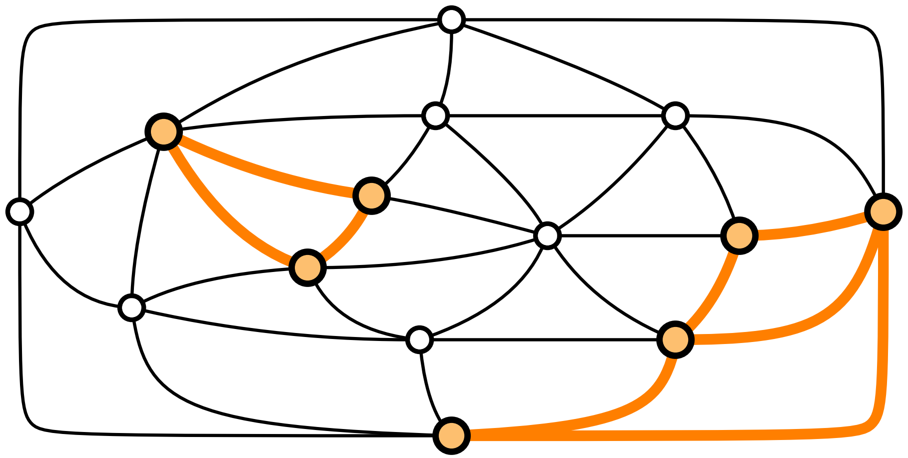
Let G be a graph.

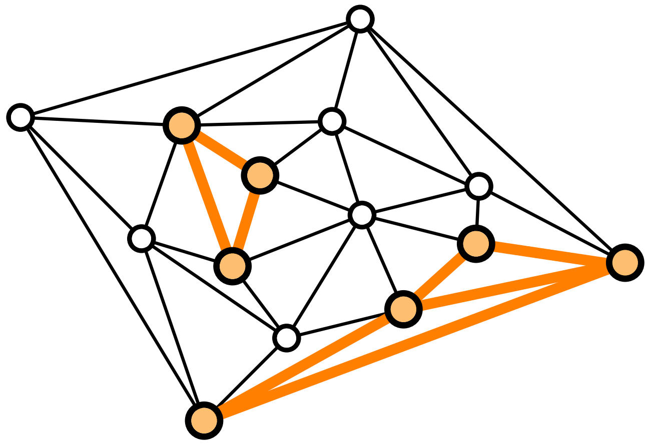
Let $V' \subseteq V(G)$ and $H = \boxed{G[V']}$

induced subgraph of G w.r.t. V':
V' and all edges among V'

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.





Partial Representation Extension Problem

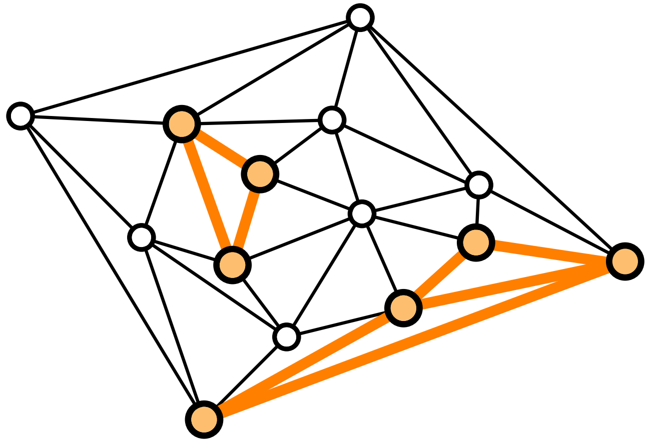
induced subgraph of G w.r.t. V':
V' and all edges among V'

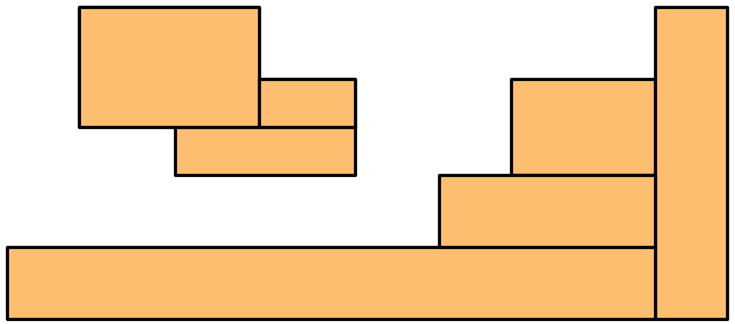
Let G be a graph.

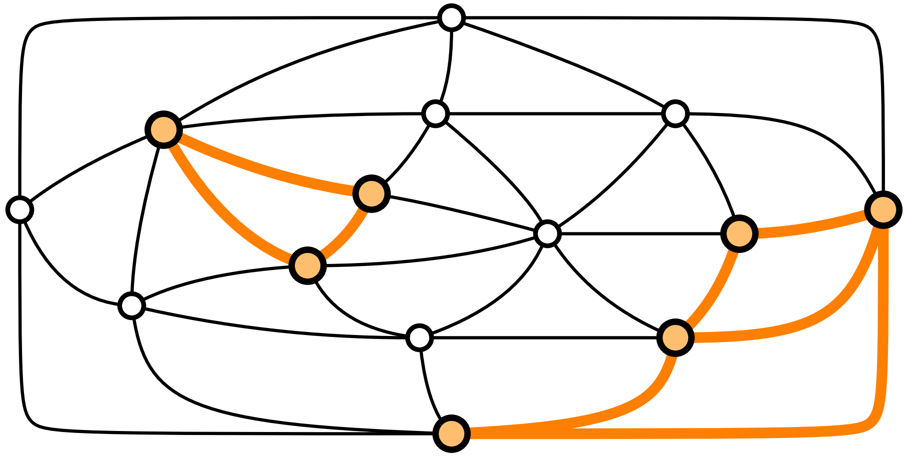
Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.







Partial Representation Extension Problem

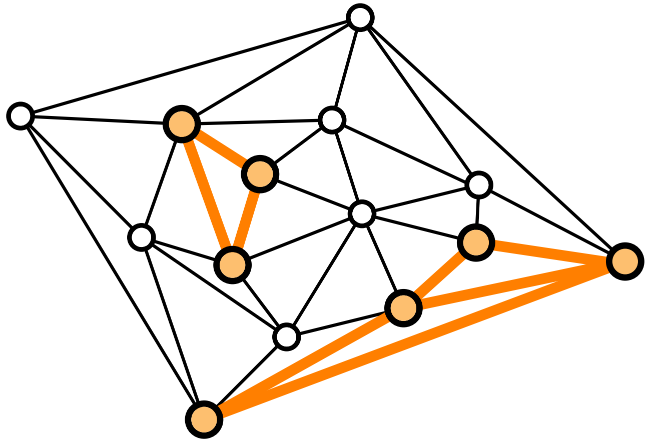
Let G be a graph.

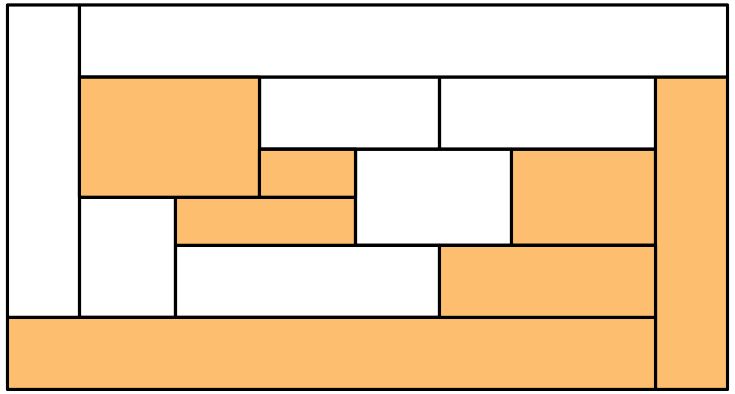
Let $V' \subseteq V(G)$ and $H = G[V']$

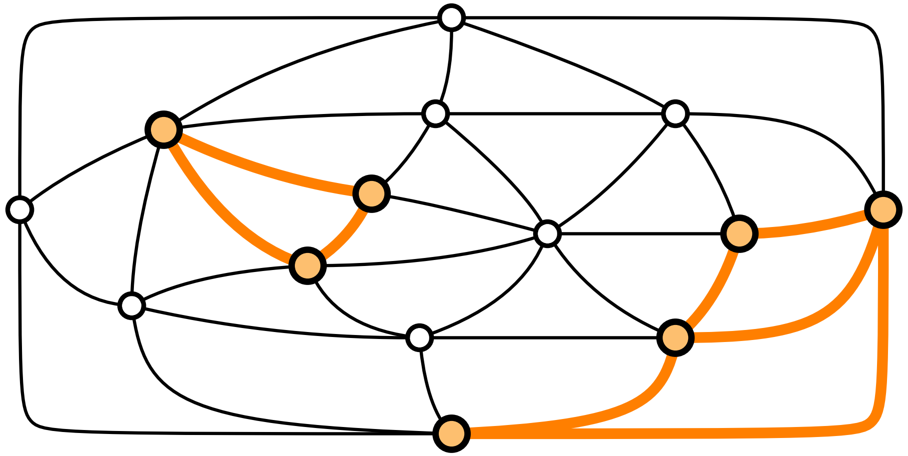
induced subgraph of G w.r.t. V': V' and all edges among V'

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.







Partial Representation Extension Problem

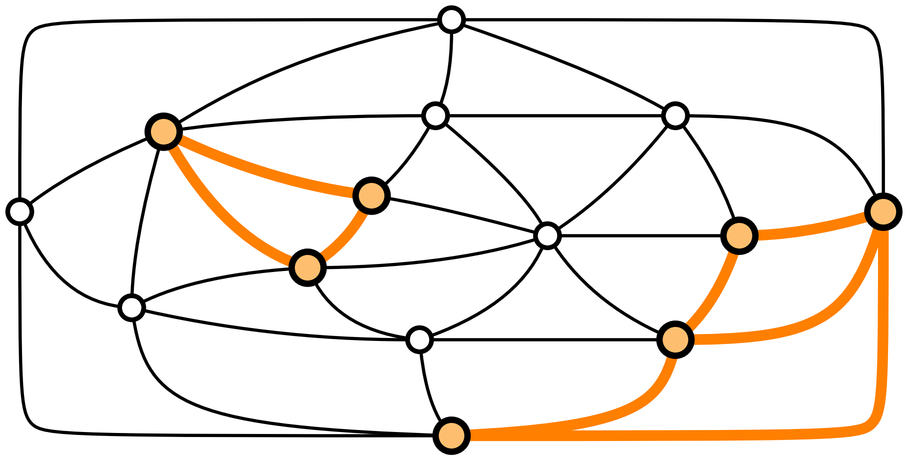
Let G be a graph.

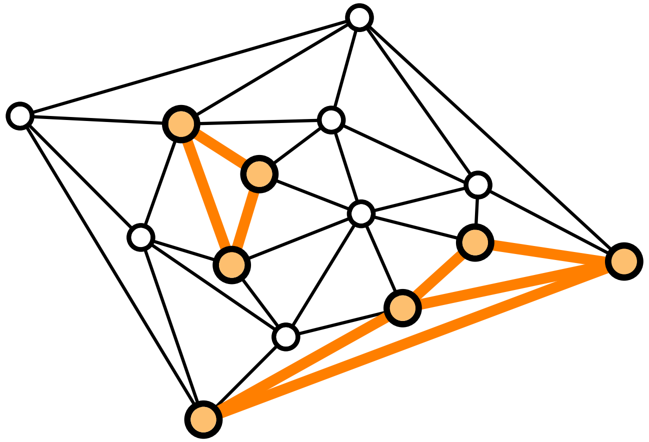
Let $V' \subseteq V(G)$ and $H = \boxed{G[V']}$

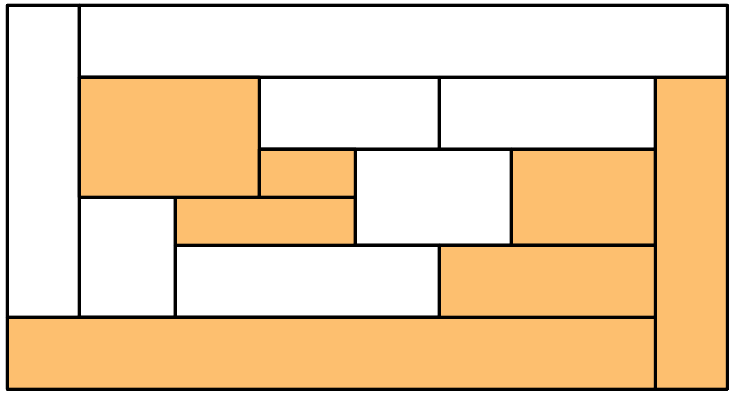
induced subgraph of G w.r.t. V':
V' and all edges among V'

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.







Polytime for:

Partial Representation Extension Problem

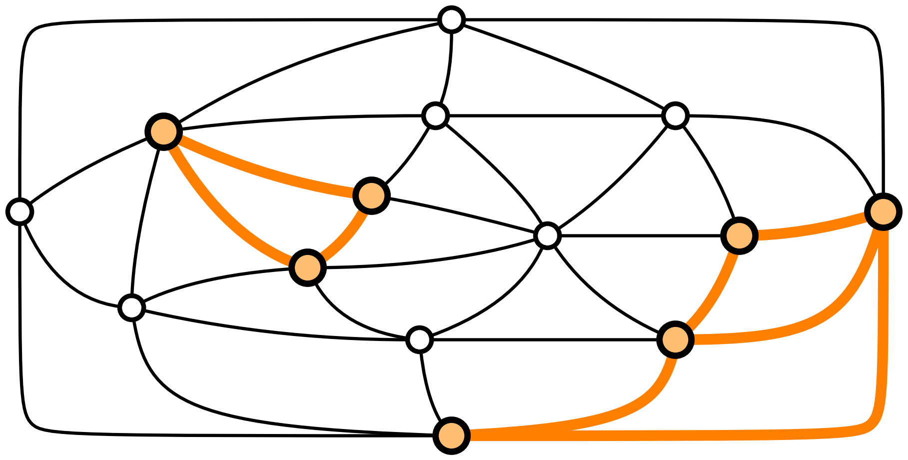
induced subgraph of G w.r.t. V':
V' and all edges among V'

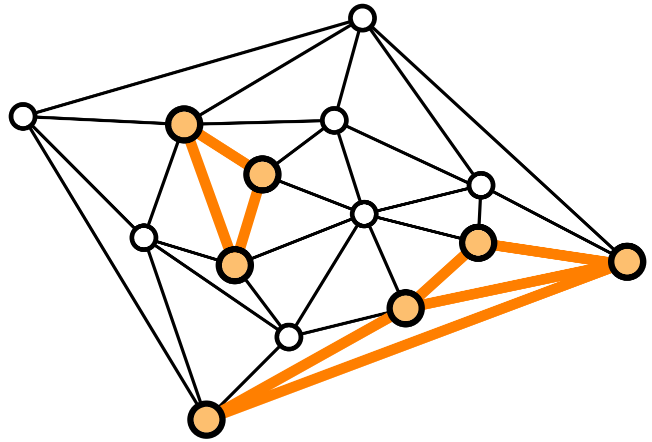
Let G be a graph.

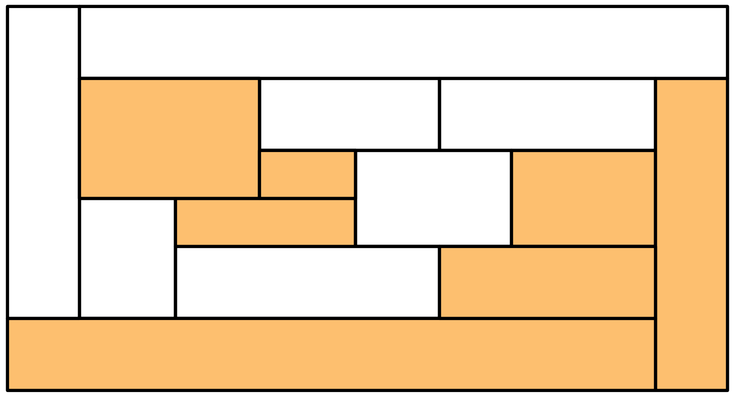
Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.

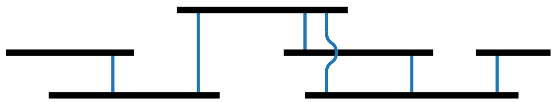






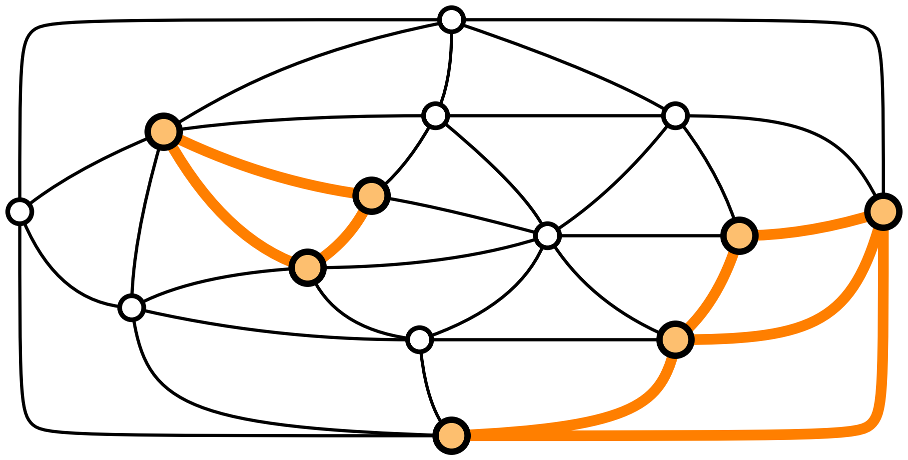
Polytime for:

- (unit) interval graphs



Partial Representation Extension Problem

induced subgraph of G w.r.t. V':
V' and all edges among V'

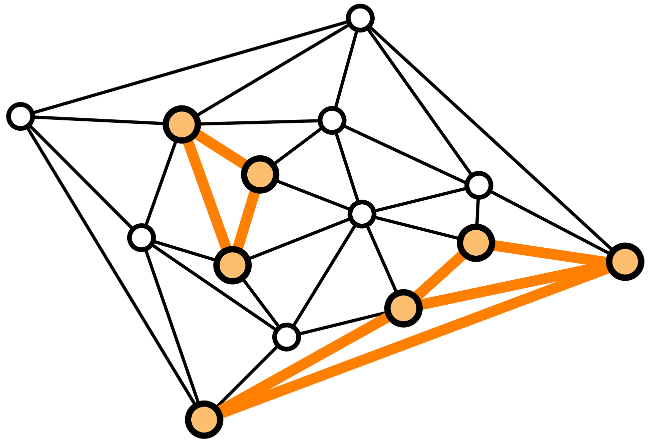


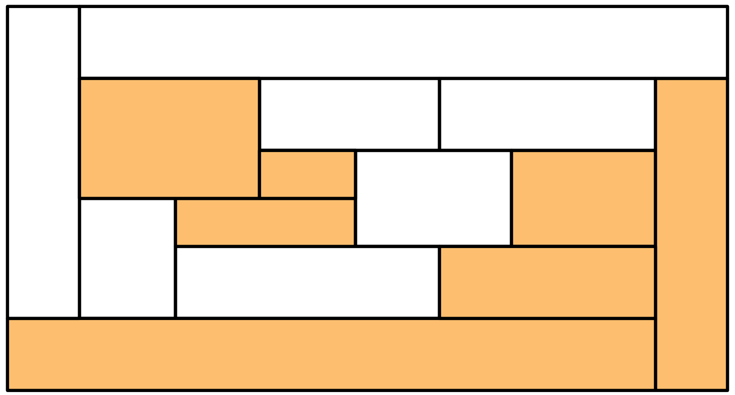
Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

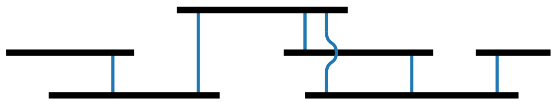
Find a representation Γ_G of G that *extends* Γ_H.



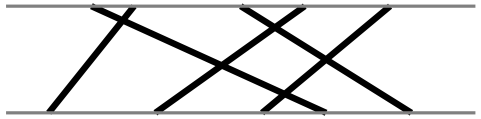


Polytime for:

- (unit) interval graphs



- permutation graphs



Partial Representation Extension Problem

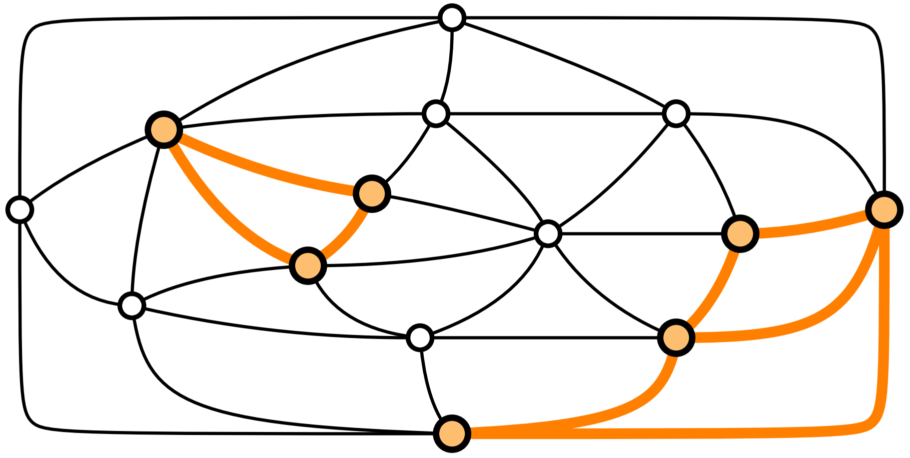
Let G be a graph.

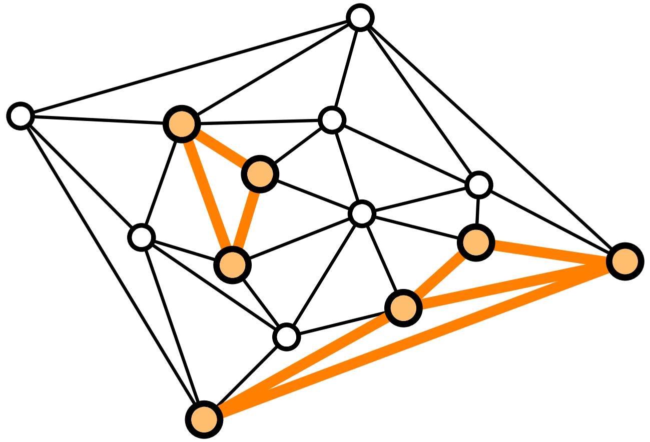
induced subgraph of G w.r.t. V': V' and all edges among V'

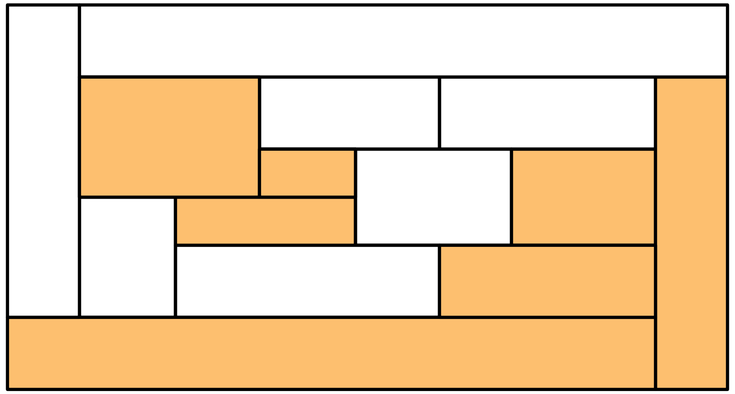
Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.

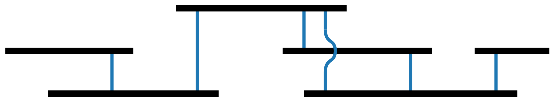




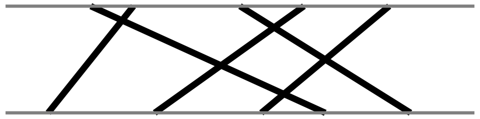


Polytime for:

- (unit) interval graphs



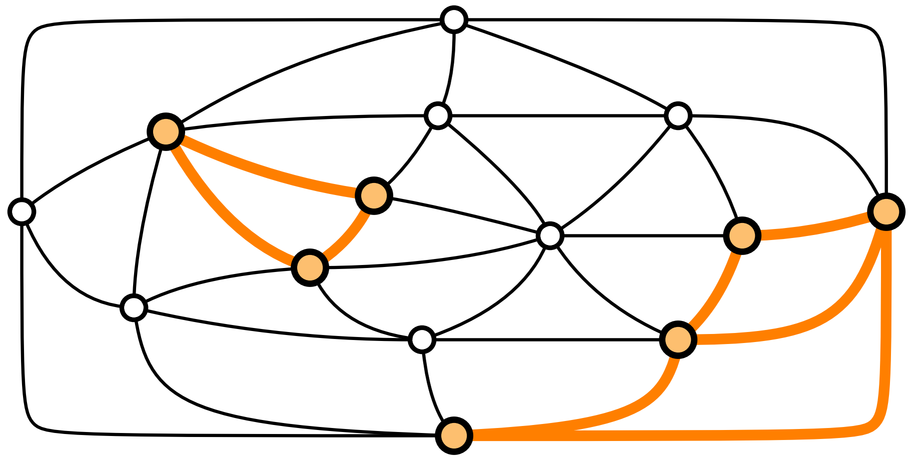
- permutation graphs



- circle graphs

Partial Representation Extension Problem

induced subgraph of G w.r.t. V':
V' and all edges among V'

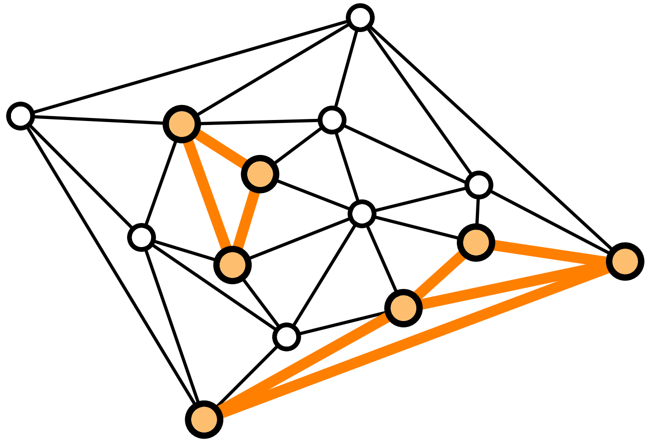


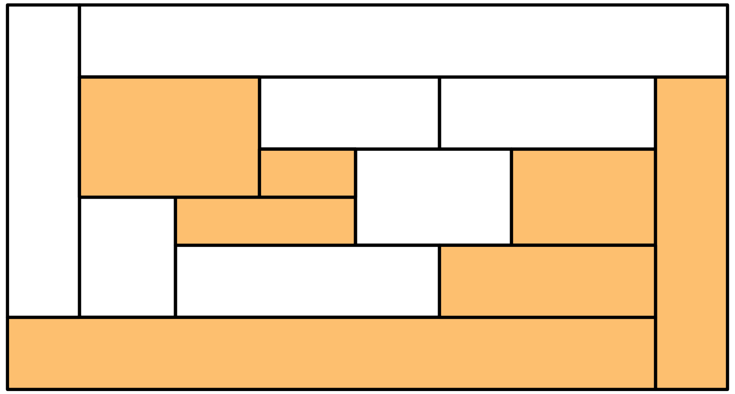
Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

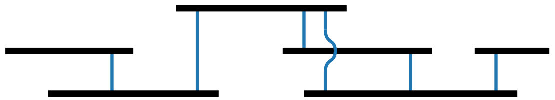
Find a representation Γ_G of G that *extends* Γ_H.



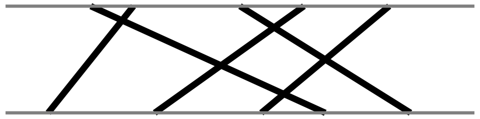


Polytime for:

- (unit) interval graphs



- permutation graphs



- circle graphs

NP-hard for:

Partial Representation Extension Problem

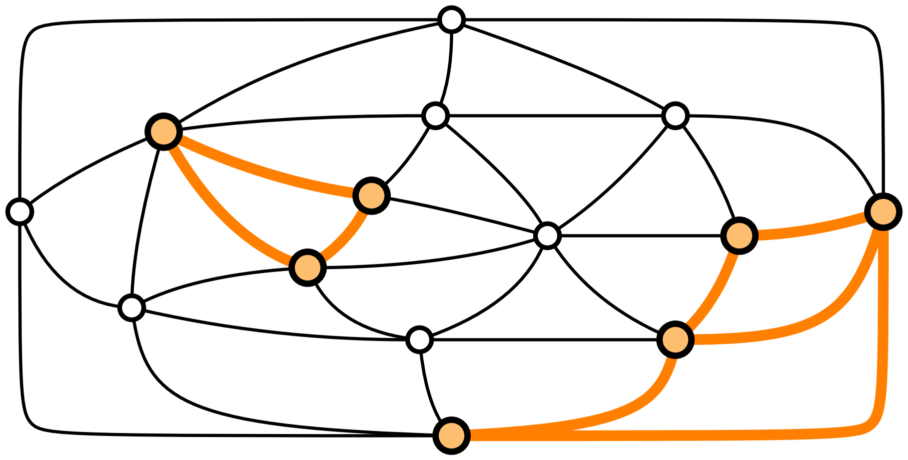
Let G be a graph.

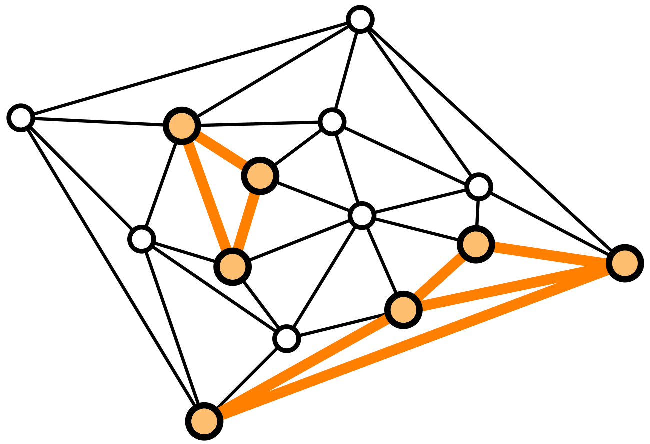
Let $V' \subseteq V(G)$ and $H = G[V']$

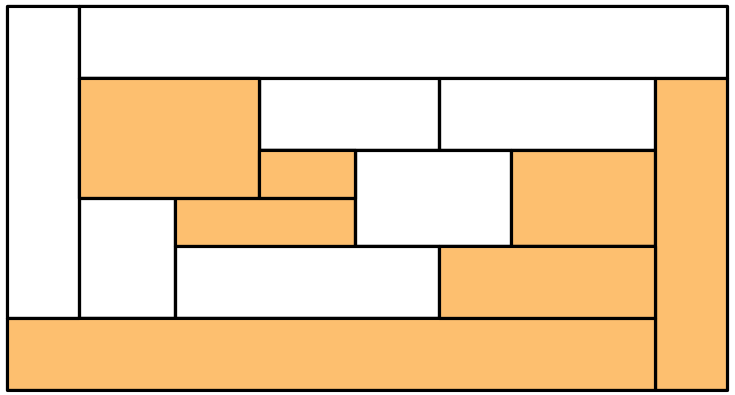
induced subgraph of G w.r.t. V':
V' and all edges among V'

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.

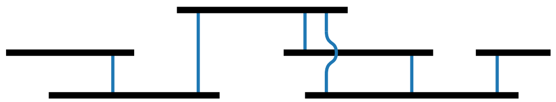




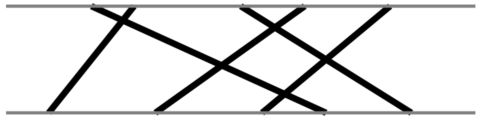


Polytime for:

- (unit) interval graphs



- permutation graphs



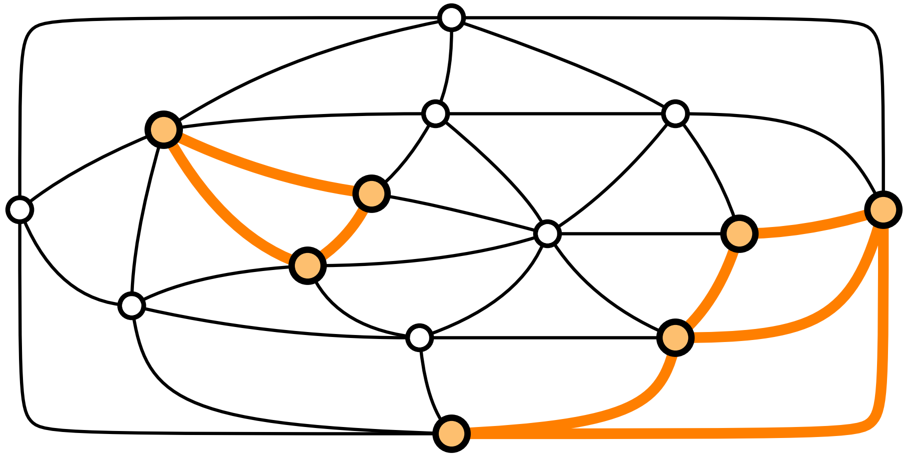
- circle graphs

NP-hard for:

- planar straight-line drawings

Partial Representation Extension Problem

induced subgraph of G w.r.t. V':
V' and all edges among V'

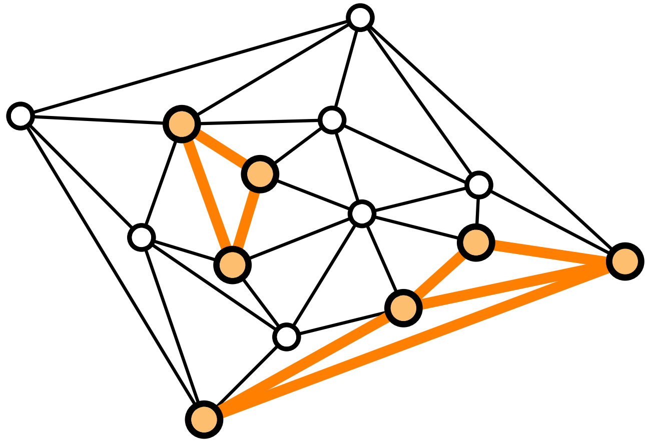


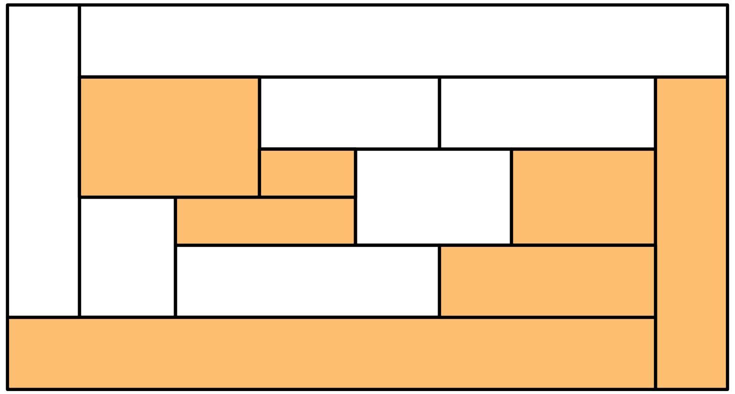
Let G be a graph.

Let $V' \subseteq V(G)$ and $H = \boxed{G[V']}$

Let Γ_H be a representation of H.

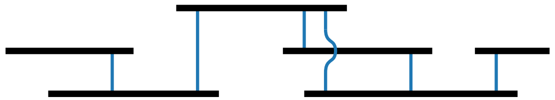
Find a representation Γ_G of G that *extends* Γ_H.



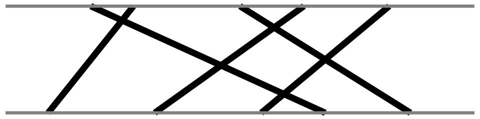


Polytime for:

- (unit) interval graphs



- permutation graphs



- circle graphs

NP-hard for:

- planar straight-line drawings
- contacts of

Partial Representation Extension Problem

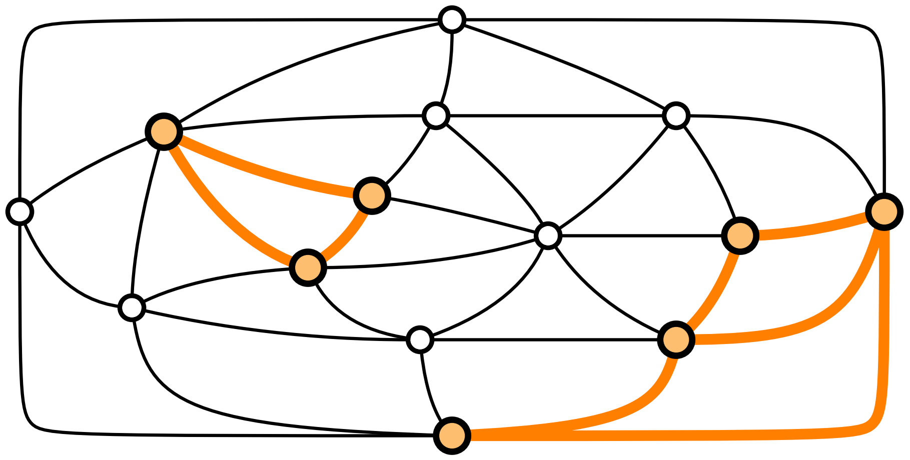
induced subgraph of G w.r.t. V': V' and all edges among V'

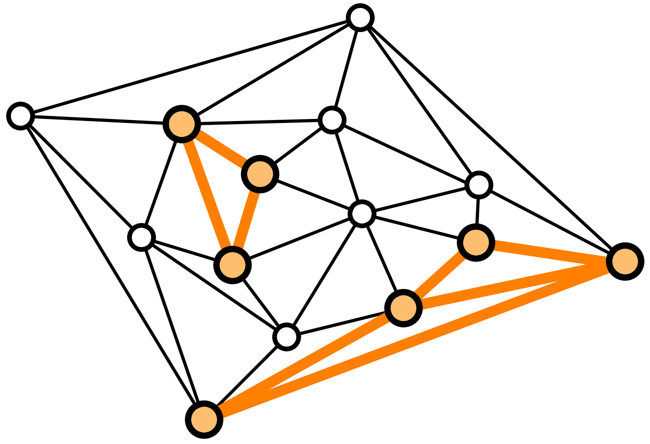
Let G be a graph.

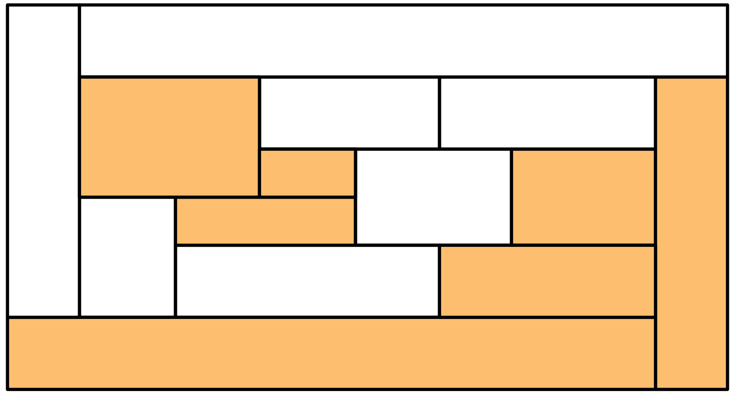
Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.

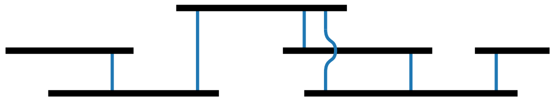




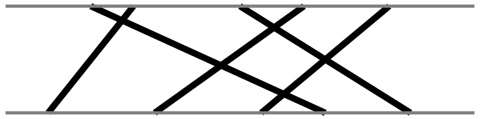


Polytime for:

- (unit) interval graphs



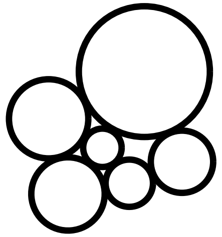
- permutation graphs



- circle graphs

NP-hard for:

- planar straight-line drawings
- contacts of
  - disks



Partial Representation Extension Problem

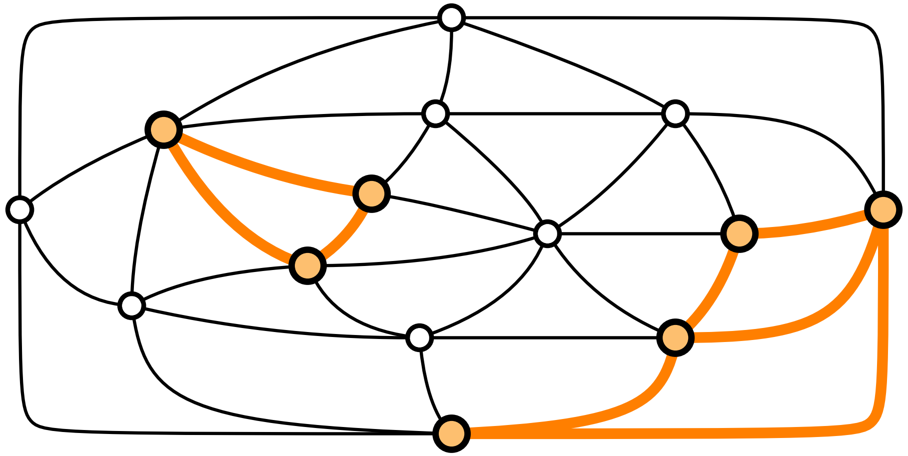
Let G be a graph.

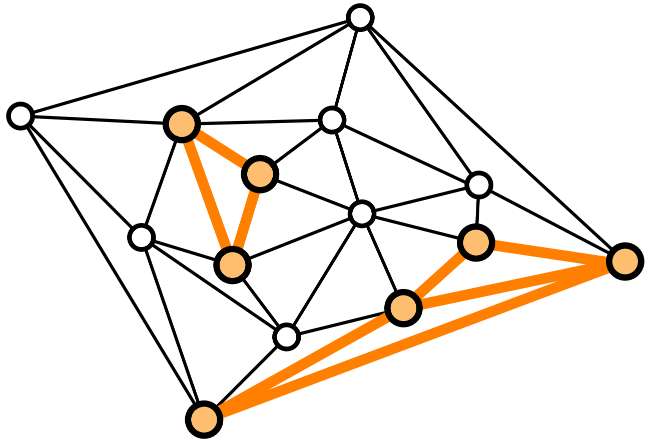
Let $V' \subseteq V(G)$ and $H = G[V']$

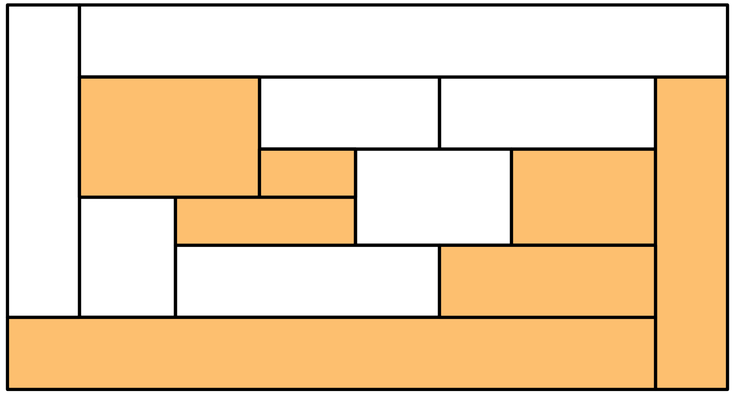
induced subgraph of G w.r.t. V': V' and all edges among V'

Let Γ_H be a representation of H.

Find a representation Γ_G of G that *extends* Γ_H.

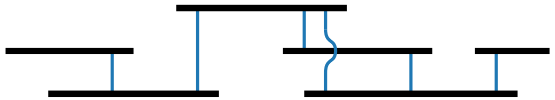




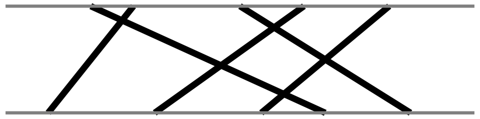


Polytime for:

- (unit) interval graphs



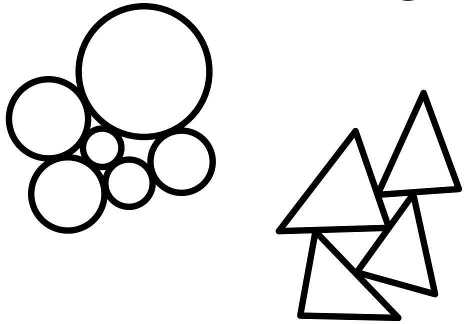
- permutation graphs



- circle graphs

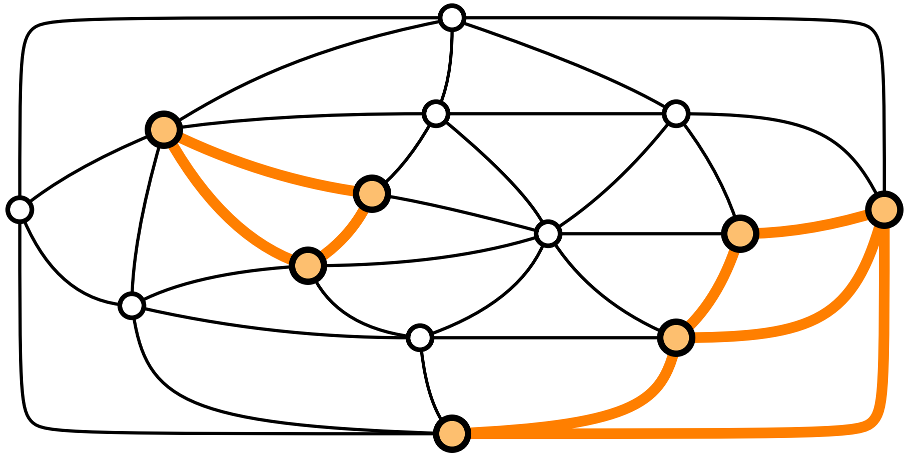
NP-hard for:

- planar straight-line drawings
- contacts of
  - disks
  - triangles



Partial Representation Extension Problem

induced subgraph of G w.r.t. V':
V' and all edges among V'

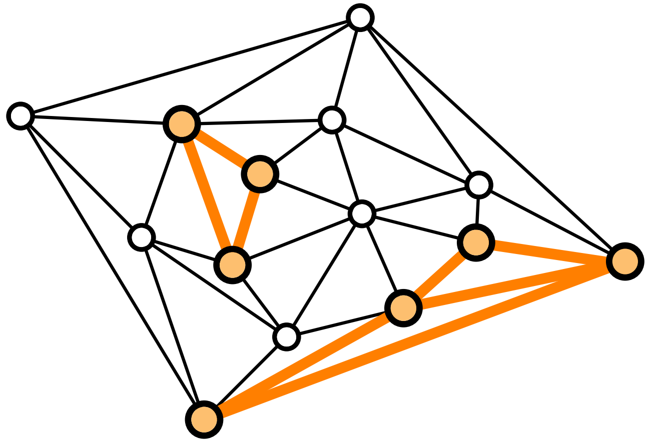


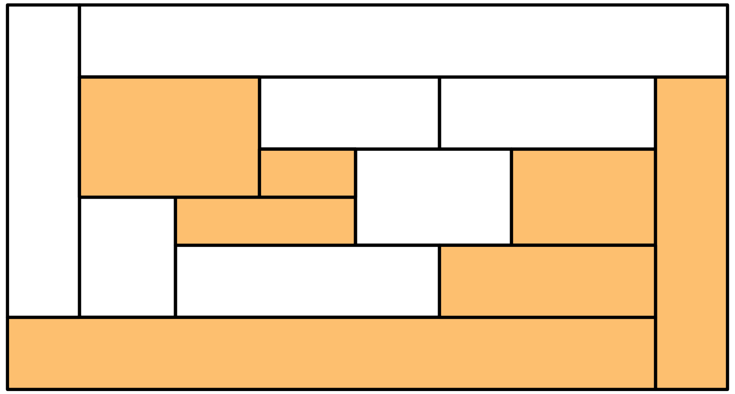
Let G be a graph.

Let $V' \subseteq V(G)$ and $H = G[V']$

Let Γ_H be a representation of H.

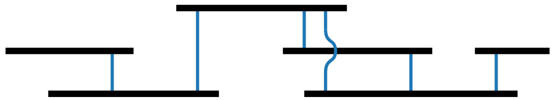
Find a representation Γ_G of G that *extends* Γ_H.



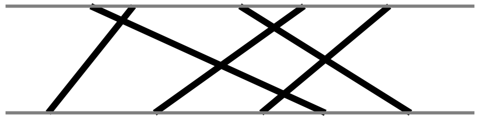


Polytime for:

- (unit) interval graphs



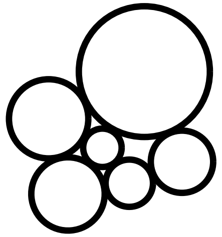
- permutation graphs

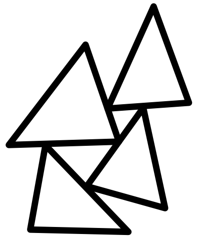


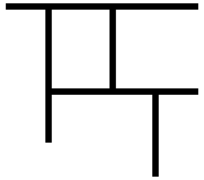
- circle graphs

NP-hard for:

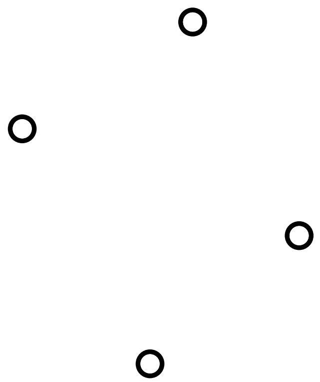
- planar straight-line drawings
- contacts of
  - disks
  - triangles
  - orthogonal segments





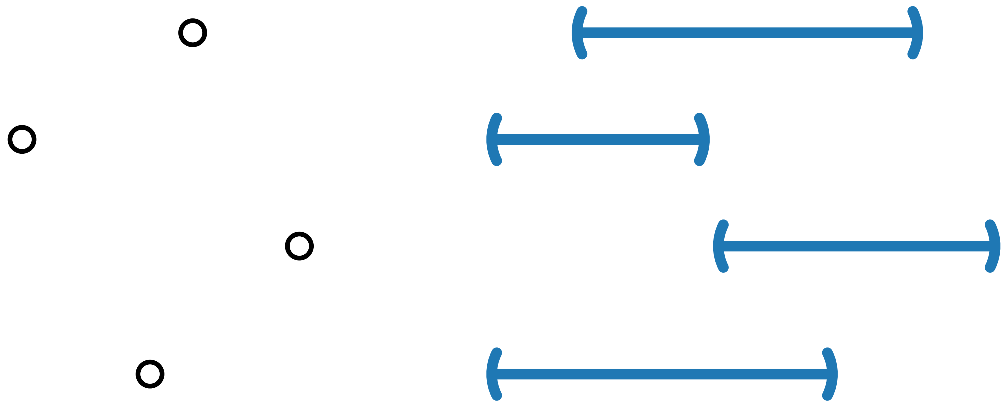


Bar Visibility Representation



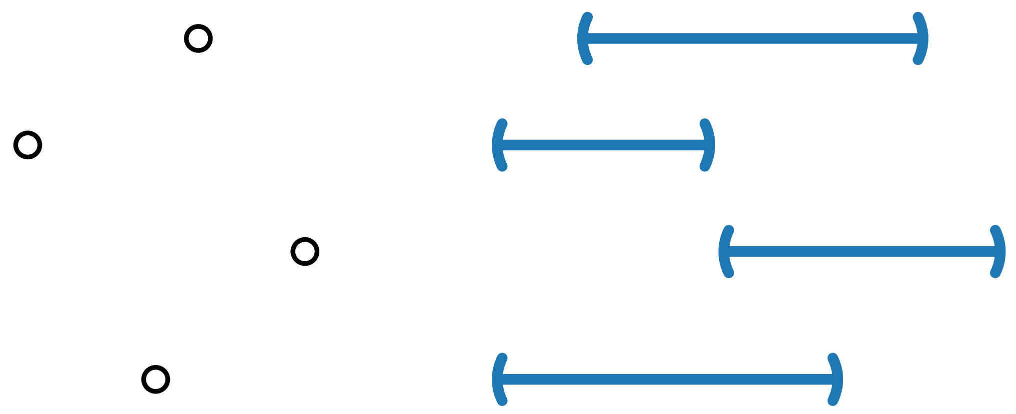
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.



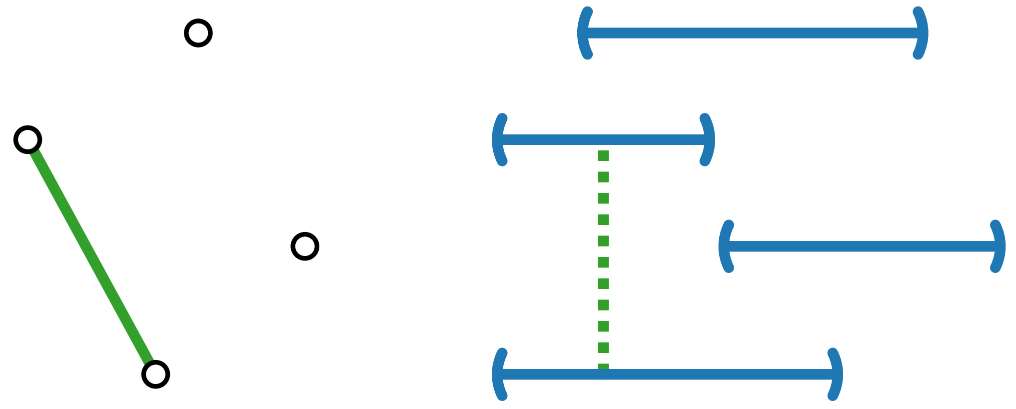
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



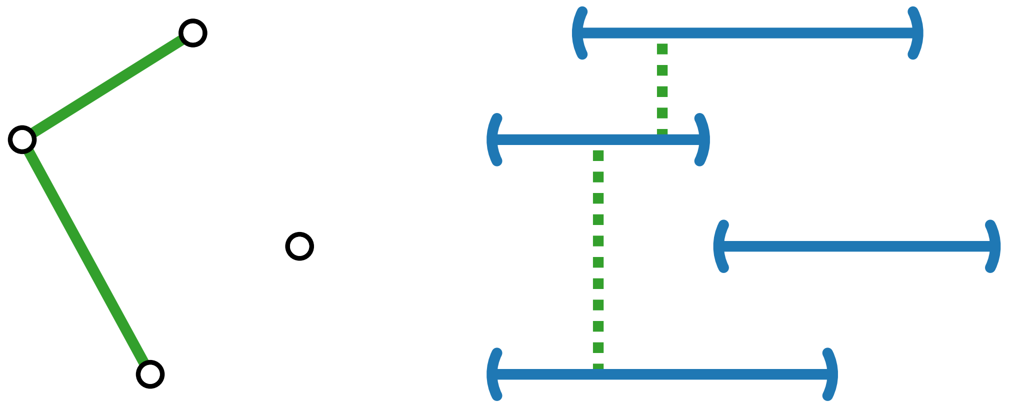
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



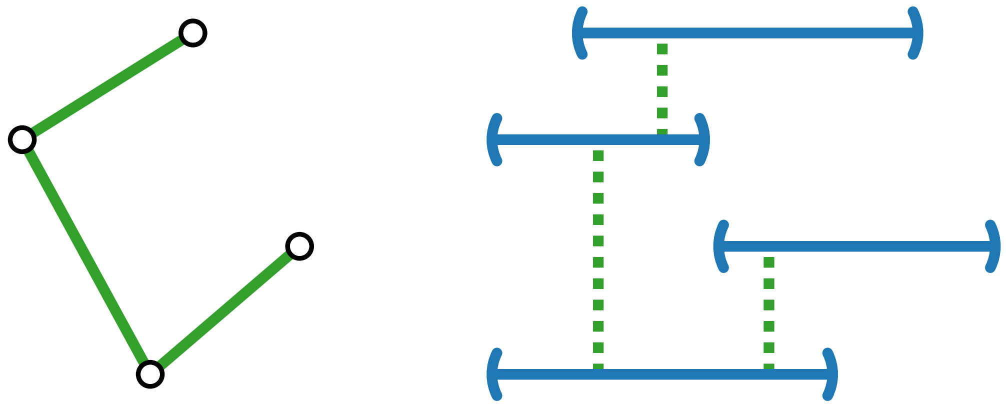
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



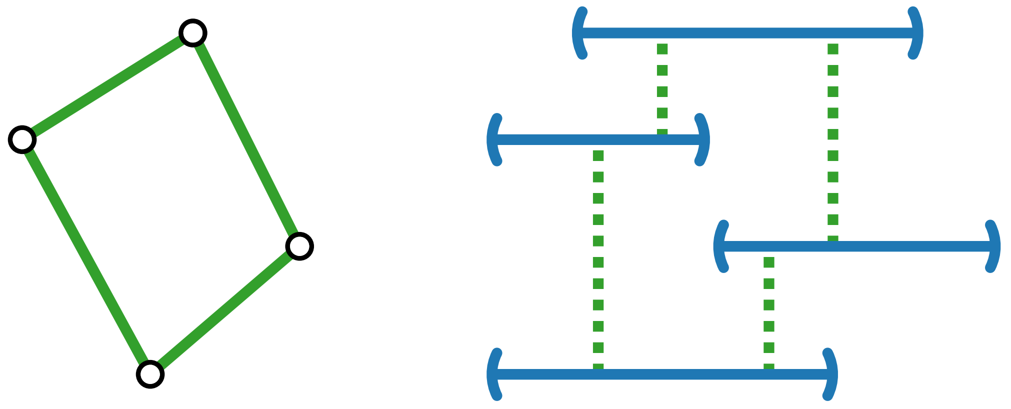
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



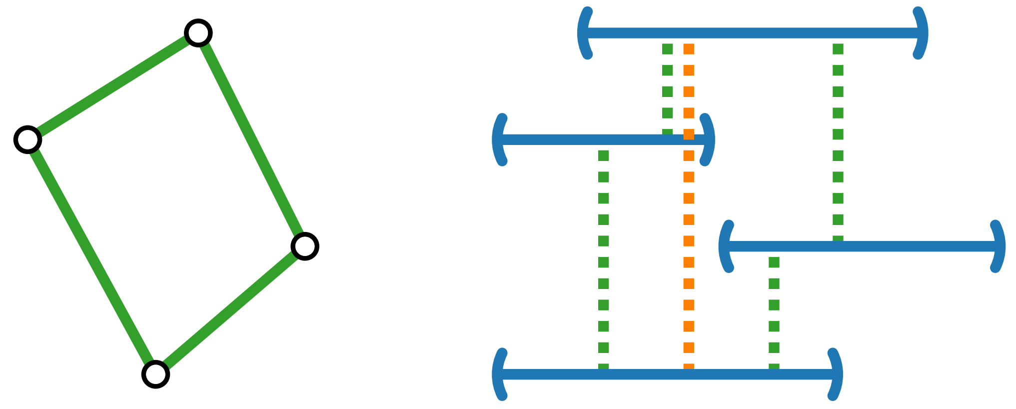
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



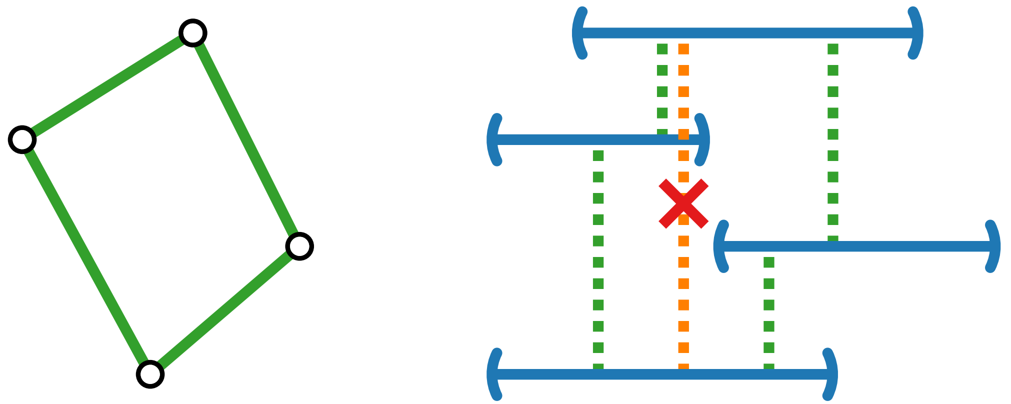
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



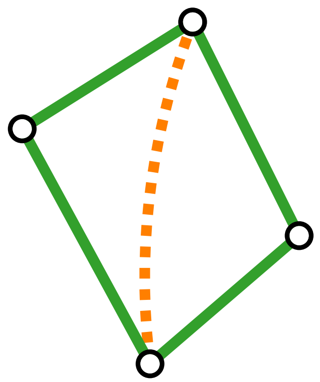
Bar Visibility Representation

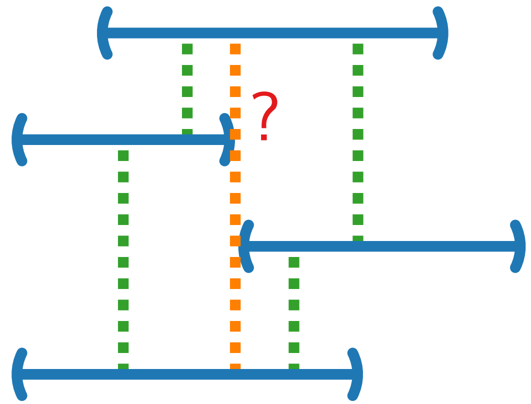
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.



Bar Visibility Representation

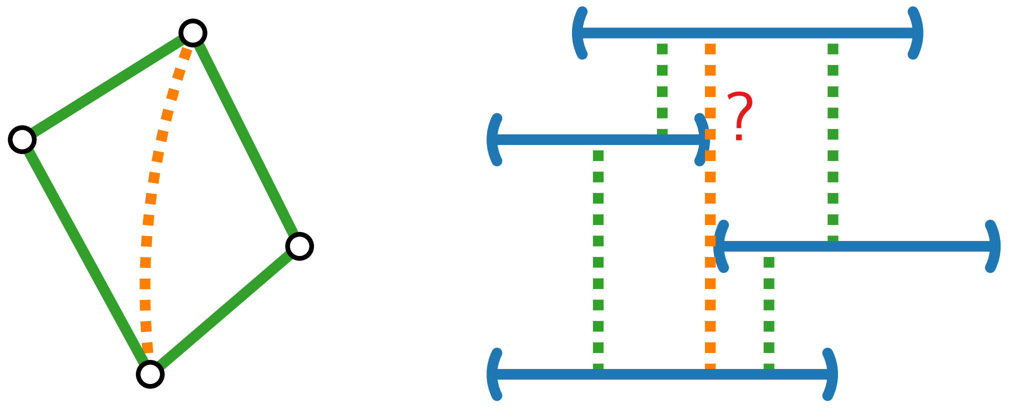
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.





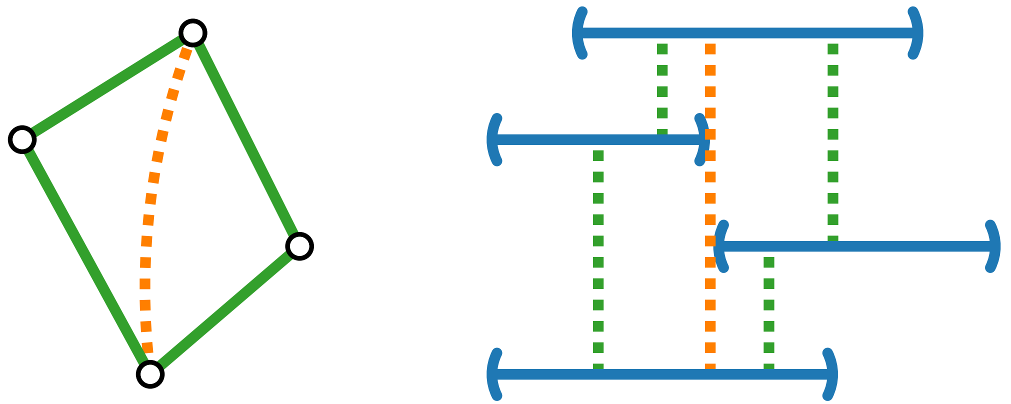
Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?



Bar Visibility Representation

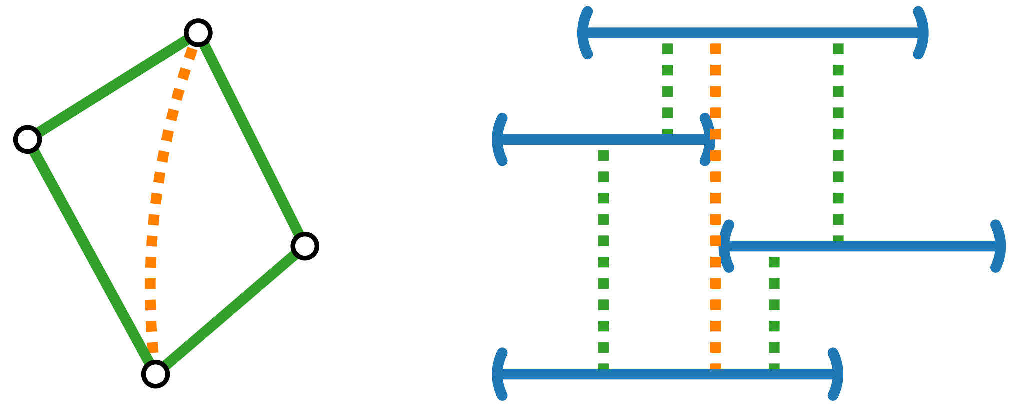
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?



Models.

Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?

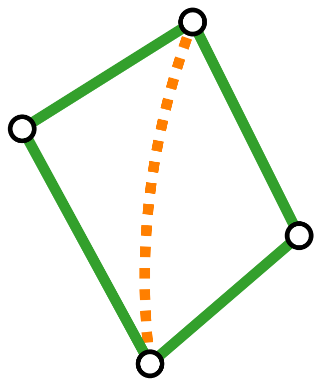


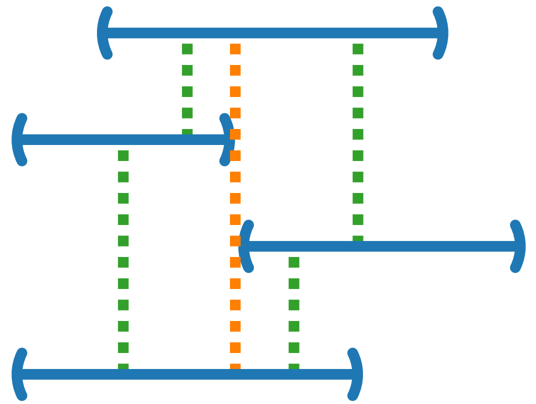
Models.

- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.

Bar Visibility Representation

- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?



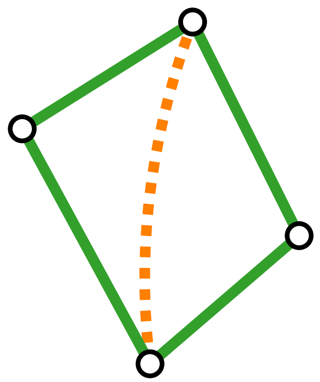


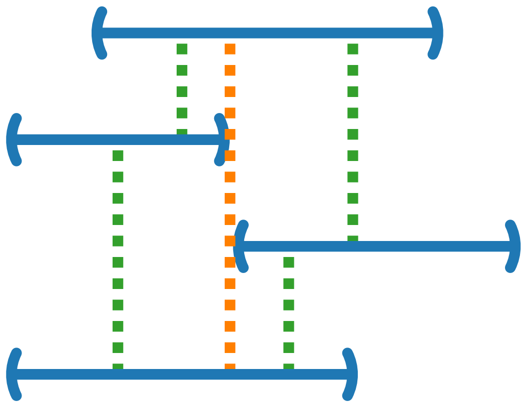
Models.

- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.
- **Epsilon:**
  Edge $uv \Leftrightarrow$ ε-wide vertical lines of sight for some $\varepsilon > 0$.

Bar Visibility Representation

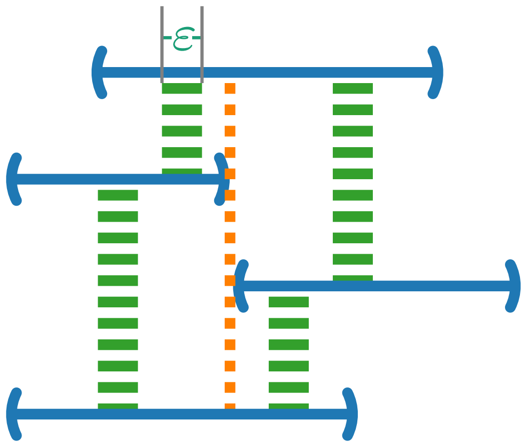
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?





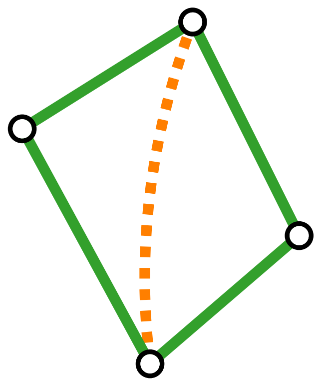
Models.

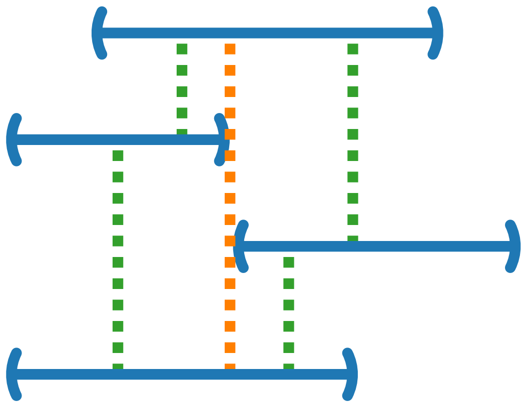
- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.
- **Epsilon:**
  Edge $uv \Leftrightarrow$ ε-wide vertical lines of sight for some $\varepsilon > 0$.



Bar Visibility Representation

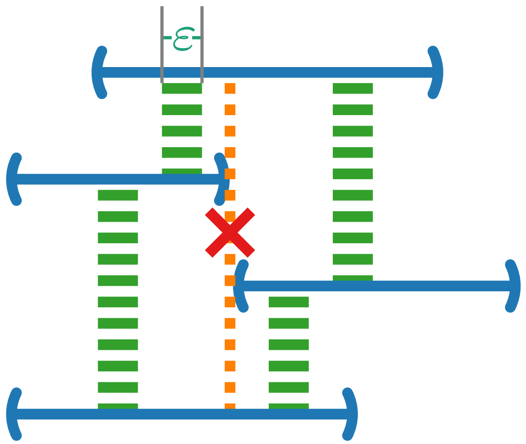
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?





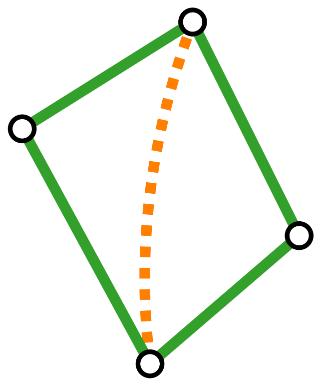
Models.

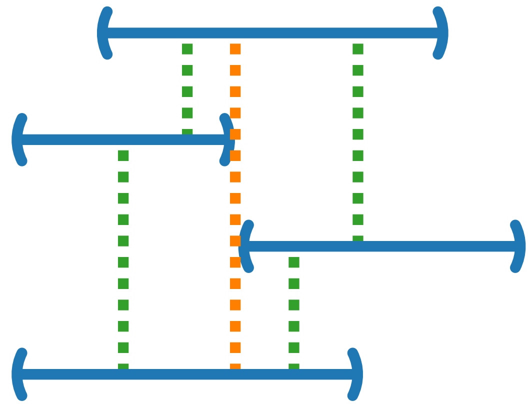
- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.
- **Epsilon:**
  Edge $uv \Leftrightarrow$ ε-wide vertical lines of sight for some $\varepsilon > 0$.



Bar Visibility Representation

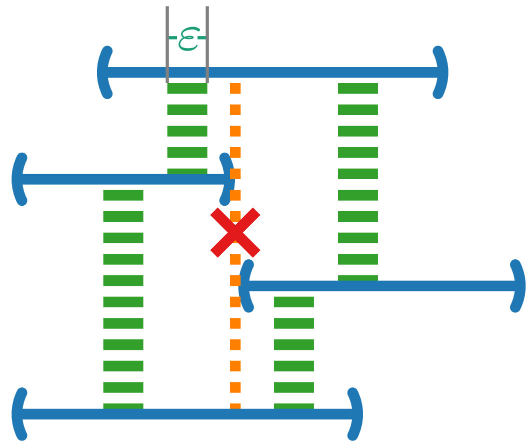
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?





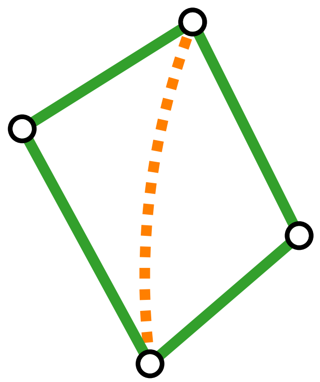
Models.

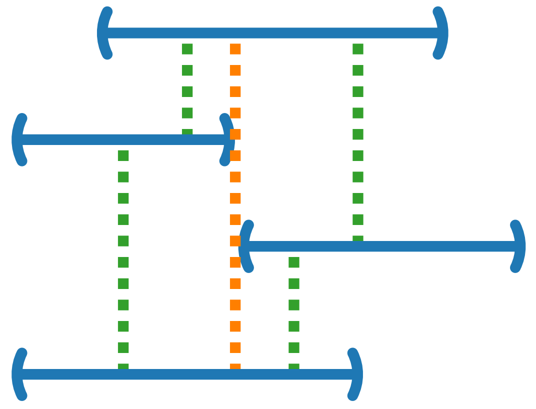
- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.
- **Epsilon:**
  Edge $uv \Leftrightarrow$ ε-wide vertical lines of sight for some $\varepsilon > 0$.
- **Weak:**
  Edge $uv \Rightarrow$ unobstructed vertical lines of sight exists,
  i.e., any subset of *visible* pairs



Bar Visibility Representation

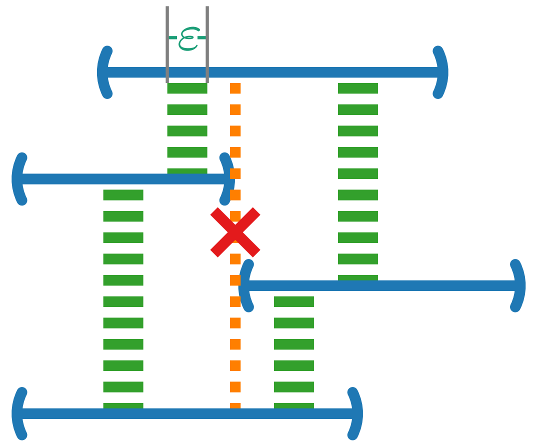
- Vertices correspond to horizontal (open) line segments called **bars**.
- **Edges** correspond to unobstructed vertical lines of sight.
- What about unobstructed **0-width** vertical lines of sight? Do all visibilities induce edges?

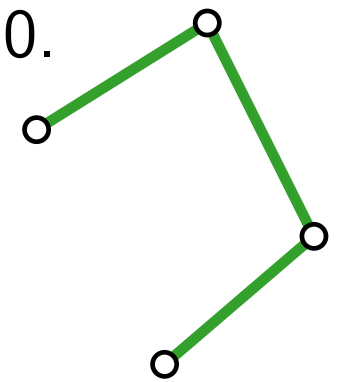




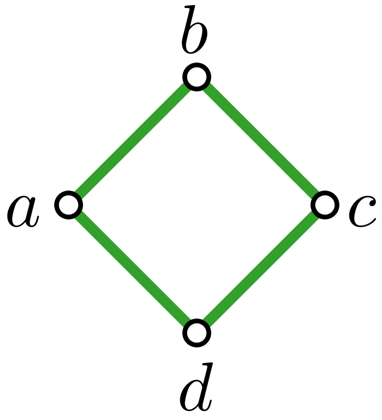
Models.

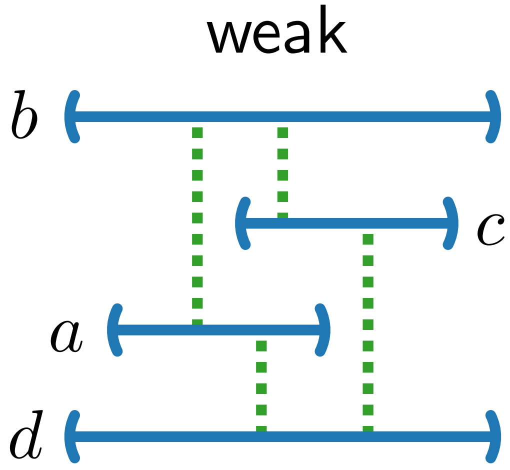
- **Strong:**
  Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical lines of sight.
- **Epsilon:**
  Edge $uv \Leftrightarrow$ ε-wide vertical lines of sight for some $\varepsilon > 0$.
- **Weak:**
  Edge $uv \Rightarrow$ unobstructed vertical lines of sight exists, i.e., any subset of *visible* pairs



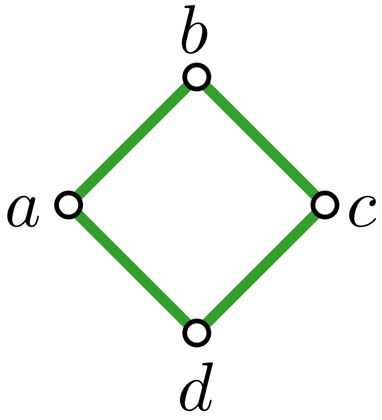


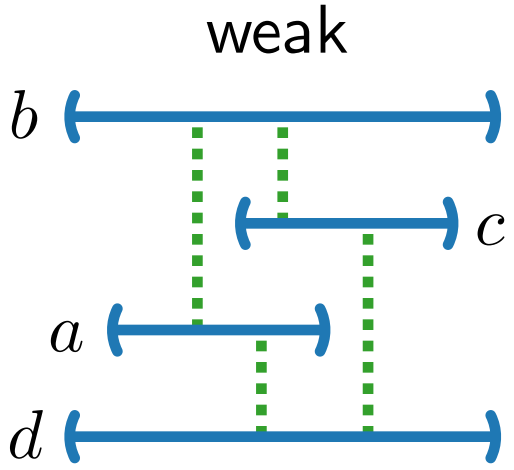
Problems

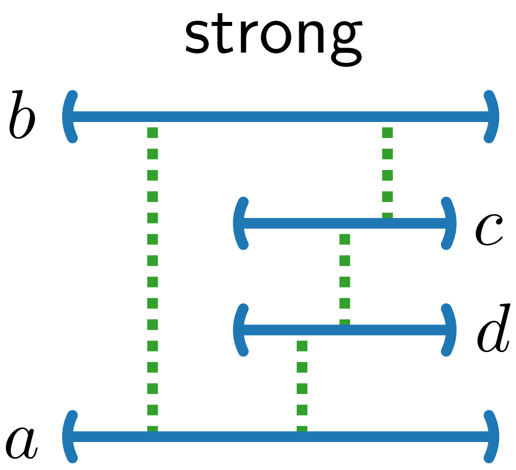




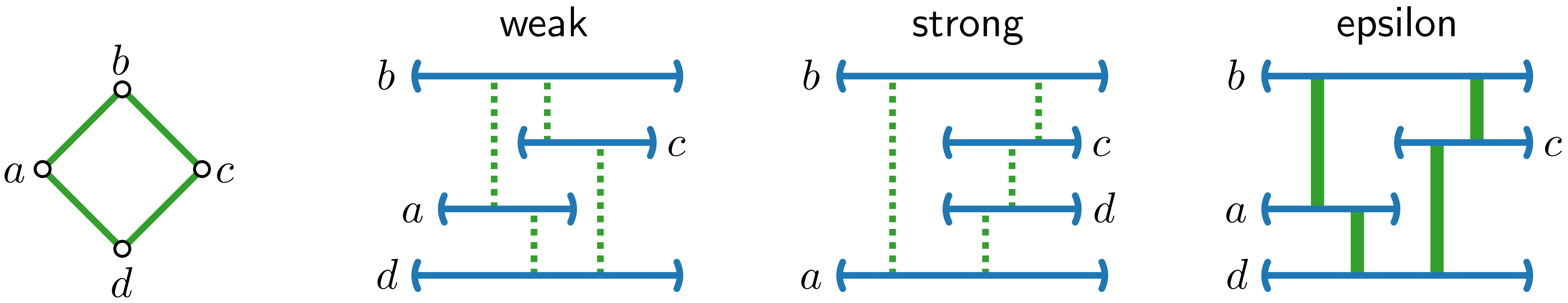
Problems



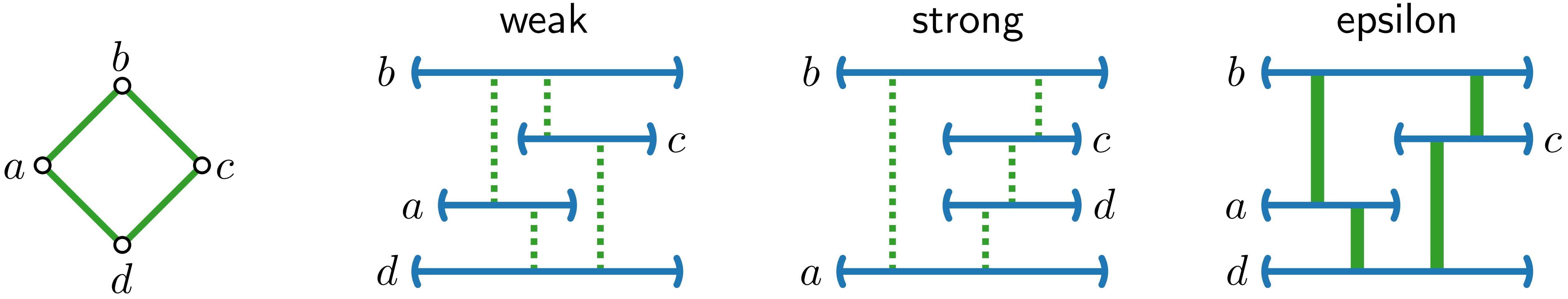




Problems

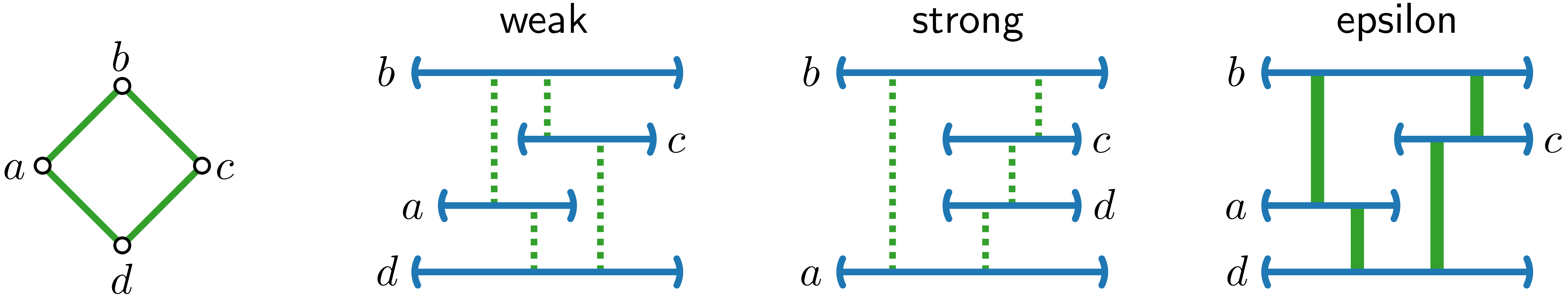


Problems



Recognition Problem.
Given a graph G, **decide** whether there exists a weak/strong/ε-bar visibility representation ψ of G.

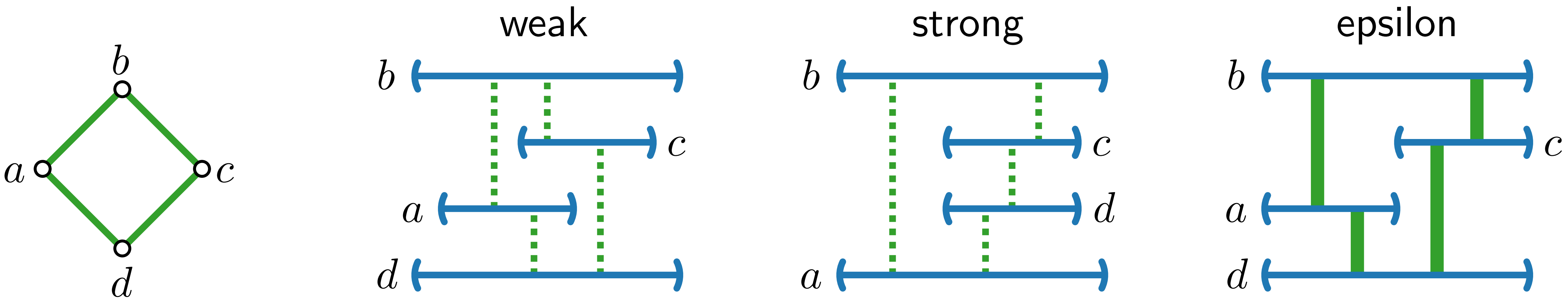
Problems



Recognition Problem.
Given a graph G, **decide** whether there exists a weak/strong/ε-bar visibility representation ψ of G.

Construction Problem.
Given a graph G, **construct** a weak/strong/ε-bar visibility representation ψ of G – if one exists.

Problems

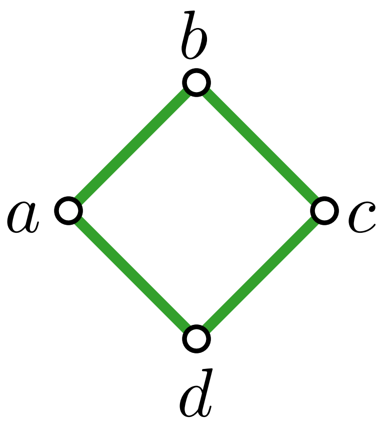


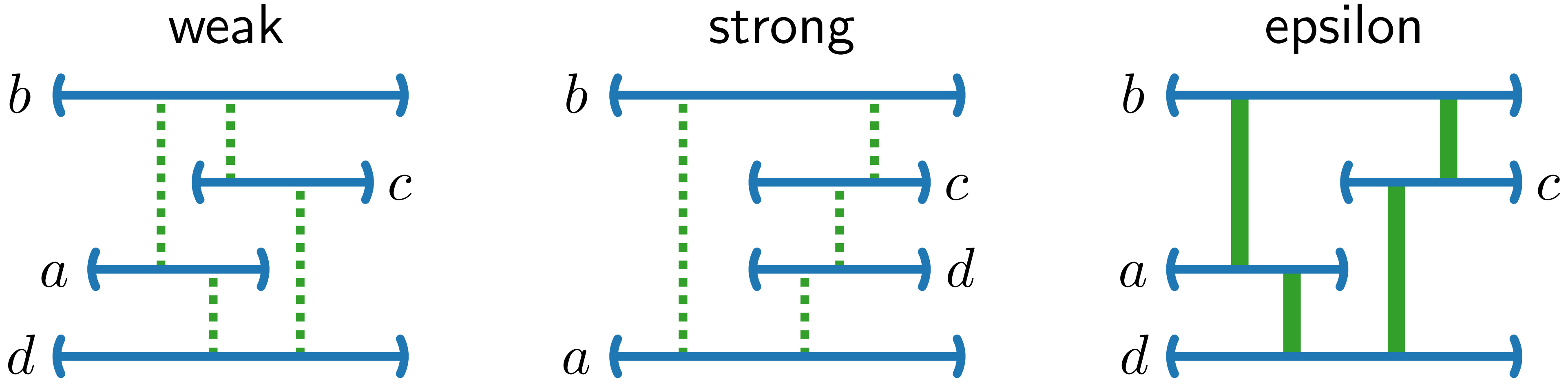
Recognition Problem.
Given a graph G, **decide** whether there exists a weak/strong/ε-bar visibility representation ψ of G.

Construction Problem.
Given a graph G, **construct** a weak/strong/ε-bar visibility representation ψ of G – if one exists.

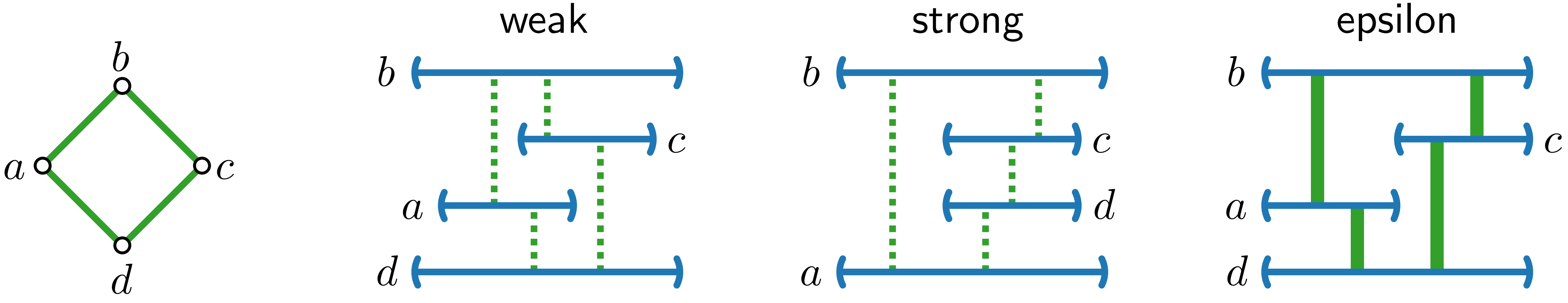
Partial Representation Extension Problem.
Given a graph G and a **set of bars ψ' of $V' \subseteq V(G)$**, **decide** whether there exists a weak/strong/ε-bar visibility representation ψ of G **where $\psi|_{V'} = \psi'$** (and **construct** ψ if a representation exists).

Background



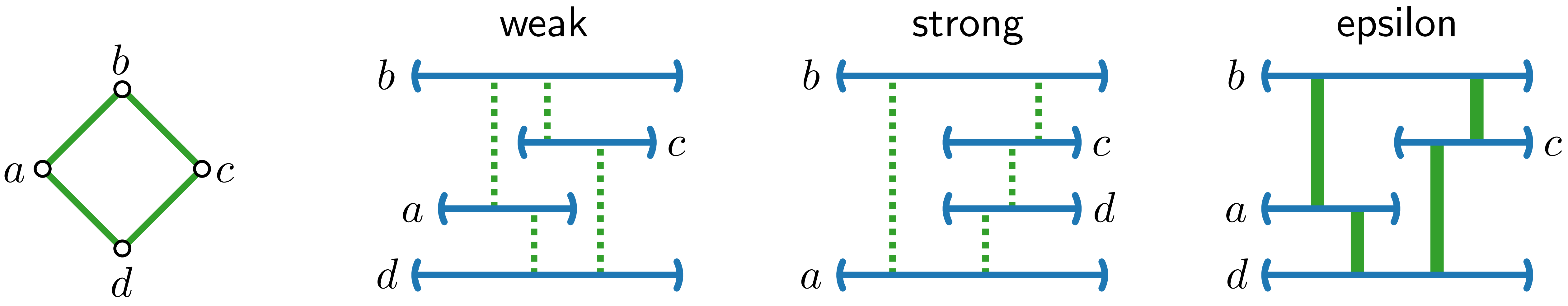


Background



Weak Bar Visibility.

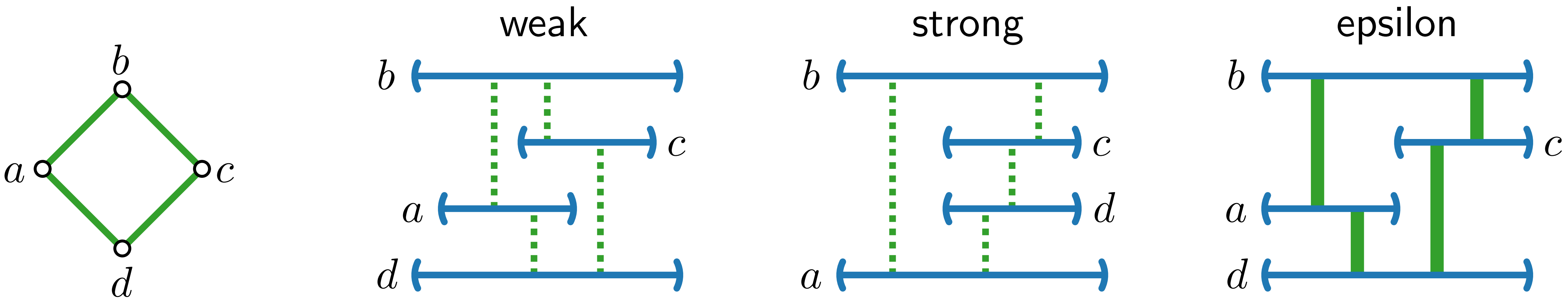
Background



Weak Bar Visibility.

- Exactly all planar graphs [Tamassia & Tollis '86; Wismath '85]

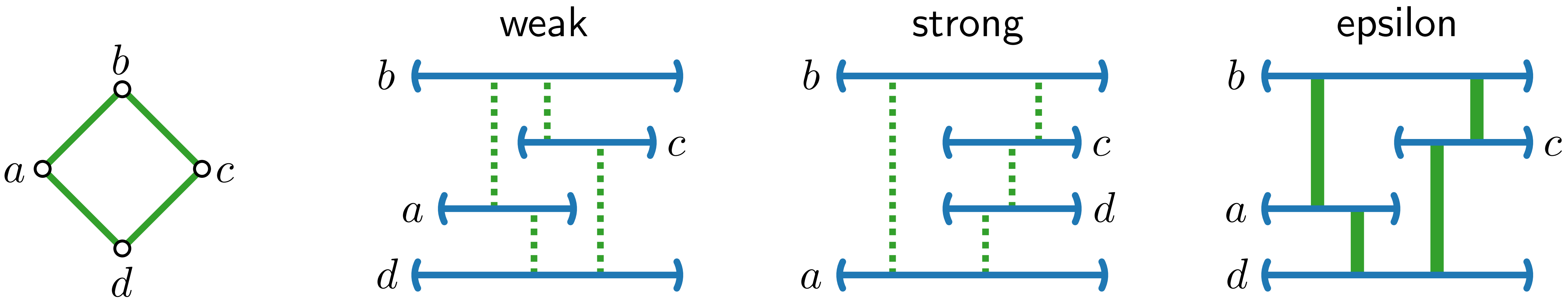
Background



Weak Bar Visibility.

- Exactly all planar graphs [Tamassia & Tollis '86; Wismath '85]
- Linear-time recognition and construction [T&T '86]

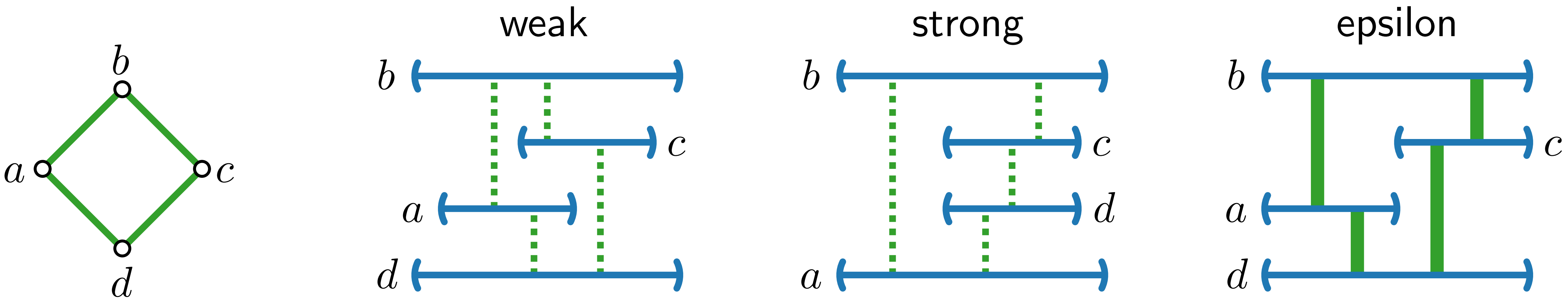
Background



Weak Bar Visibility.

- Exactly all planar graphs [Tamassia & Tollis '86; Wismath '85]
- Linear-time recognition and construction [T&T '86]
- Representation extension is NP-complete [Chaplick et al. '14]

Background

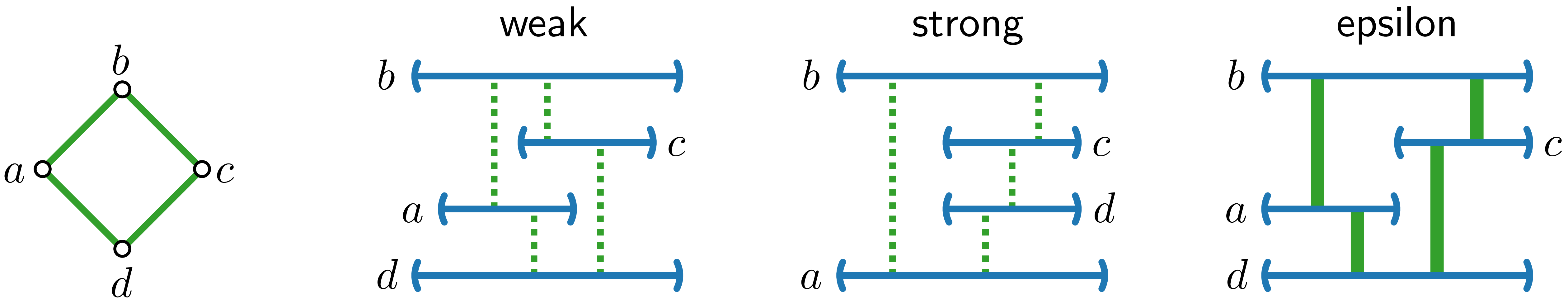


Weak Bar Visibility.

- Exactly all planar graphs [Tamassia & Tollis '86; Wismath '85]
- Linear-time recognition and construction [T&T '86]
- Representation extension is NP-complete [Chaplick et al. '14]

Strong Bar Visibility.

Background



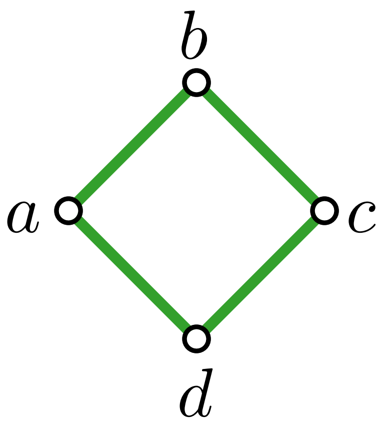
Weak Bar Visibility.

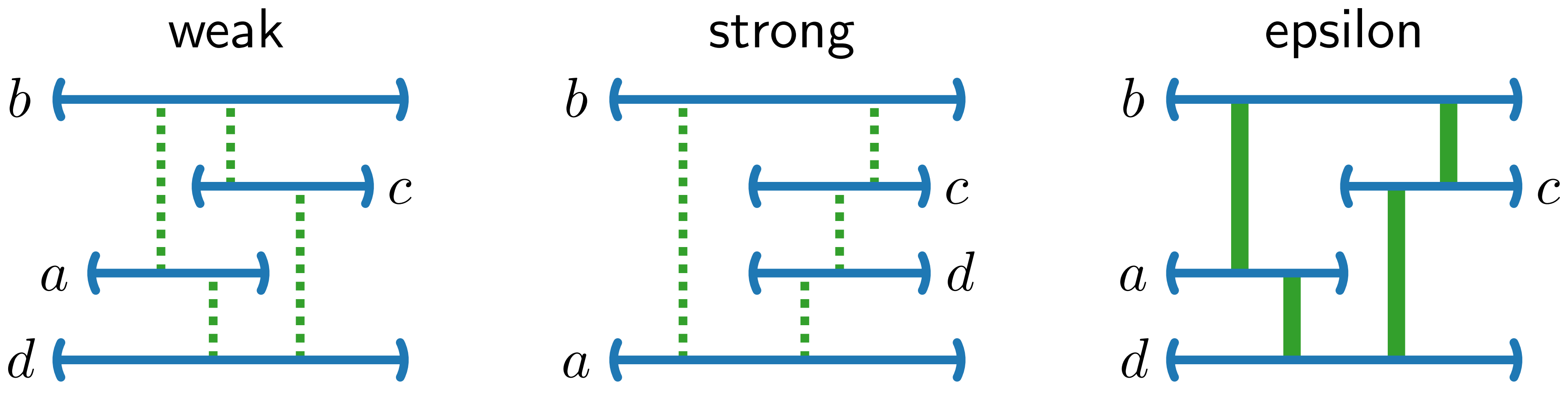
- Exactly all planar graphs [Tamassia & Tollis '86; Wismath '85]
- Linear-time recognition and construction [T&T '86]
- Representation extension is NP-complete [Chaplick et al. '14]

Strong Bar Visibility.

- NP-complete to recognize [Andreae '92]

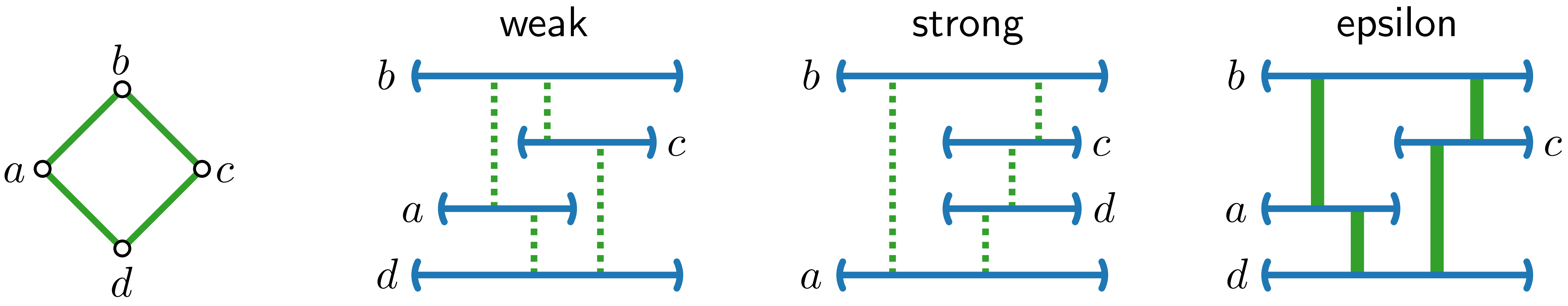
Background





ε-Bar Visibility.

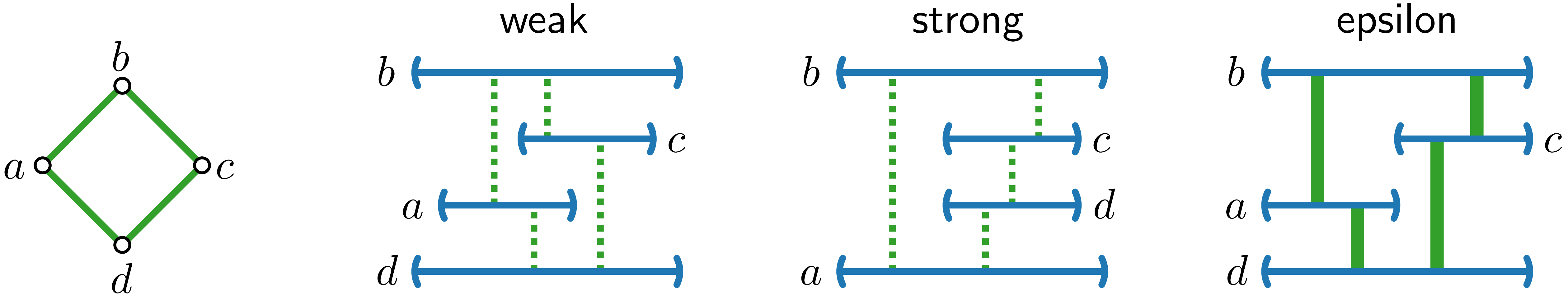
Background



ε-Bar Visibility.

- Exactly all planar graphs that can be embedded with all **cut vertices** on the outer face [T&T '86, Wismath '85]

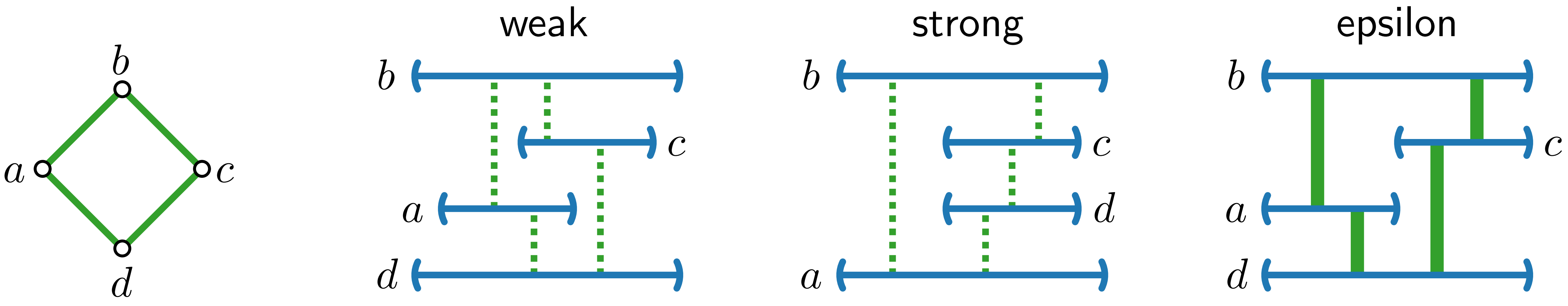
Background



ε-Bar Visibility.

- Exactly all planar graphs that can be embedded with all **cut vertices** on the outer face [T&T '86, Wismath '85]
- Linear-time recognition and construction [T&T '86]

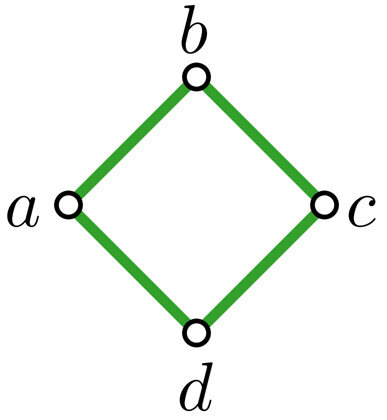
Background

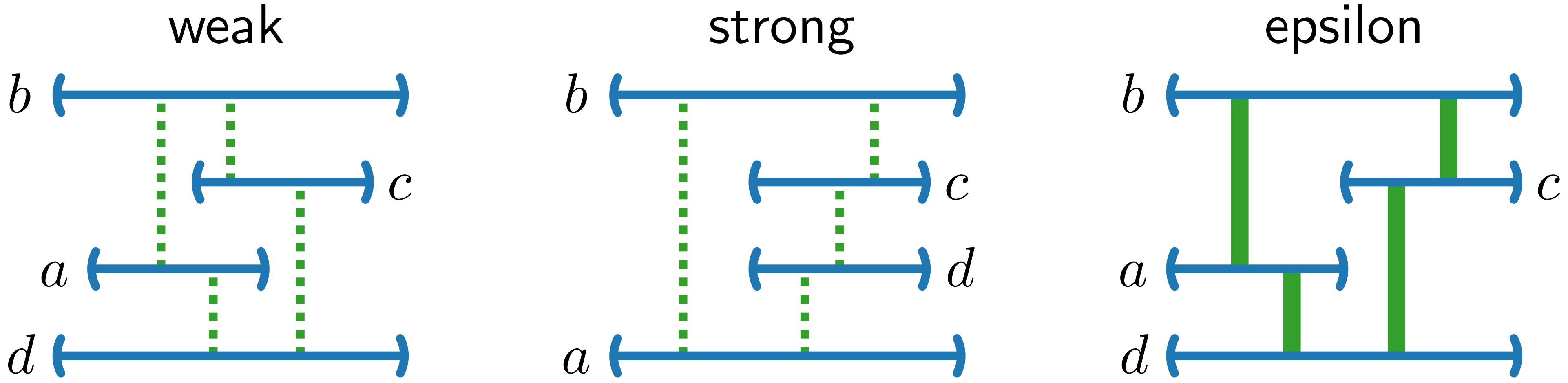


ε-Bar Visibility.

- Exactly all planar graphs that can be embedded with all **cut vertices** on the outer face [T&T '86, Wismath '85]
- Linear-time recognition and construction [T&T '86]
- Representation extension?

Background





ε-Bar Visibility.

- Exactly all planar graphs that can be embedded with all **cut vertices** on the outer face [T&T '86, Wismath '85]
- Linear-time recognition and construction [T&T '86]
- Representation extension? This Lecture!

Bar Visibility Representation of Digraphs

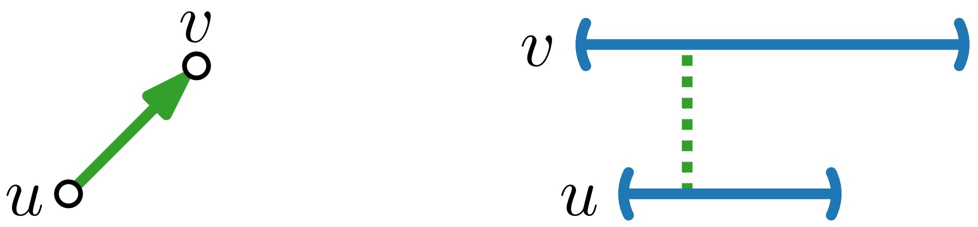
- Instead of an undirected graph, we are given a directed graph G.

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...

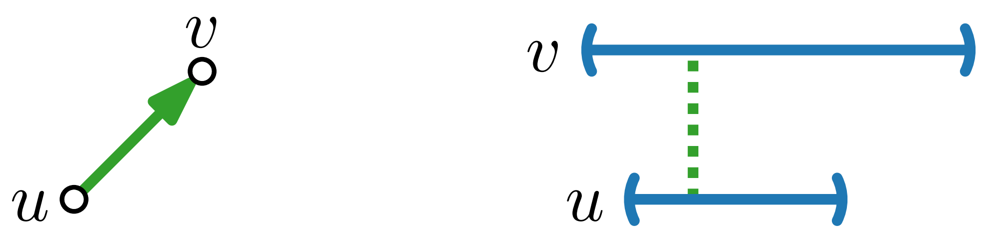
Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.



Bar Visibility Representation of Digraphs

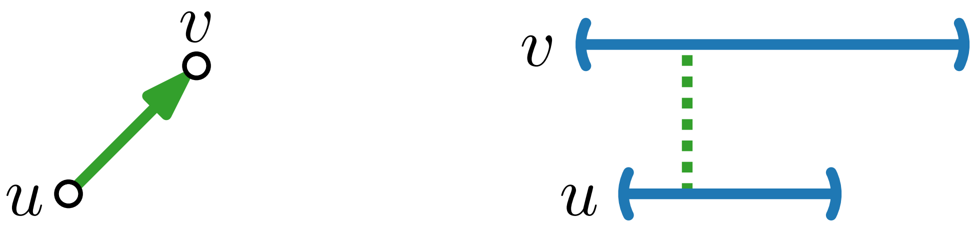
- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.



Weak Bar Visibility.

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.

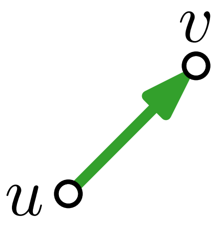


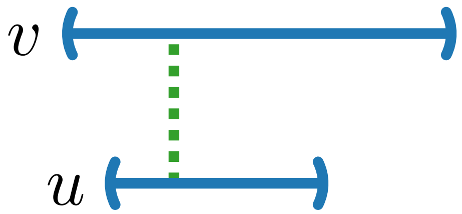
Weak Bar Visibility.

- NP-complete for directed (acyclic planar) graphs!

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.



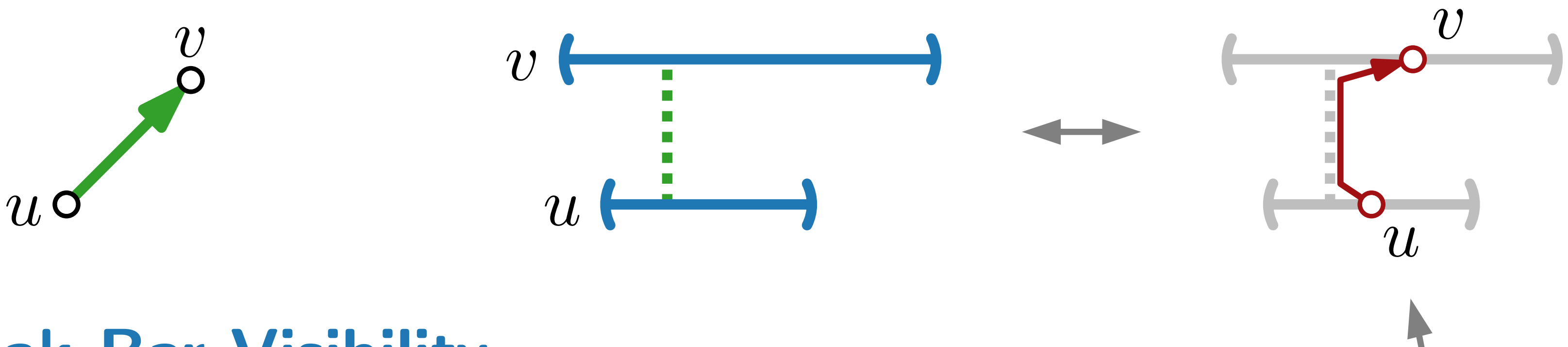


Weak Bar Visibility.

- NP-complete for directed (acyclic planar) graphs!
- This is because upward planarity testing is NP-complete. [Garg & Tamassia '01]

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.

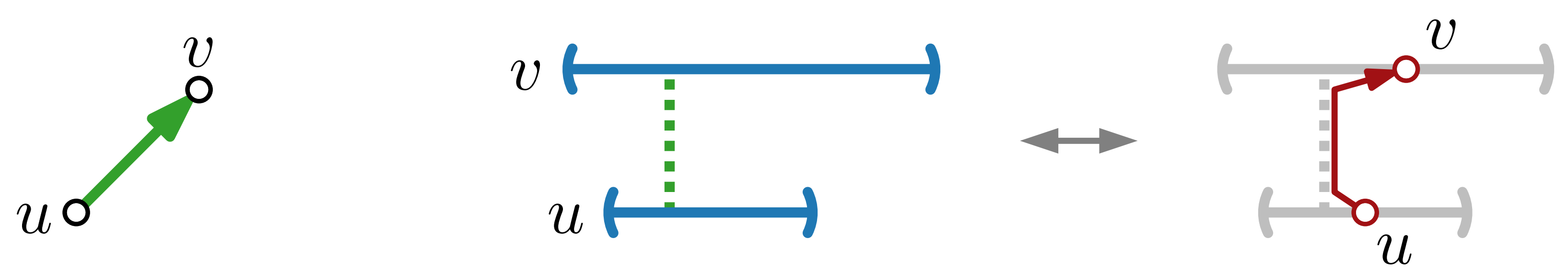


Weak Bar Visibility.

- NP-complete for directed (acyclic planar) graphs!
- This is because upward planarity testing is NP-complete. [Garg & Tamassia '01]

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that …
- …for each directed edge uv, the bar representing u is below the bar representing v.



Weak Bar Visibility.

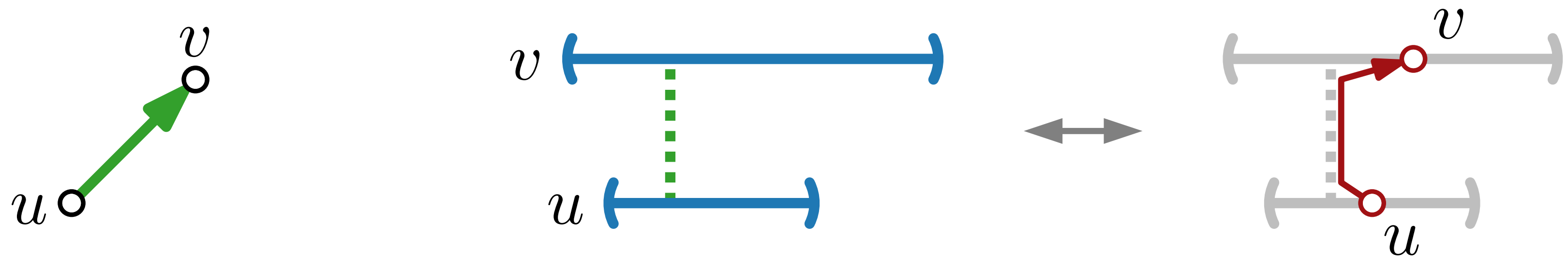
- NP-complete for directed (acyclic planar) graphs!
- This is because upward planarity testing is NP-complete. [Garg & Tamassia '01]

Strong/ε-Bar Visibility.

- Open for directed graphs!

Bar Visibility Representation of Digraphs

- Instead of an undirected graph, we are given a directed graph G.
- The task is to construct a weak/strong/ε-bar visibility representation of G such that ...
- ... for each directed edge uv, the bar representing u is below the bar representing v.



Weak Bar Visibility.

- NP-complete for directed (acyclic planar) graphs!
- This is because upward planarity testing is NP-complete. [Garg & Tamassia '01]

Strong/ε-Bar Visibility.

- Open for directed graphs!

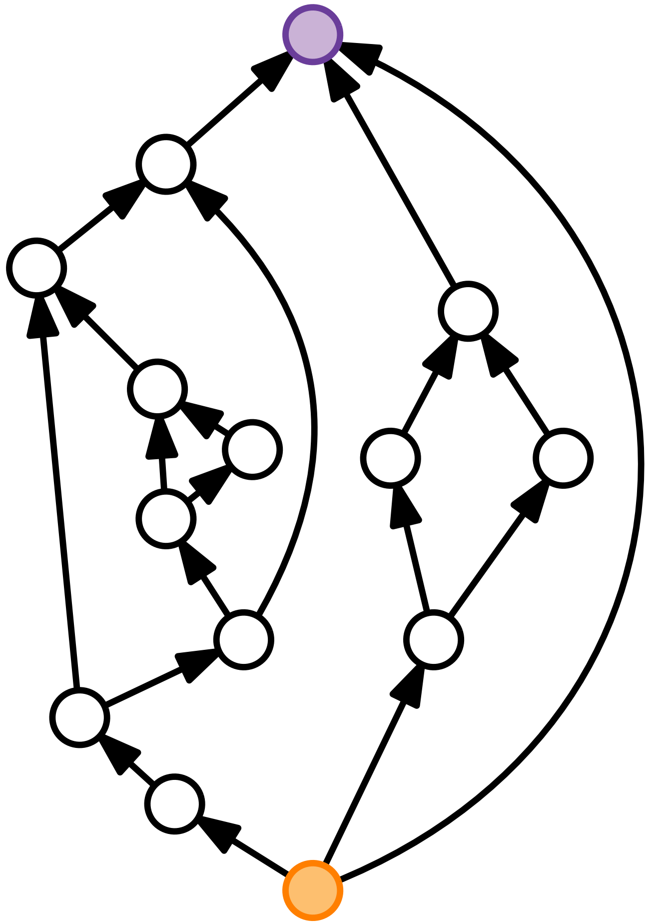
Next, we consider ε-bar visibility representations of specific directed graphs ($\rightarrow$ st-graphs)

ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

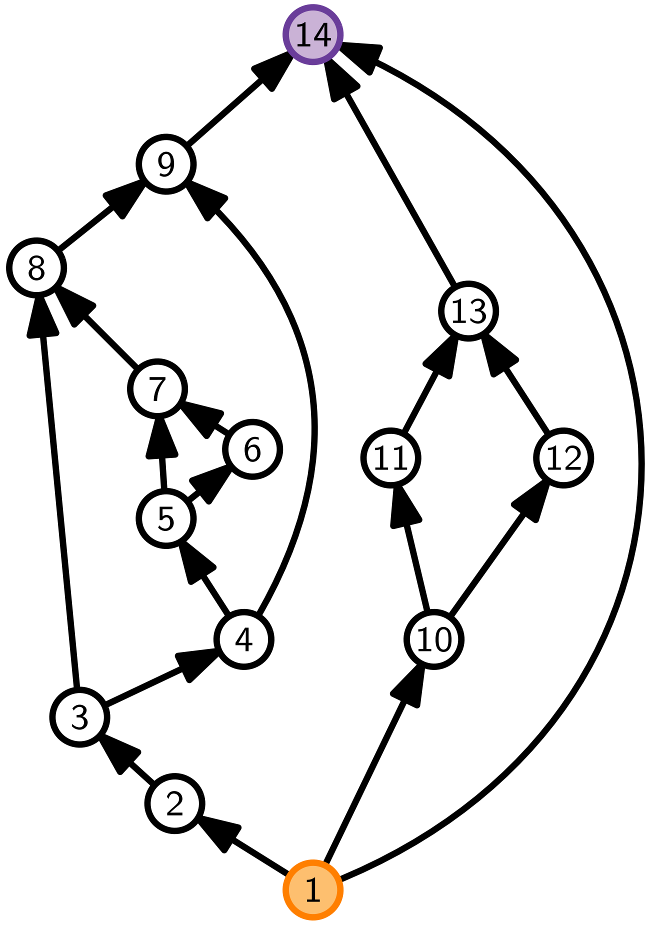
ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.



ε-Bar Visibility and st-Graphs

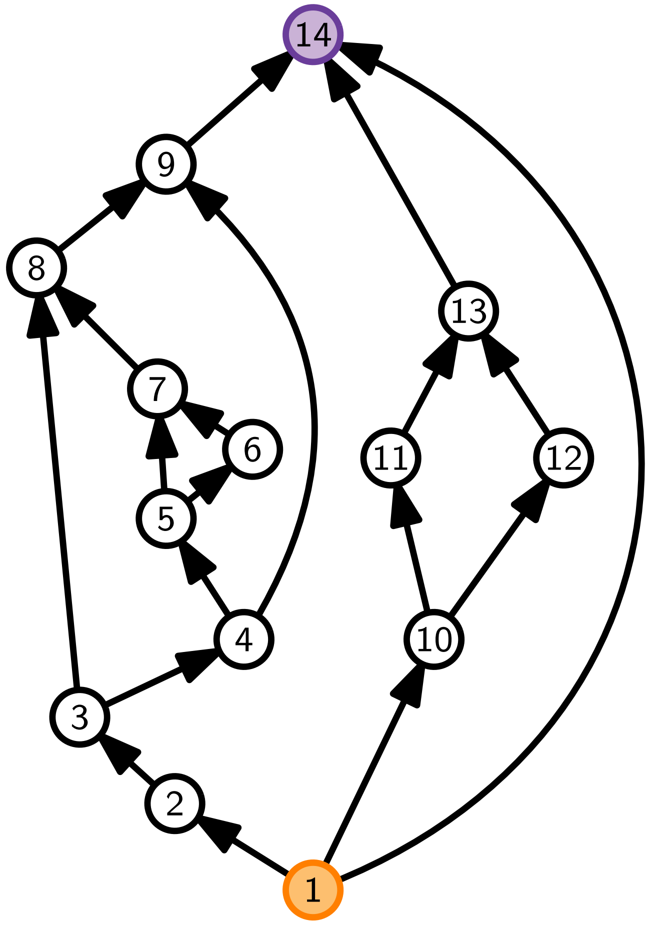
Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.



ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

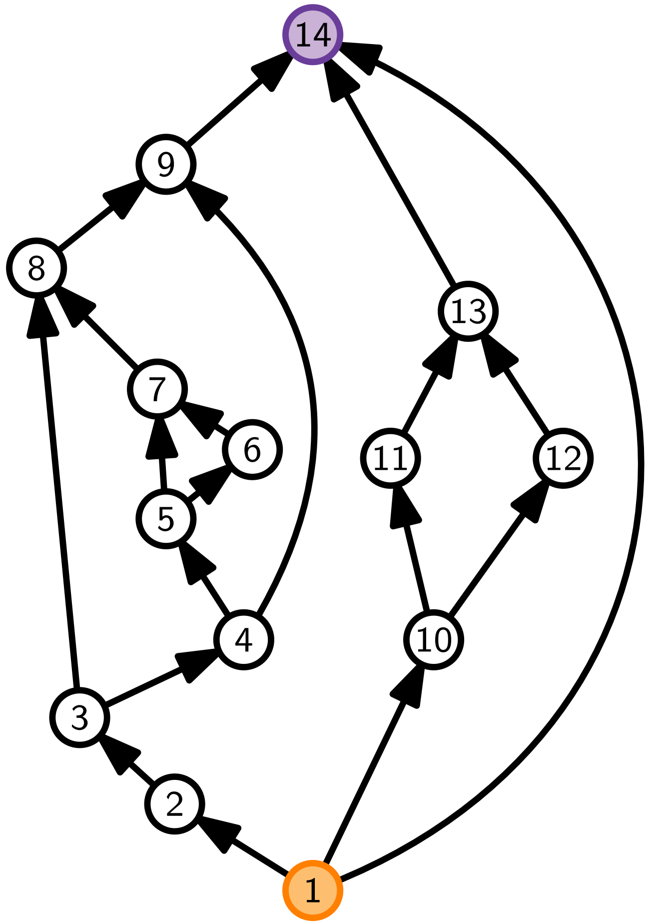
Observation.
st-orientations correspond to ε-bar visibility representations.



ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

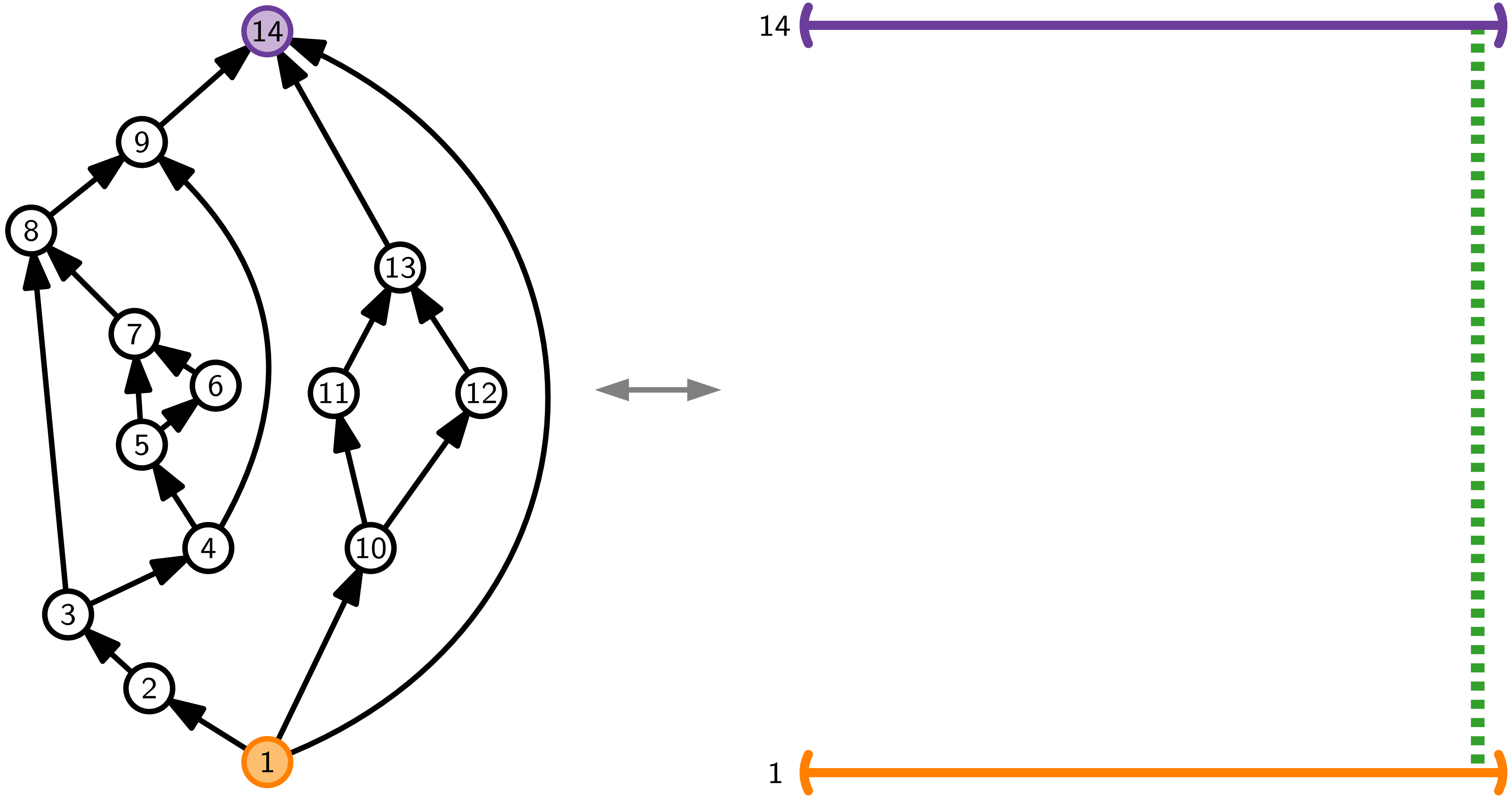
Observation.
st-orientations correspond to ε-bar visibility representations.



ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

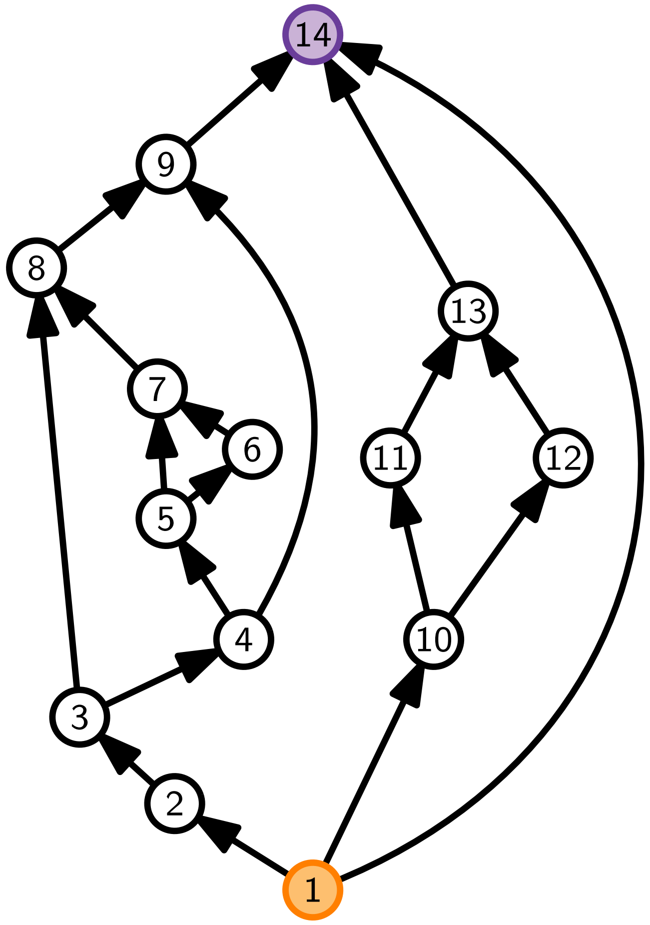
Observation.
st-orientations correspond to ε-bar visibility representations.

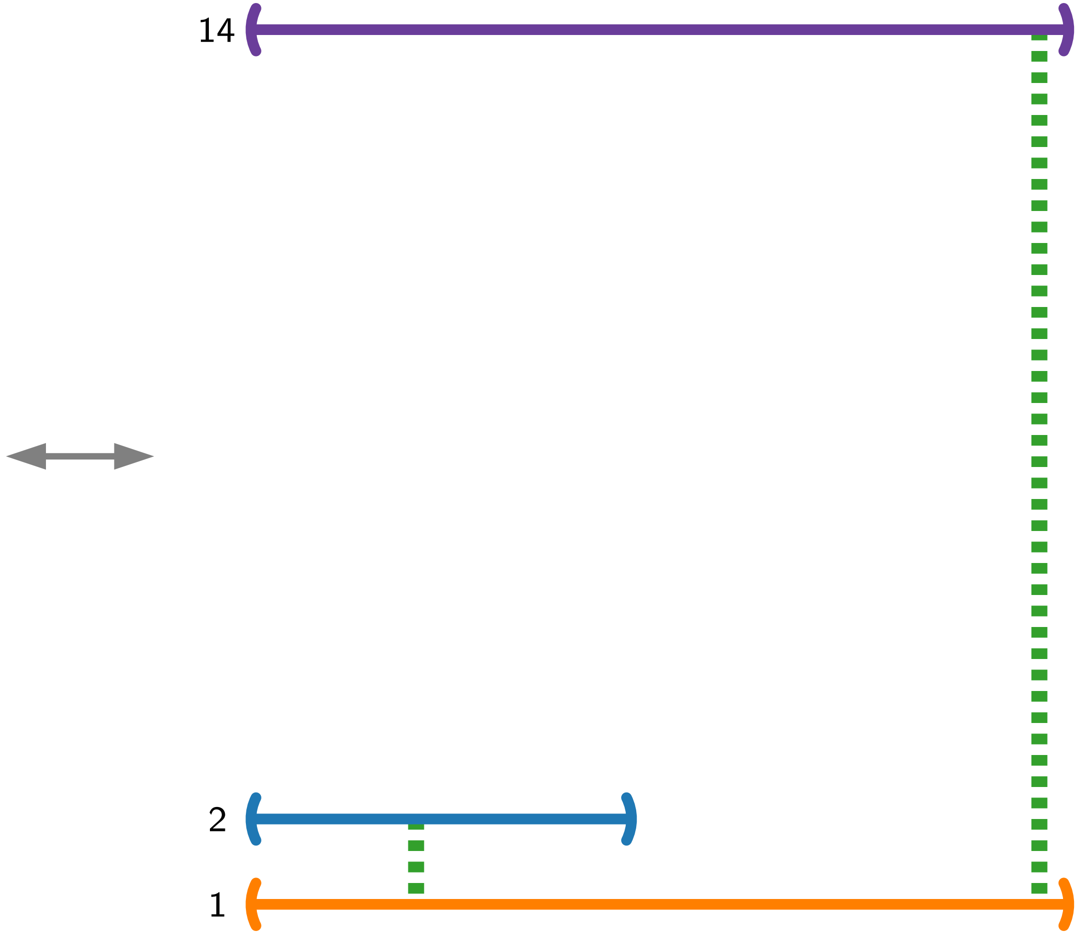


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

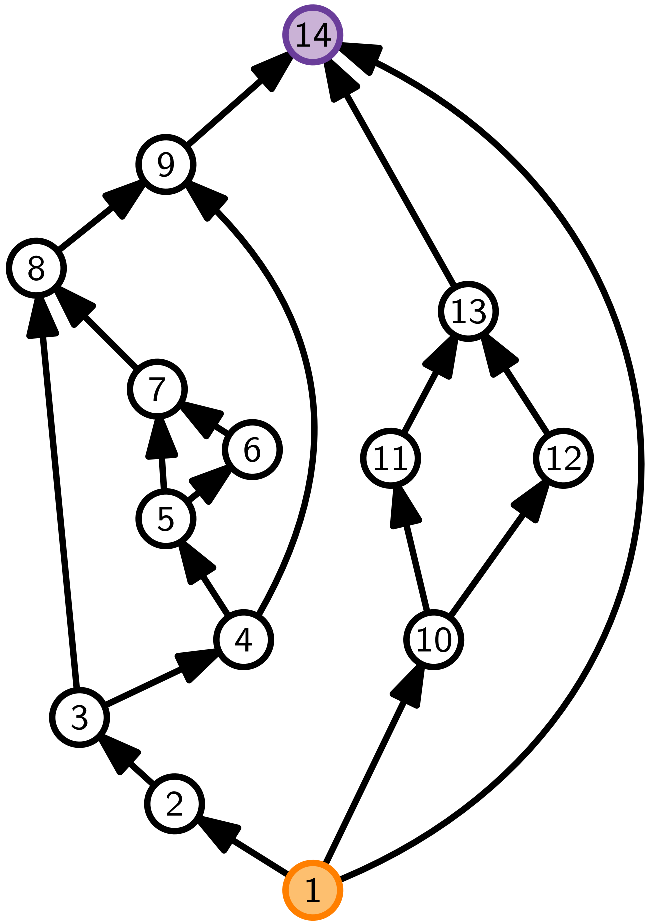


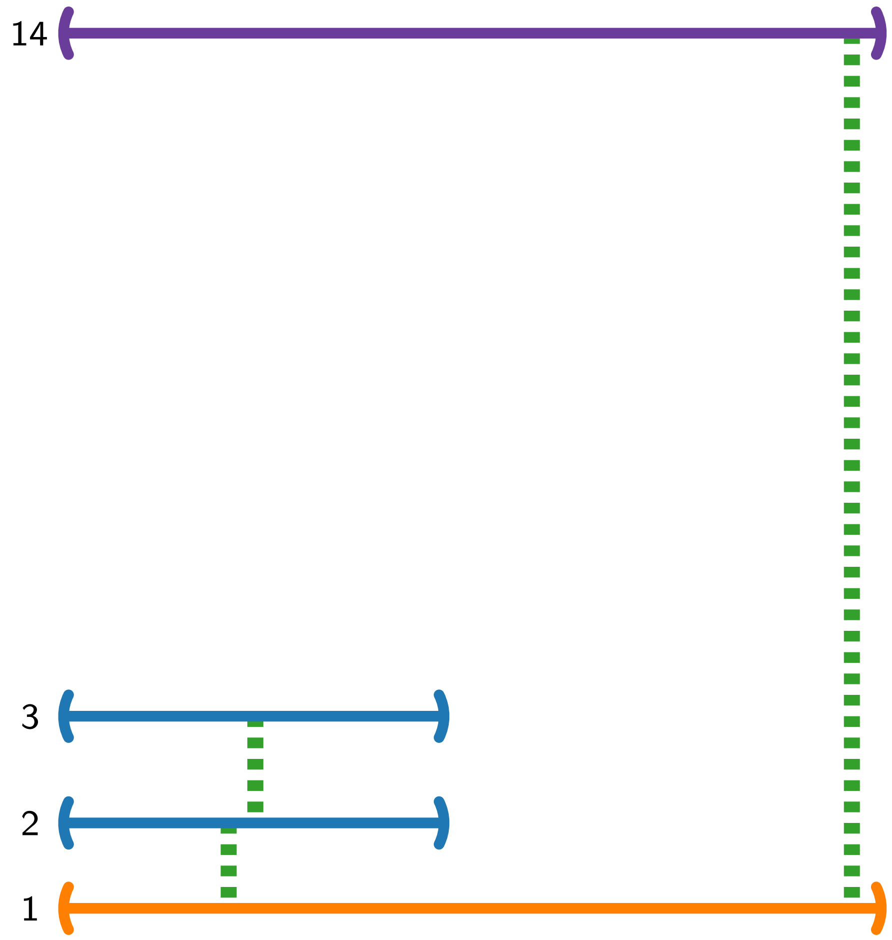


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

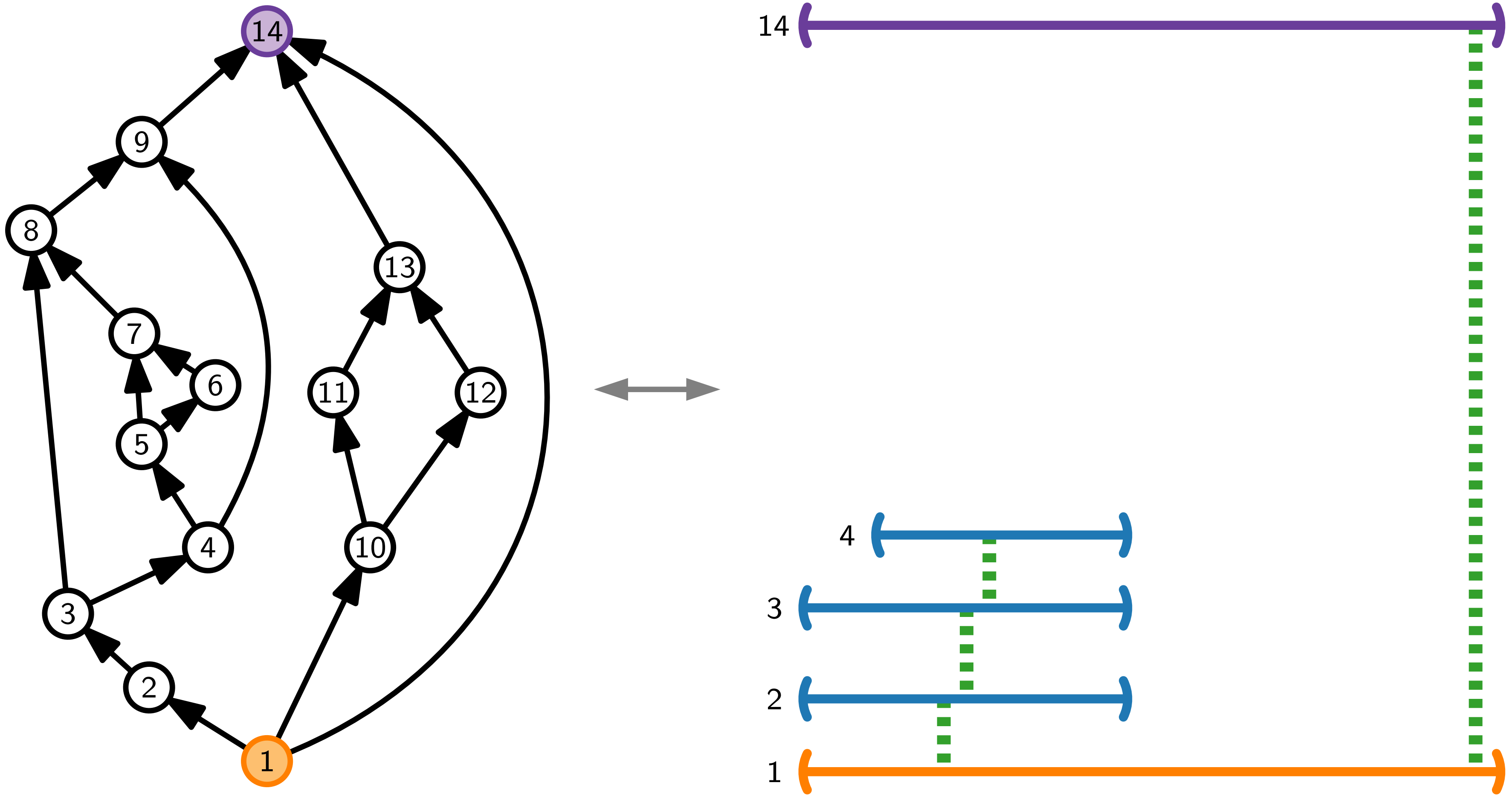




ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

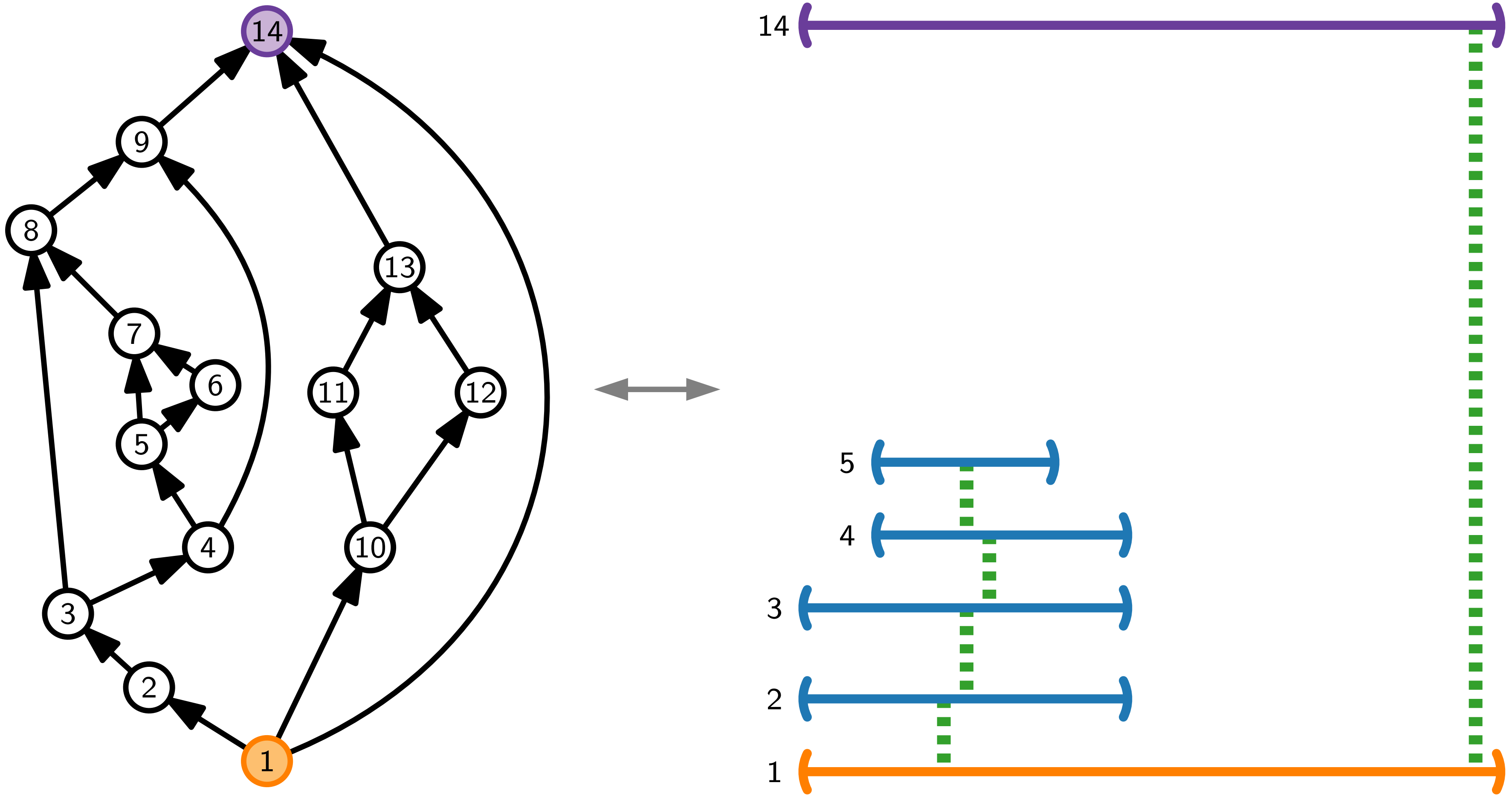
Observation.
st-orientations correspond to ε-bar visibility representations.



ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

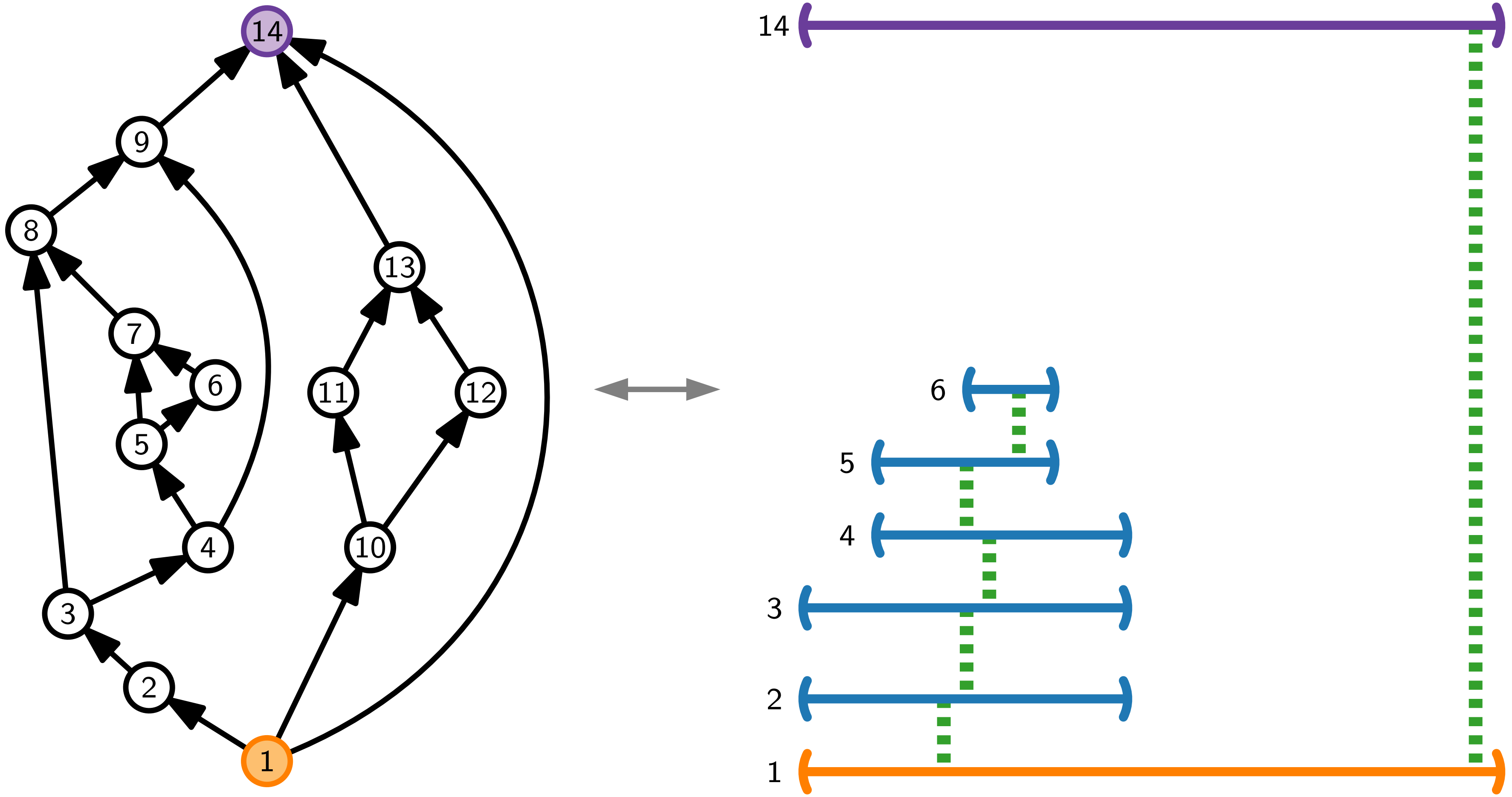
Observation.
st-orientations correspond to ε-bar visibility representations.



ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

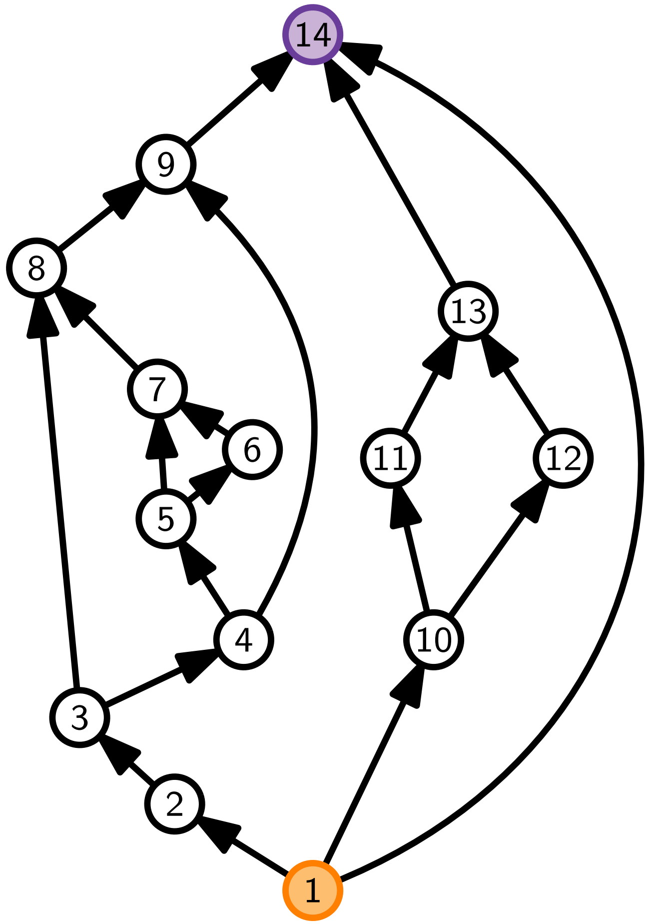
Observation.
st-orientations correspond to ε-bar visibility representations.

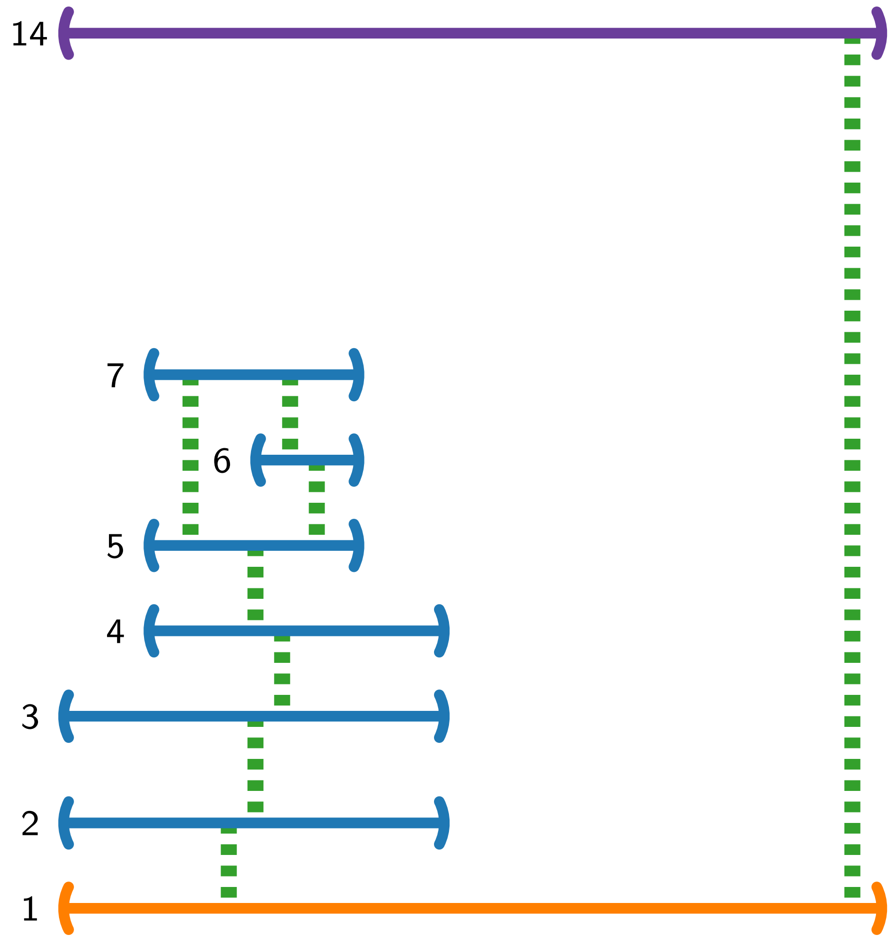


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

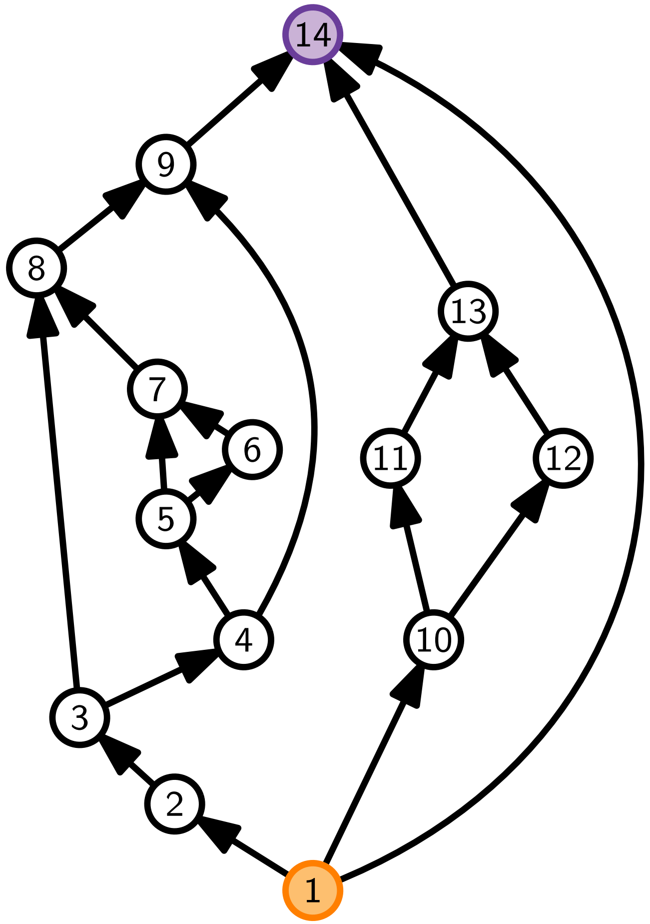


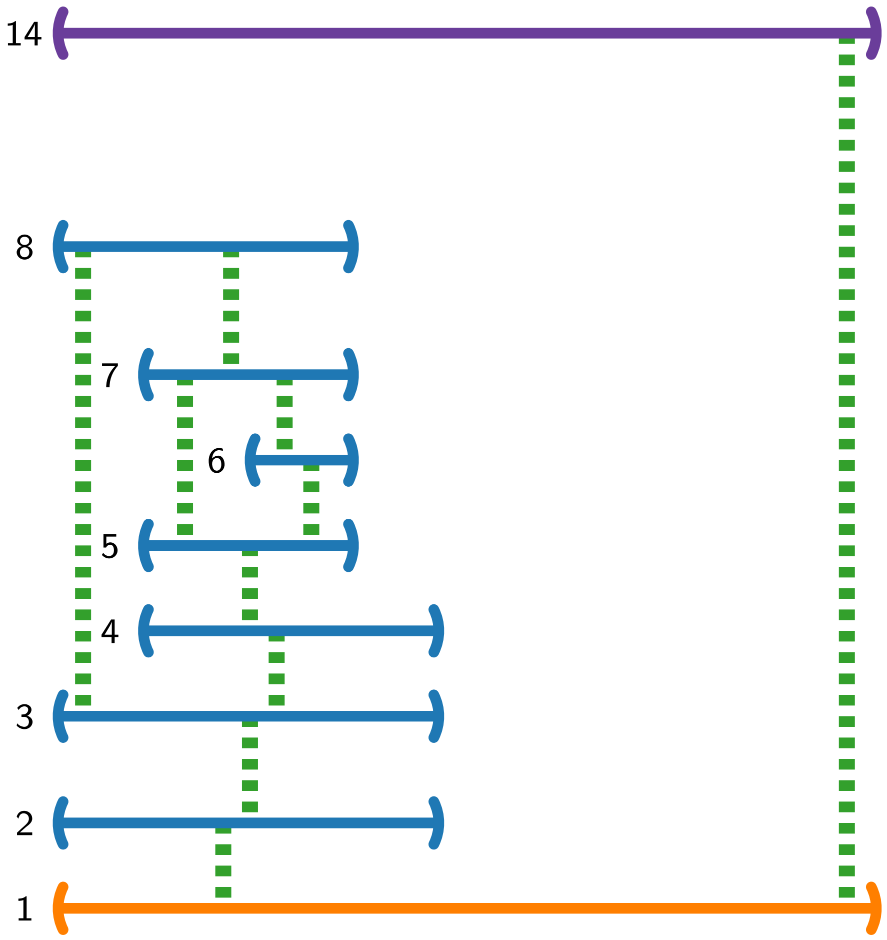


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

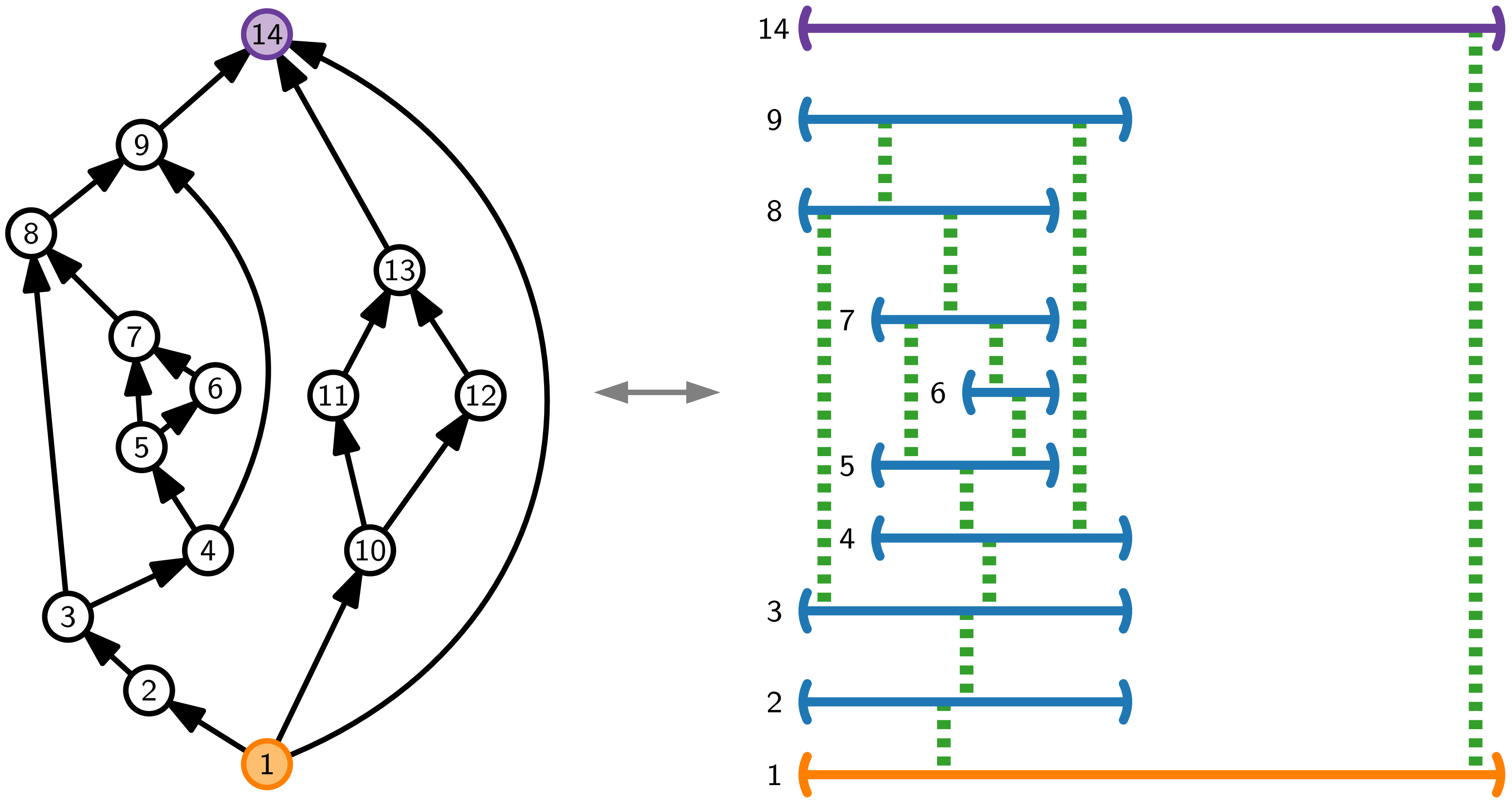




ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

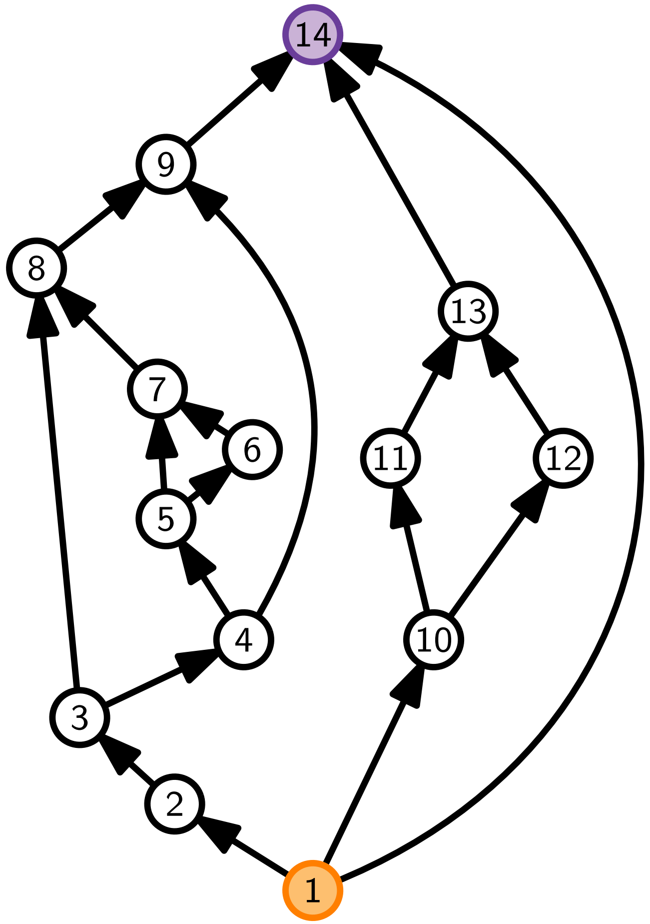
Observation.
st-orientations correspond to ε-bar visibility representations.

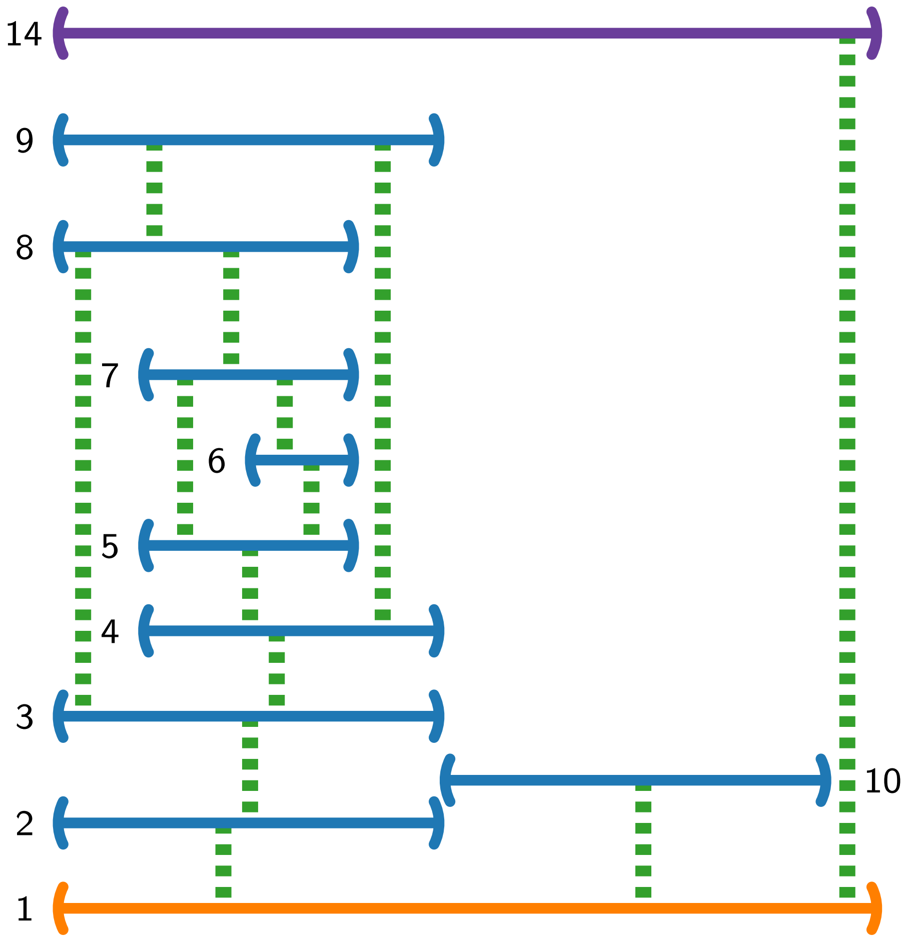


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

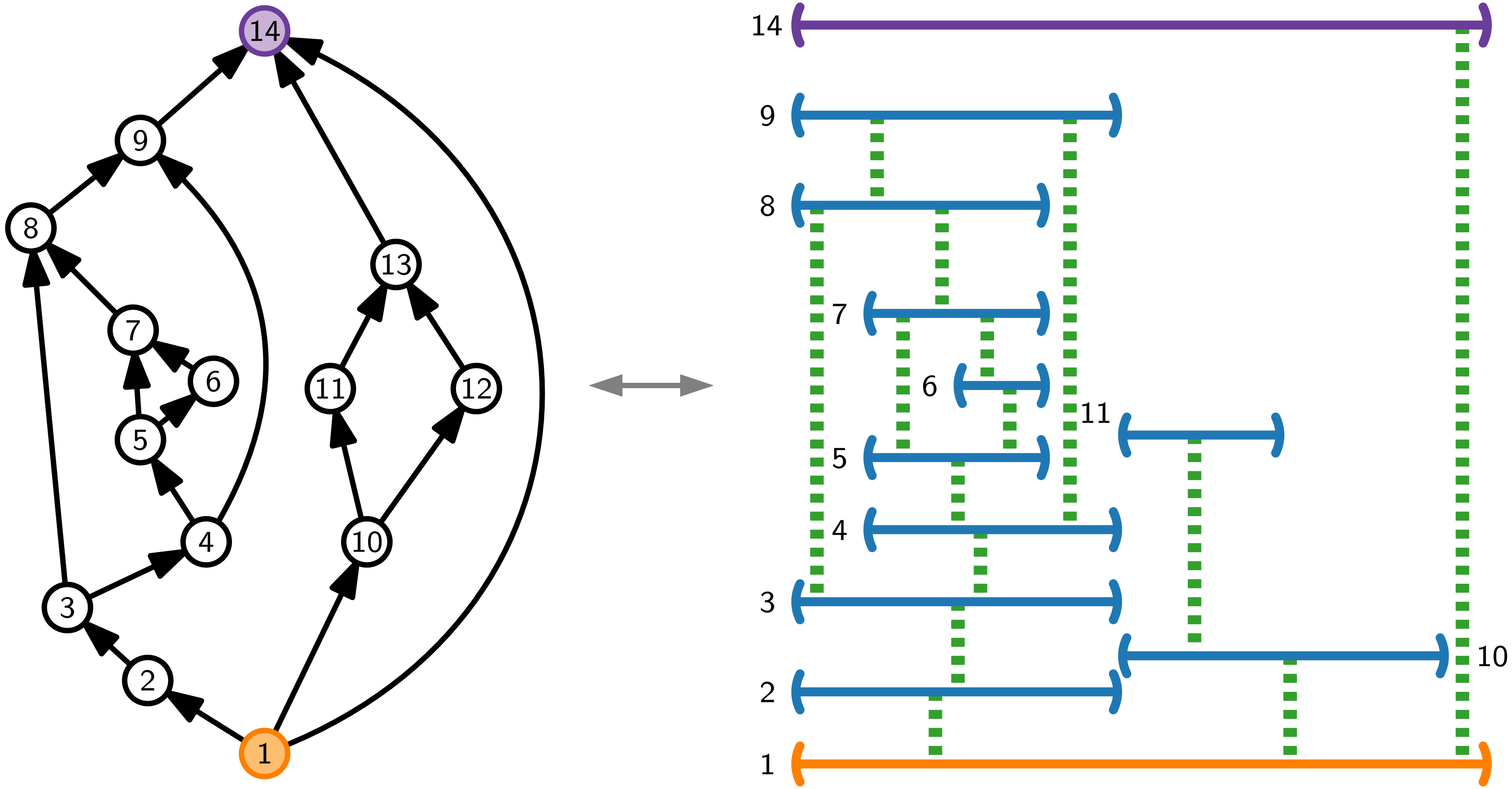




ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

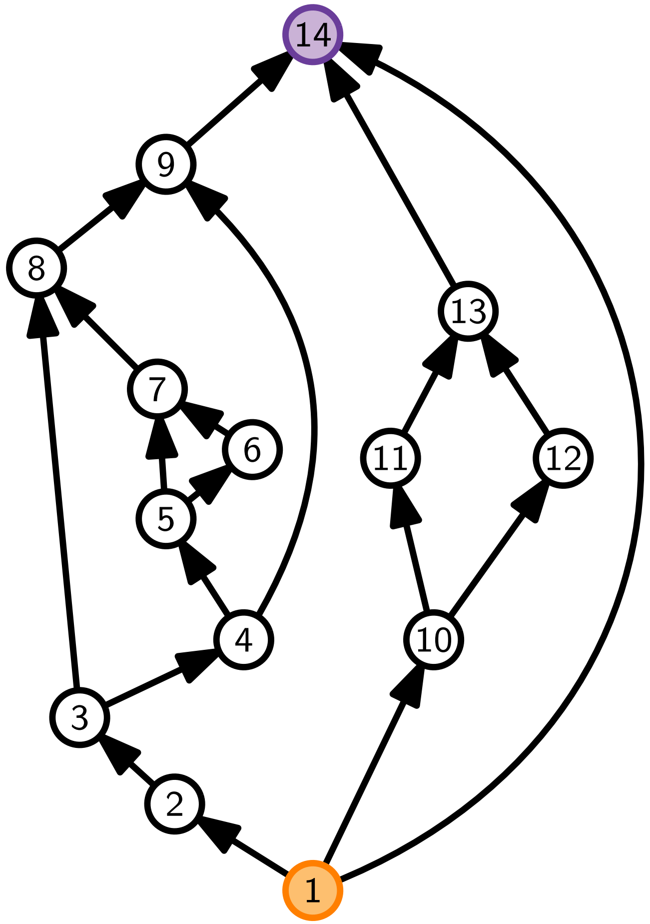
Observation.
st-orientations correspond to ε-bar visibility representations.

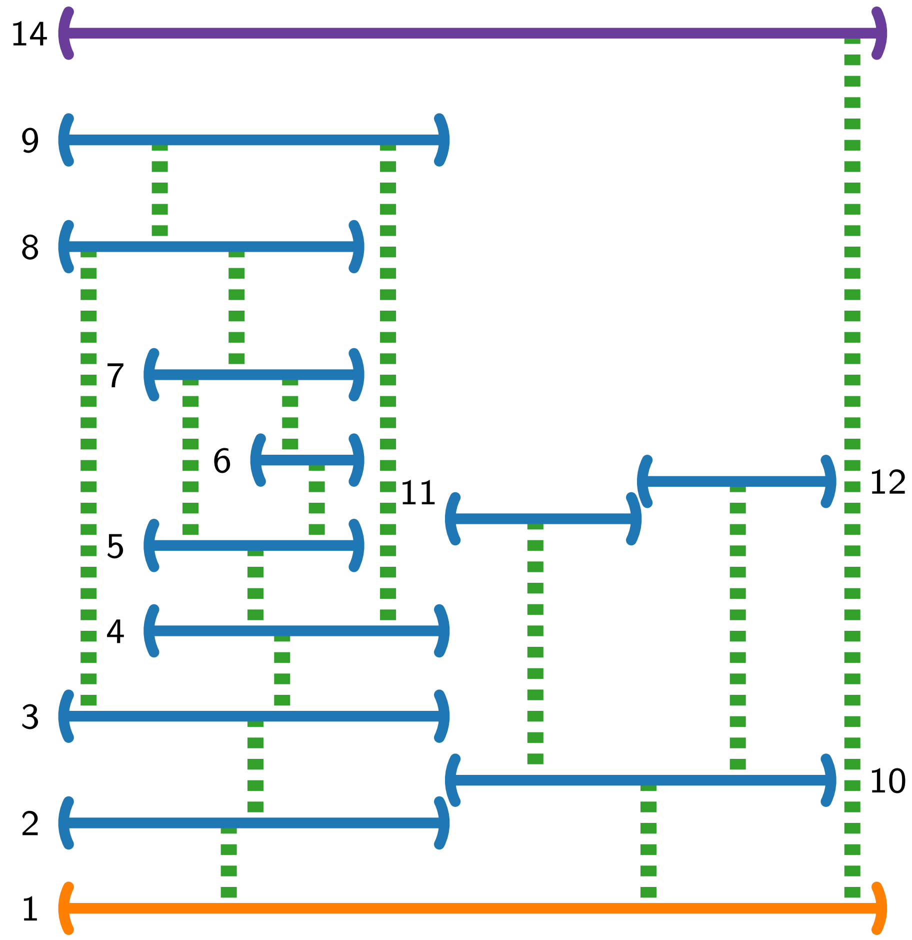


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

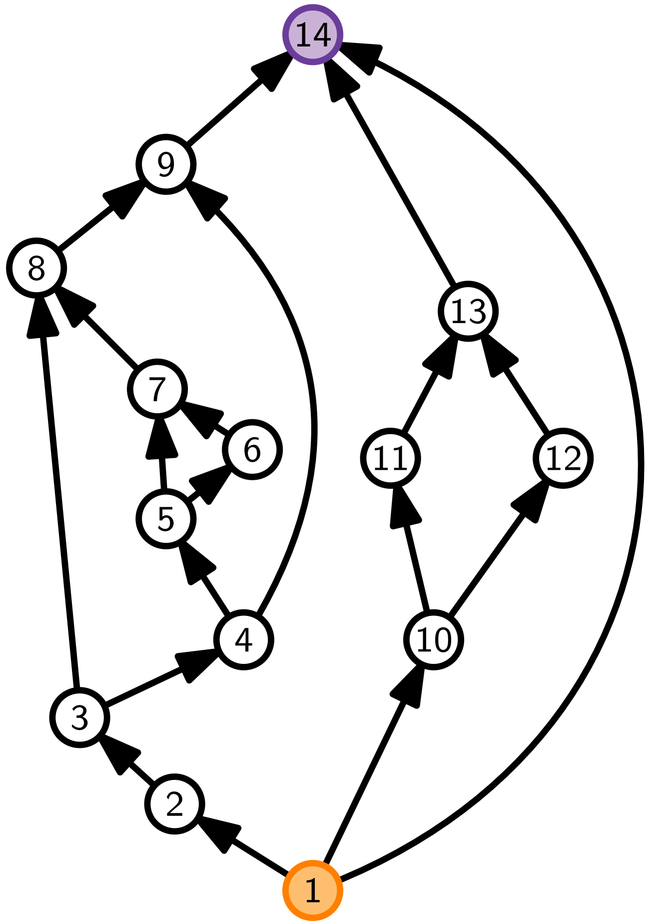


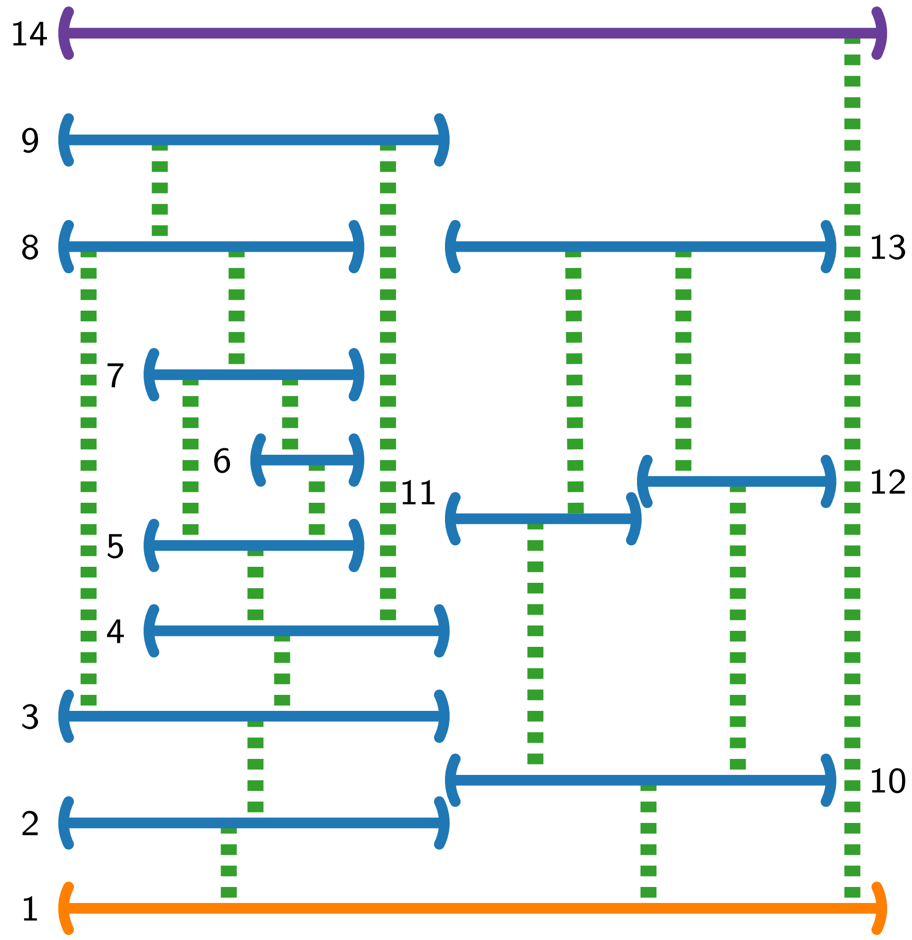


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

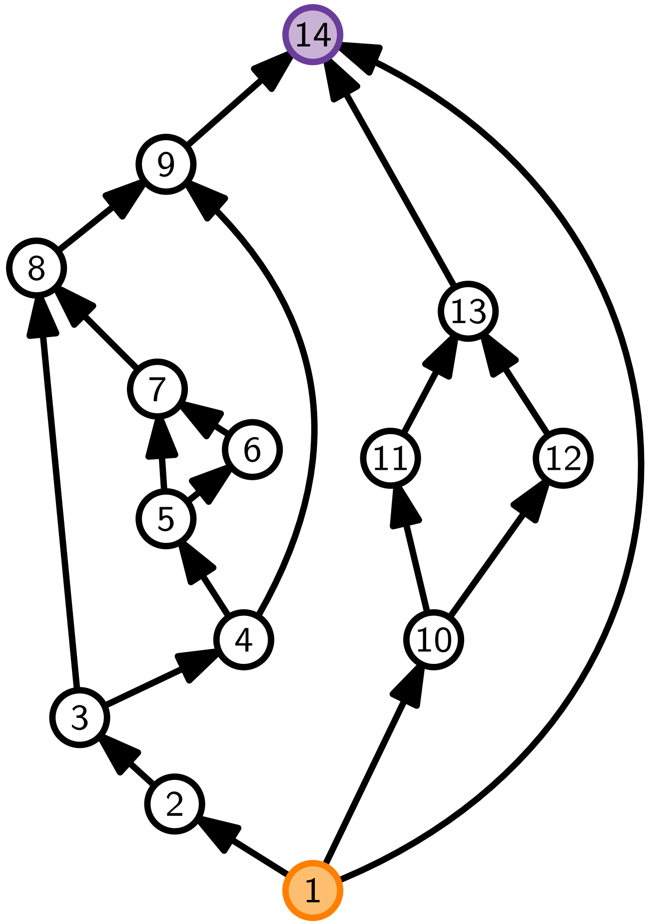


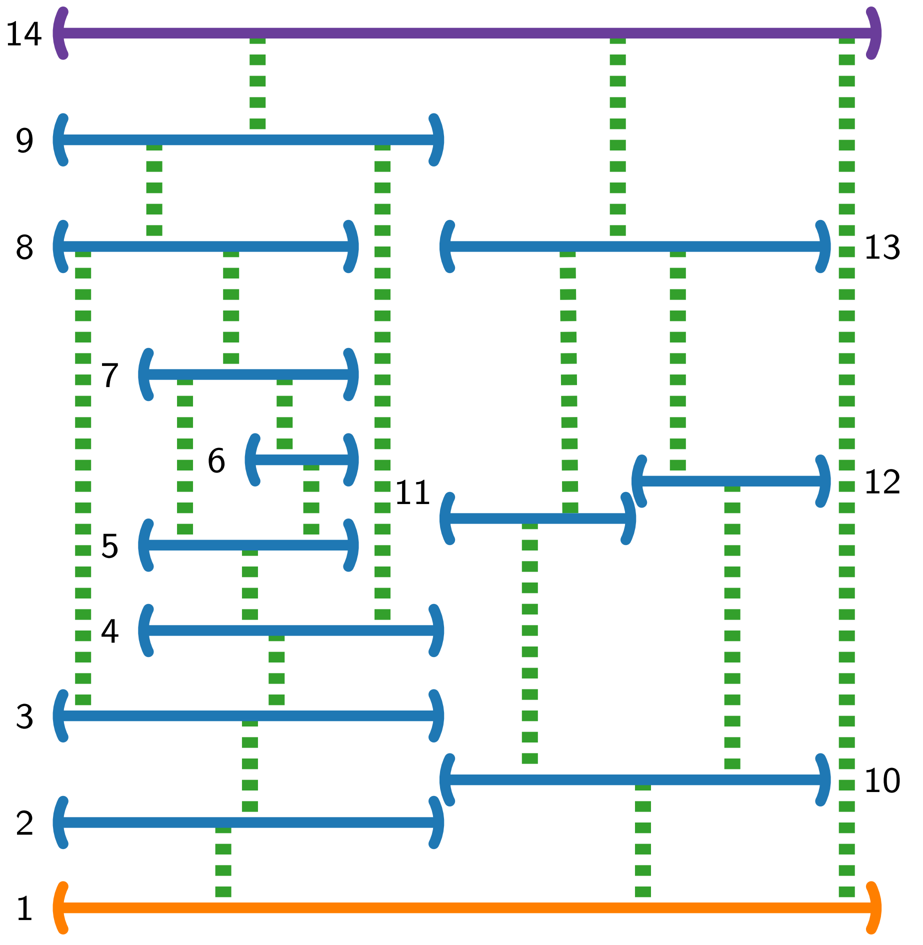


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.



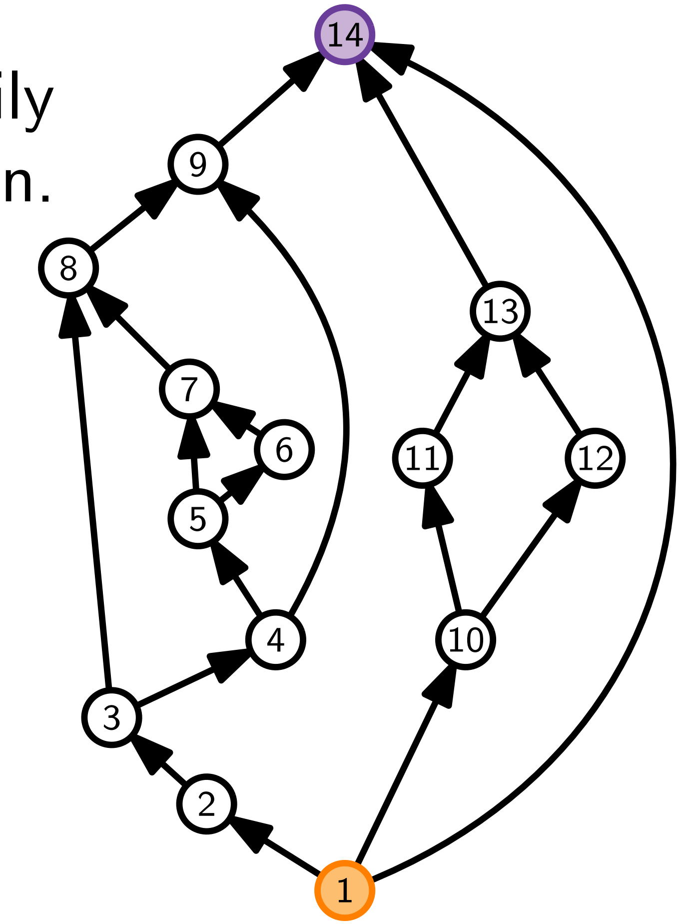


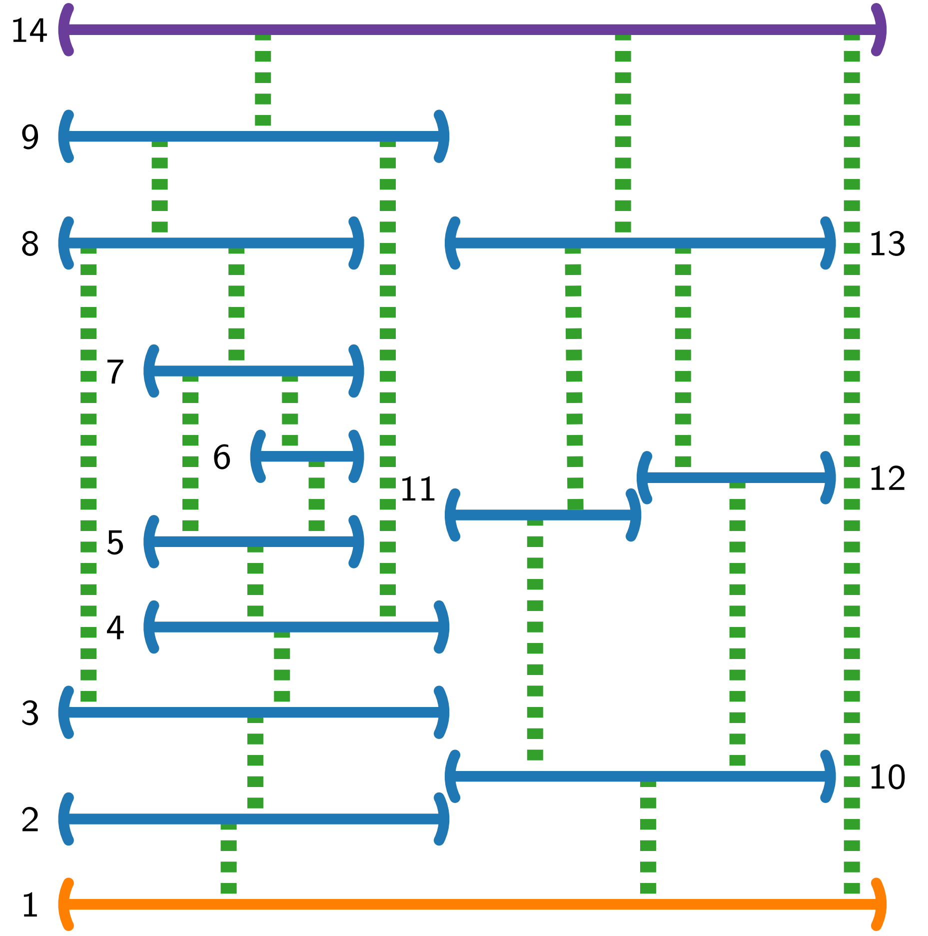
ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

- ε-bar visibility testing is easily done via st-graph recognition.



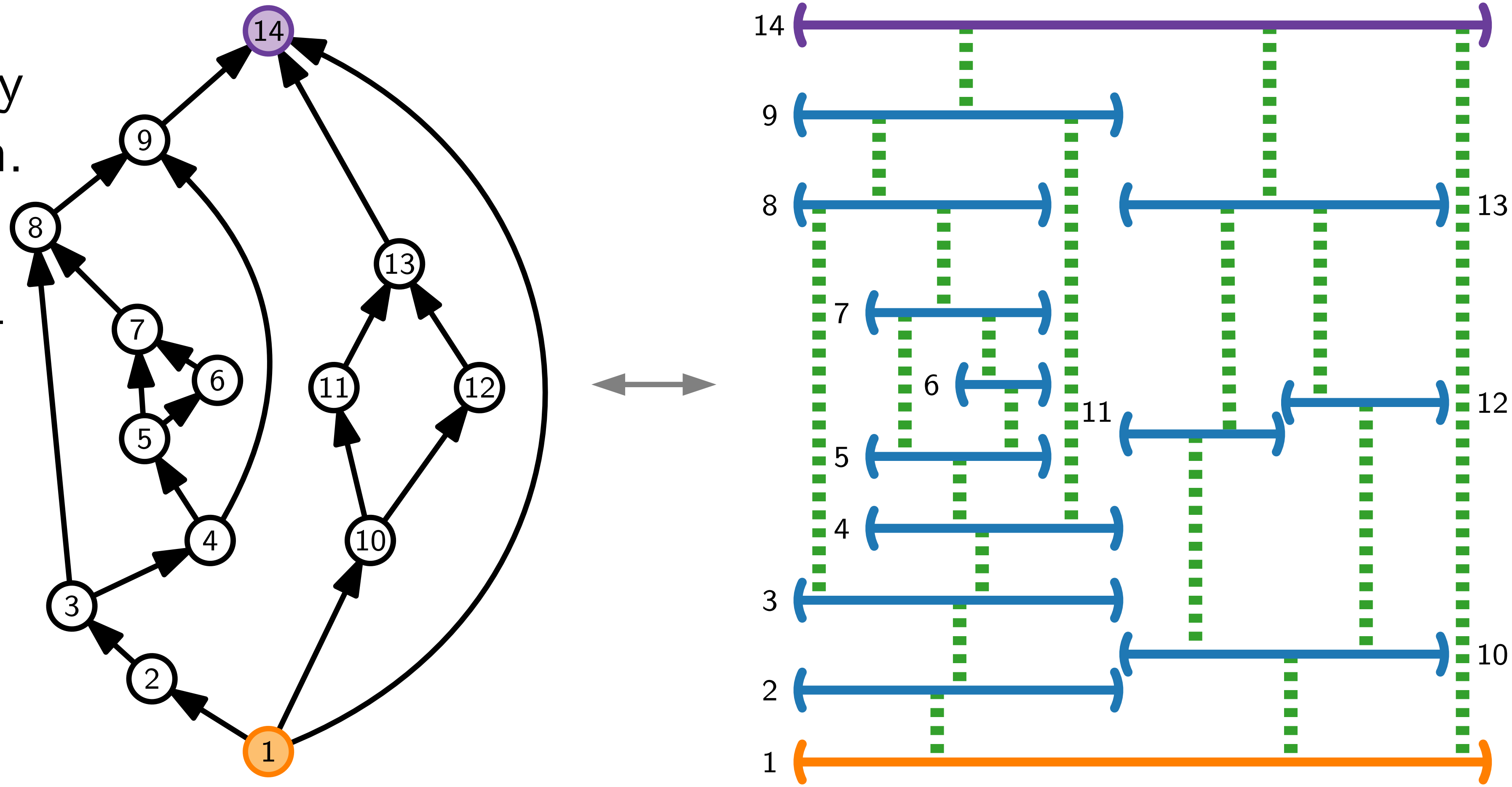


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

- ε-bar visibility testing is easily done via st-graph recognition.
- Strong bar visibility recognition. . . open!

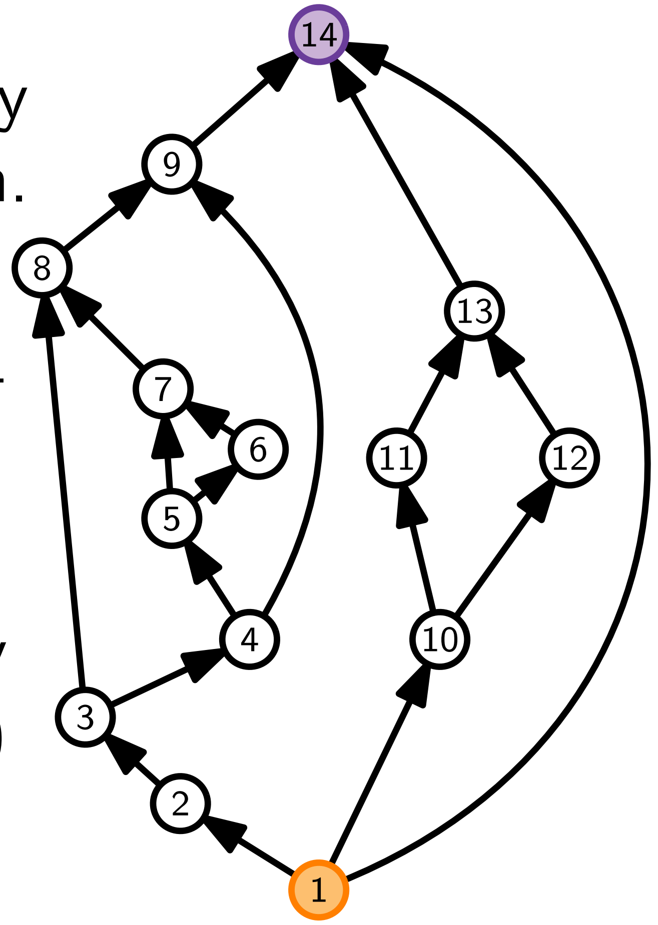


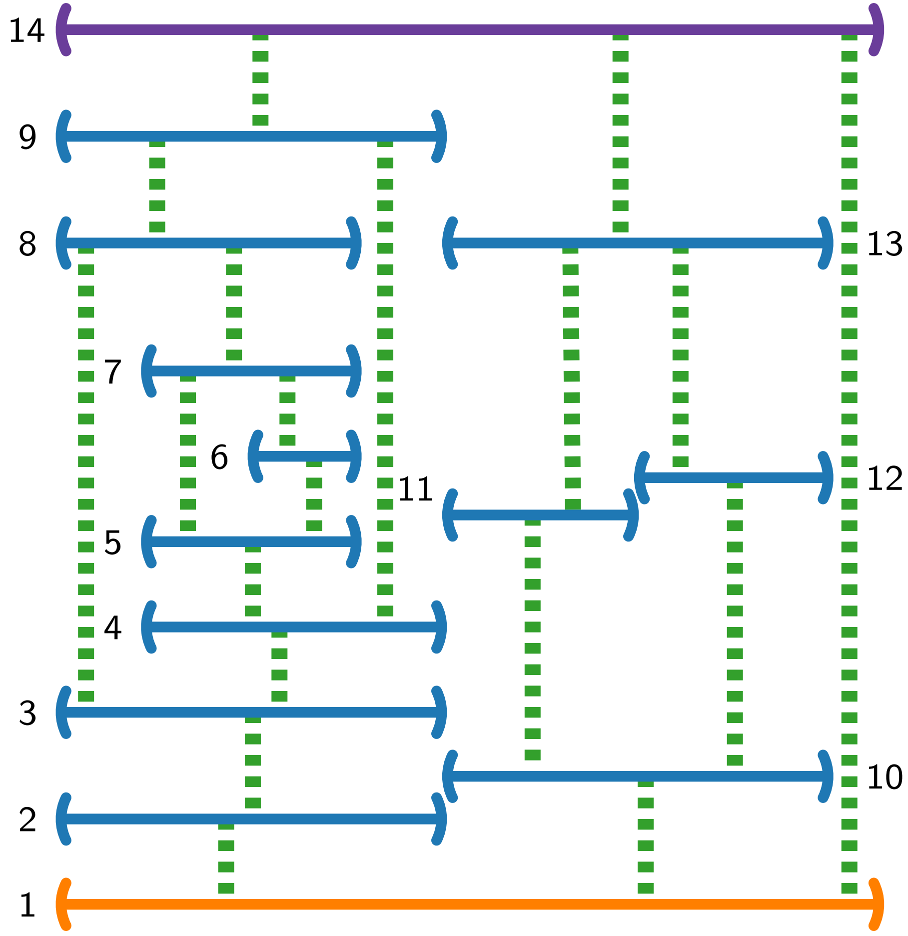
ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

- ε-bar visibility testing is easily done via st-graph recognition.
- Strong bar visibility recognition. . . open!
- In a **rectangular** bar visibility representation $\psi(s)$ and $\psi(t)$ span an enclosing rectangle.



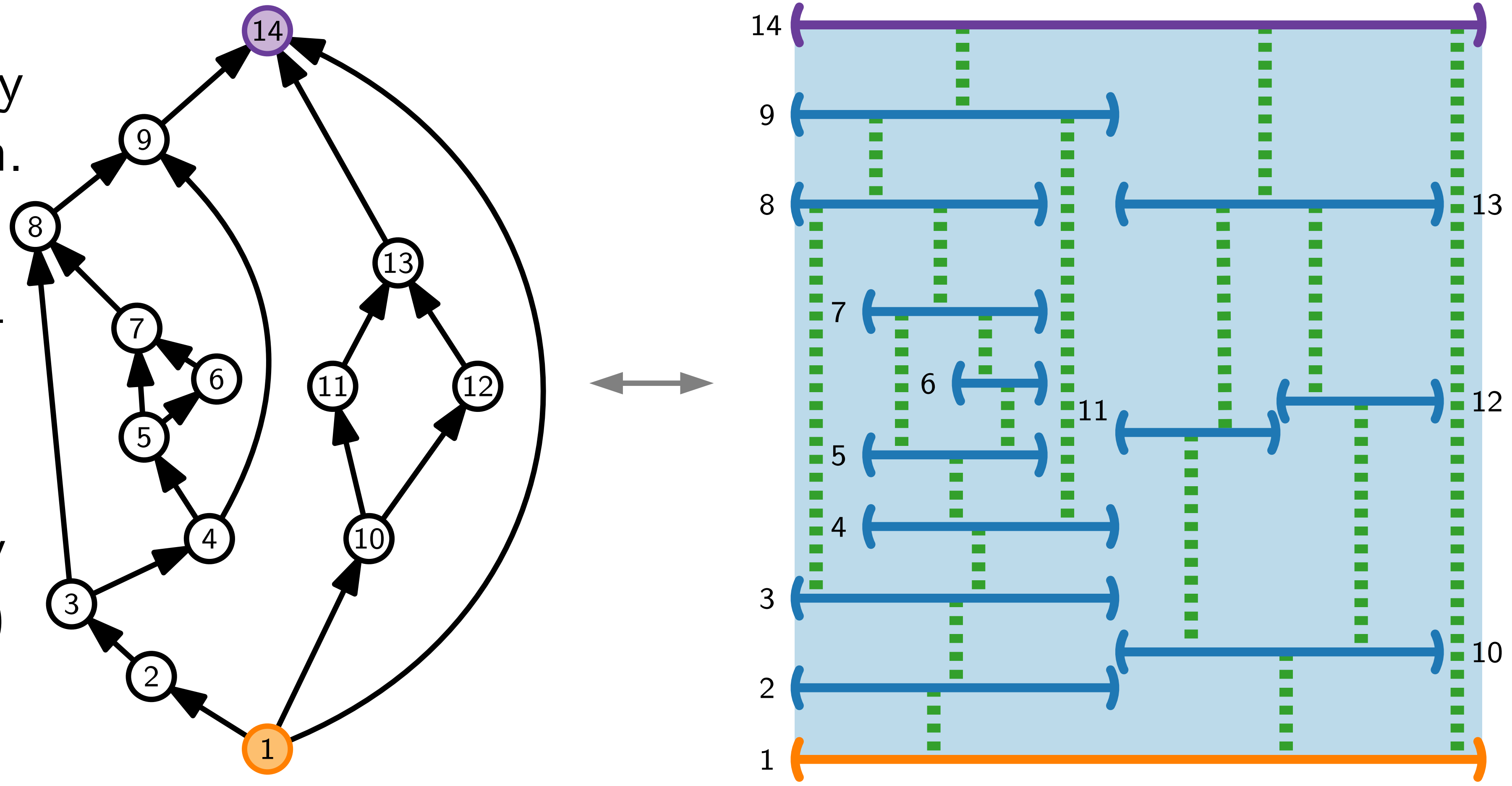


ε-Bar Visibility and st-Graphs

Recall that an **st-graph** is a planar acylic digraph G with exactly one source s and one sink t where s and t occur on the outer face of an embedding of G.

Observation.
st-orientations correspond to ε-bar visibility representations.

- ε-bar visibility testing is easily done via st-graph recognition.
- Strong bar visibility recognition. . . open!
- In a **rectangular** bar visibility representation $\psi(s)$ and $\psi(t)$ span an enclosing rectangle.



Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from Planar Monotone 3-SAT

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from Planar Monotone 3-SAT

Theorem 3.
ε-bar visibility representation extension is NP-complete even for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from $\textsc{Planar Monotone 3-SAT}$

Theorem 3.
ε-bar visibility representation extension is NP-complete even for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from $\textsc{3-Partition}$

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time for st-graphs.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from Planar Monotone 3-SAT

Theorem 3.
ε-bar visibility representation extension is NP-complete even for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

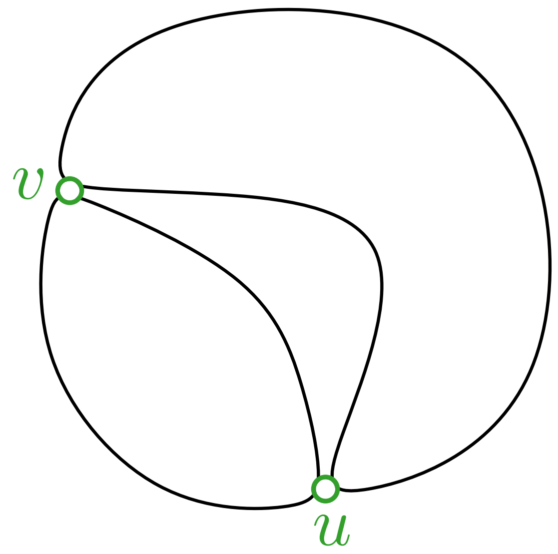
- Reduction from 3-Partition

SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.

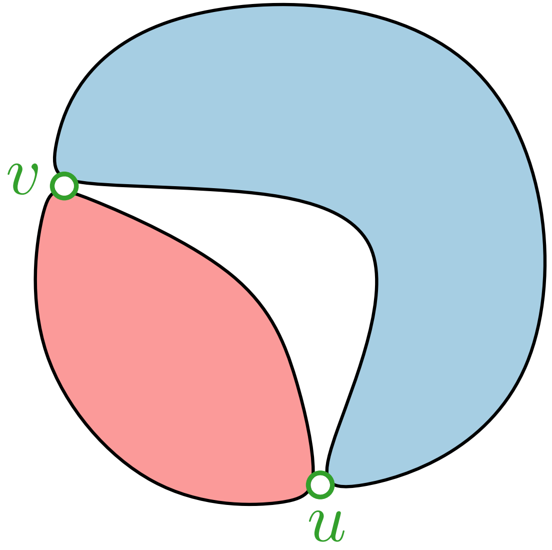
SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.



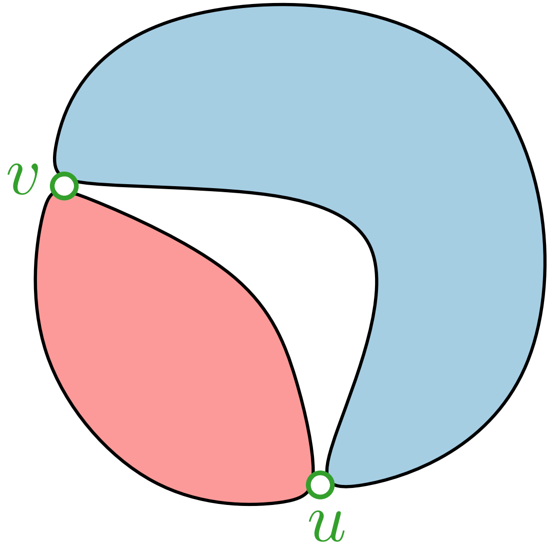
SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.



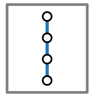
SPQR-Tree

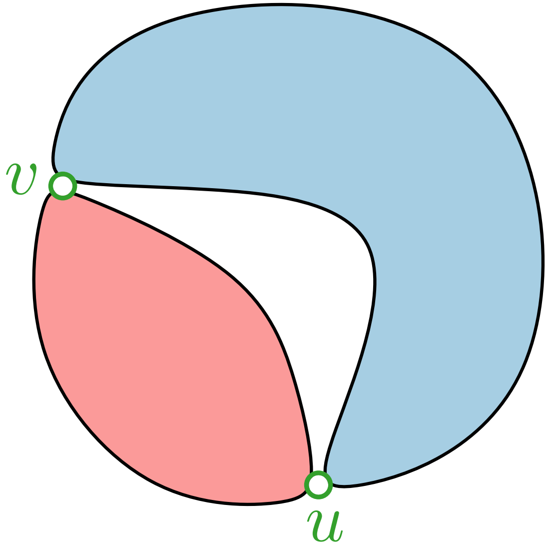
- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:



SPQR-Tree

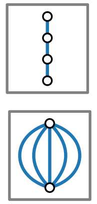
- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition

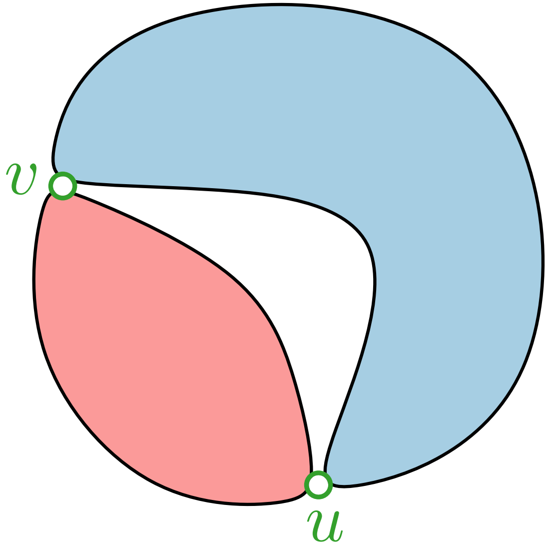




SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition

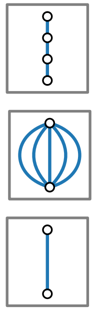


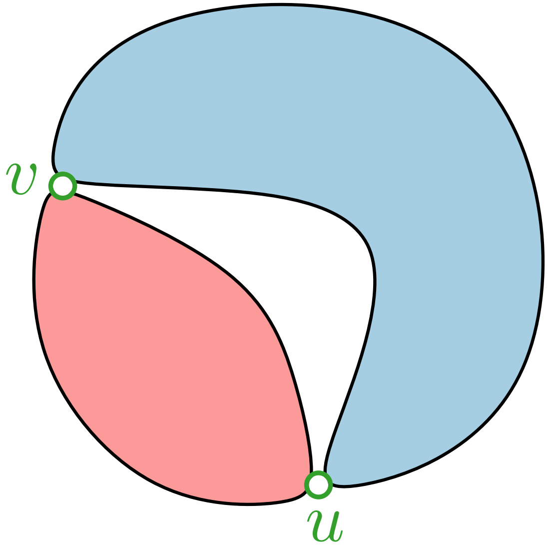


SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.

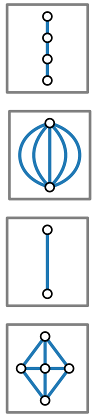
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition
  - **Q**-nodes represent a single edge

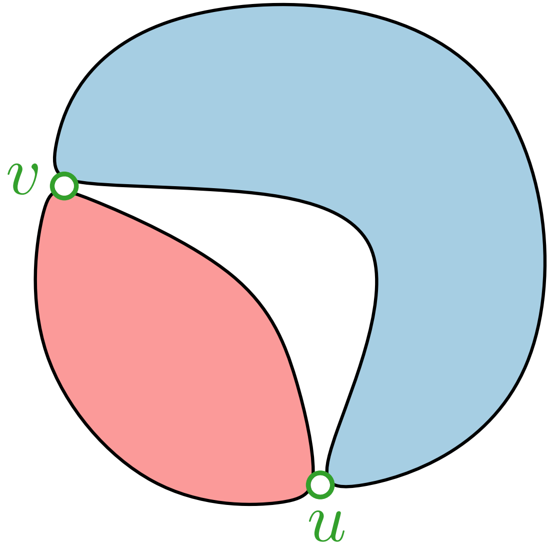




SPQR-Tree

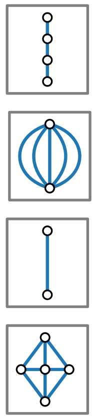
- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition
  - **Q**-nodes represent a single edge
  - **R**-nodes represent 3-connected (*rigid*) subgraphs

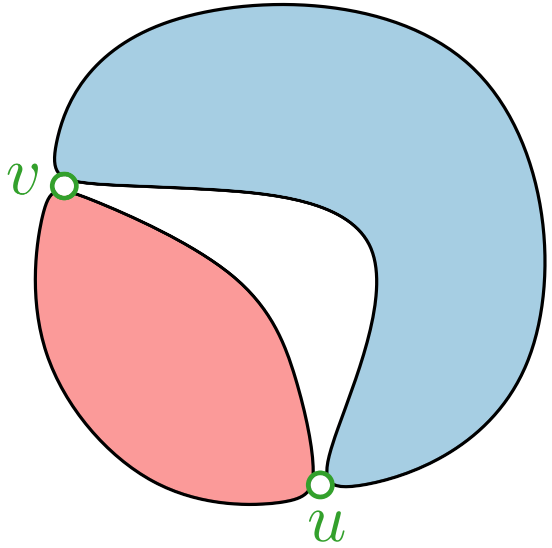




SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition
  - **Q**-nodes represent a single edge
  - **R**-nodes represent 3-connected (*rigid*) subgraphs

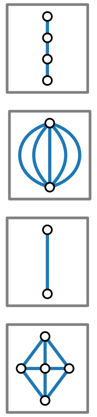


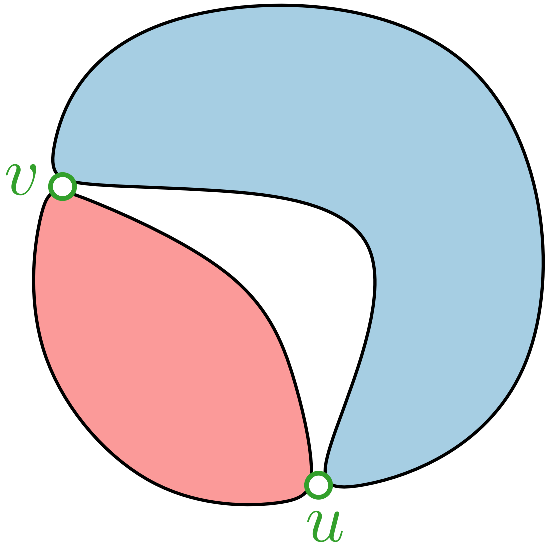


- A decomposition tree of a series-parallel graph is an SPQR-tree without **R**-nodes.

SPQR-Tree

- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition
  - **Q**-nodes represent a single edge
  - **R**-nodes represent 3-connected (*rigid*) subgraphs

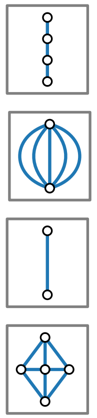


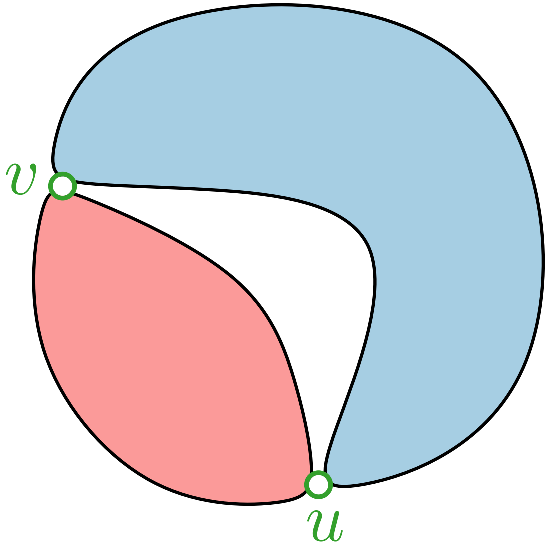


- A decomposition tree of a series-parallel graph is an SPQR-tree without **R**-nodes.
- T represents all planar embeddings of G.

SPQR-Tree

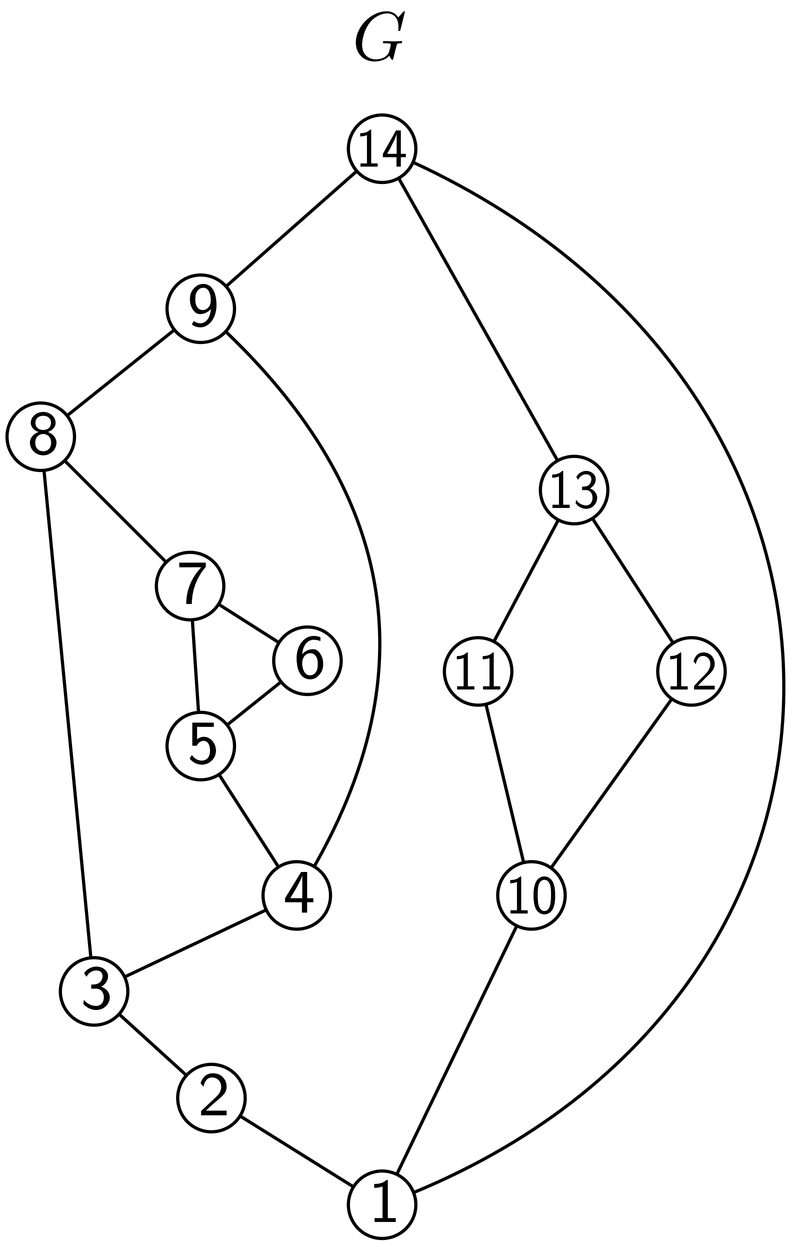
- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S**-nodes represent a series composition
  - **P**-nodes represent a parallel composition
  - **Q**-nodes represent a single edge
  - **R**-nodes represent 3-connected (*rigid*) subgraphs



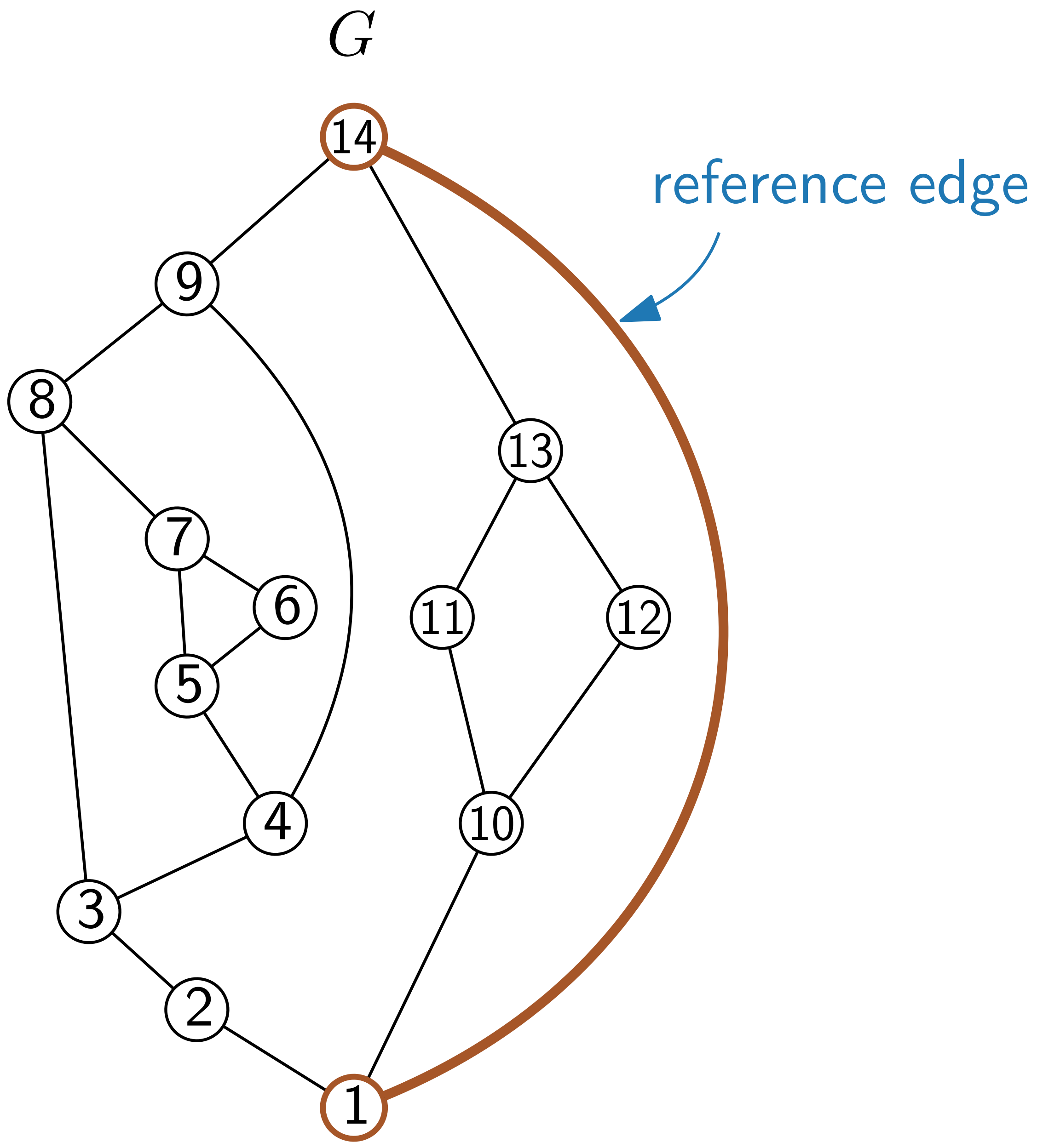


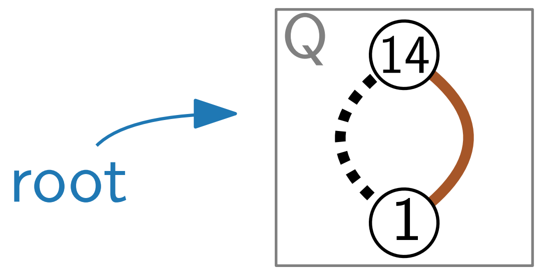
- A decomposition tree of a series-parallel graph is an SPQR-tree without **R**-nodes.
- T represents all planar embeddings of G.
- T can be computed in time linear in the size of G. [Gutwenger, Mutzel '01]

SPQR-Tree – Example

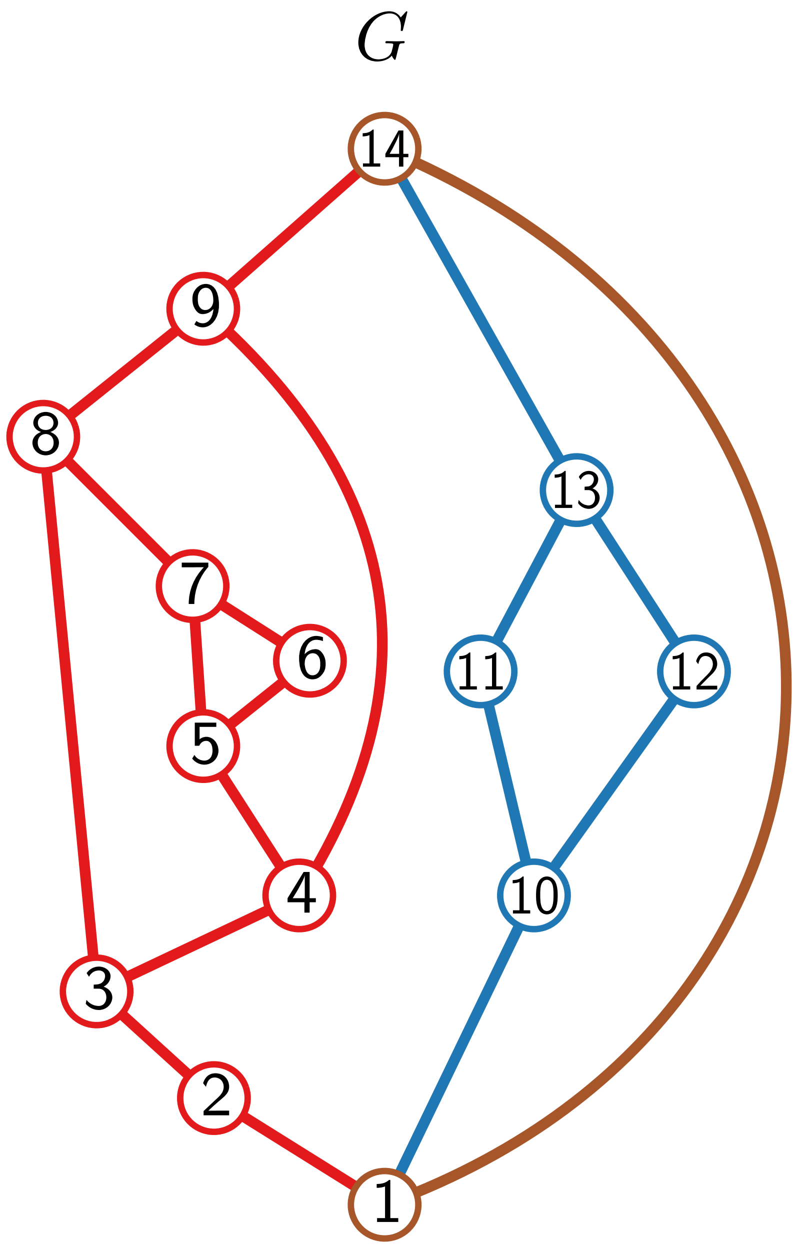


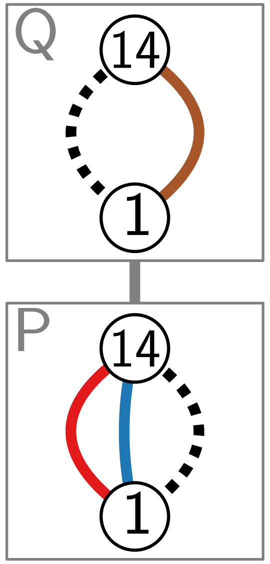
SPQR-Tree – Example



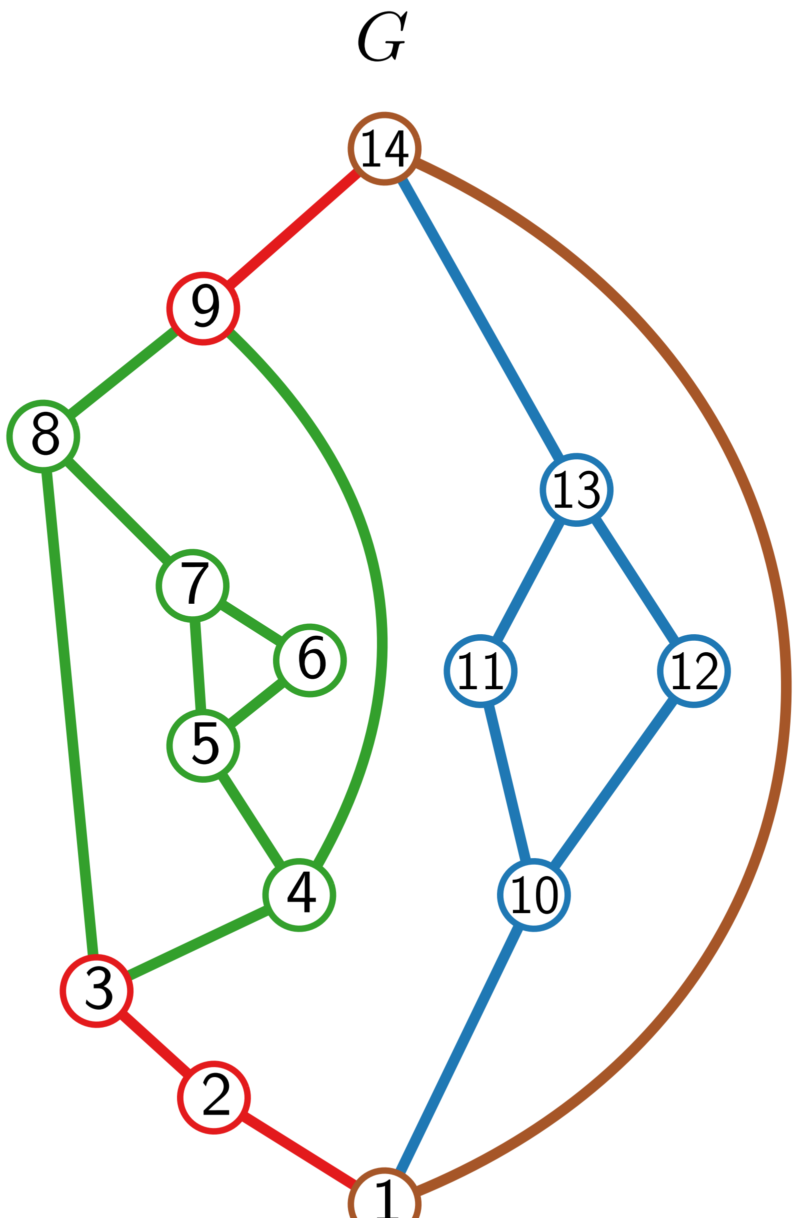


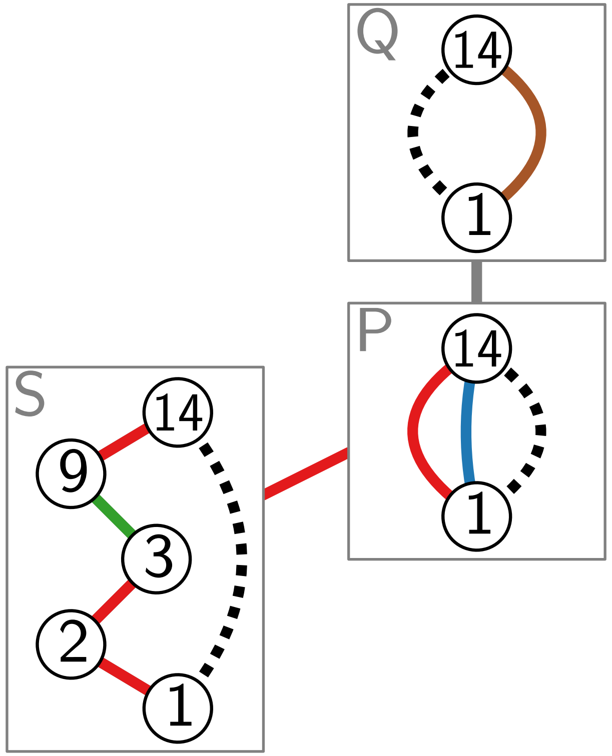
SPQR-Tree – Example



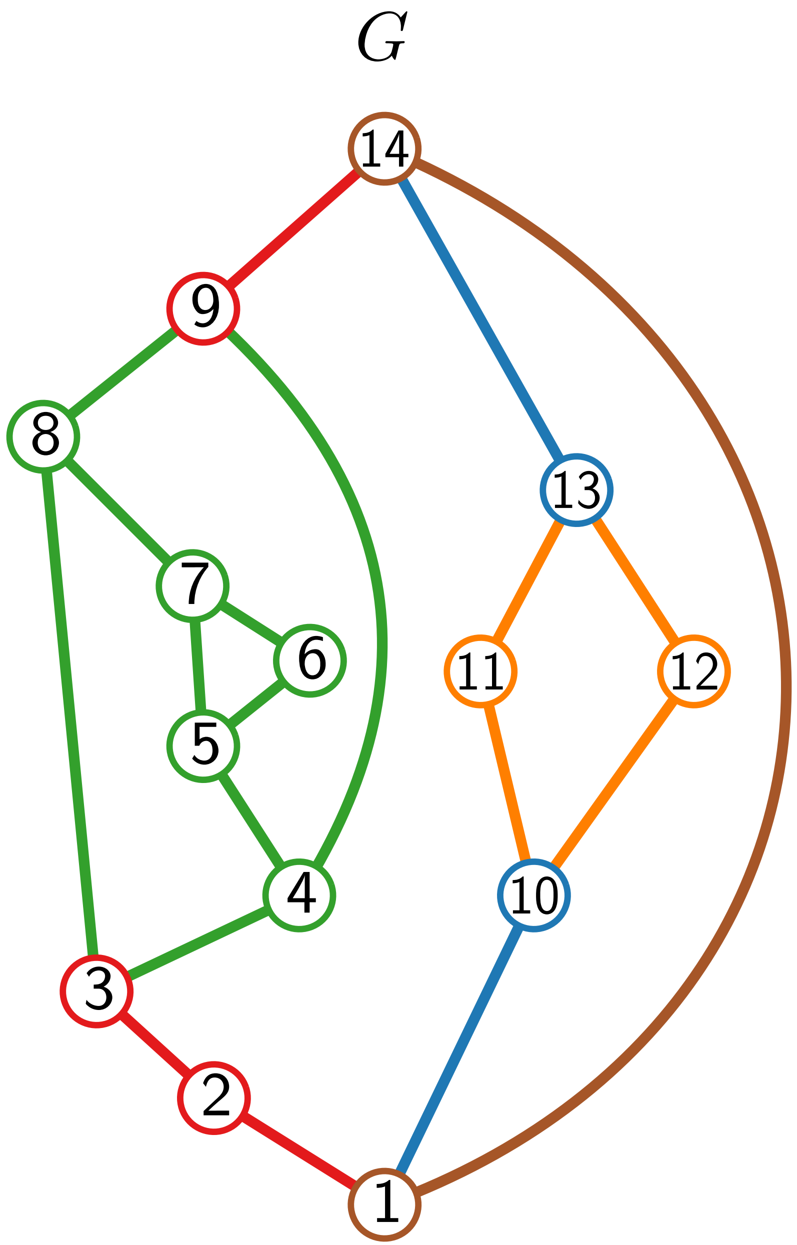


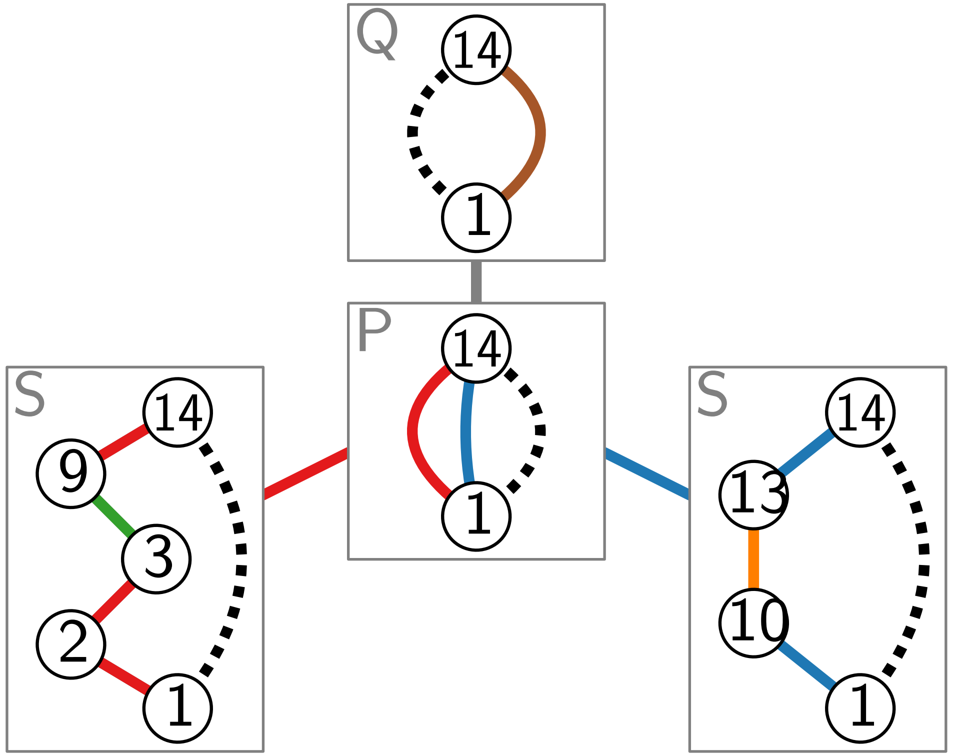
SPQR-Tree – Example



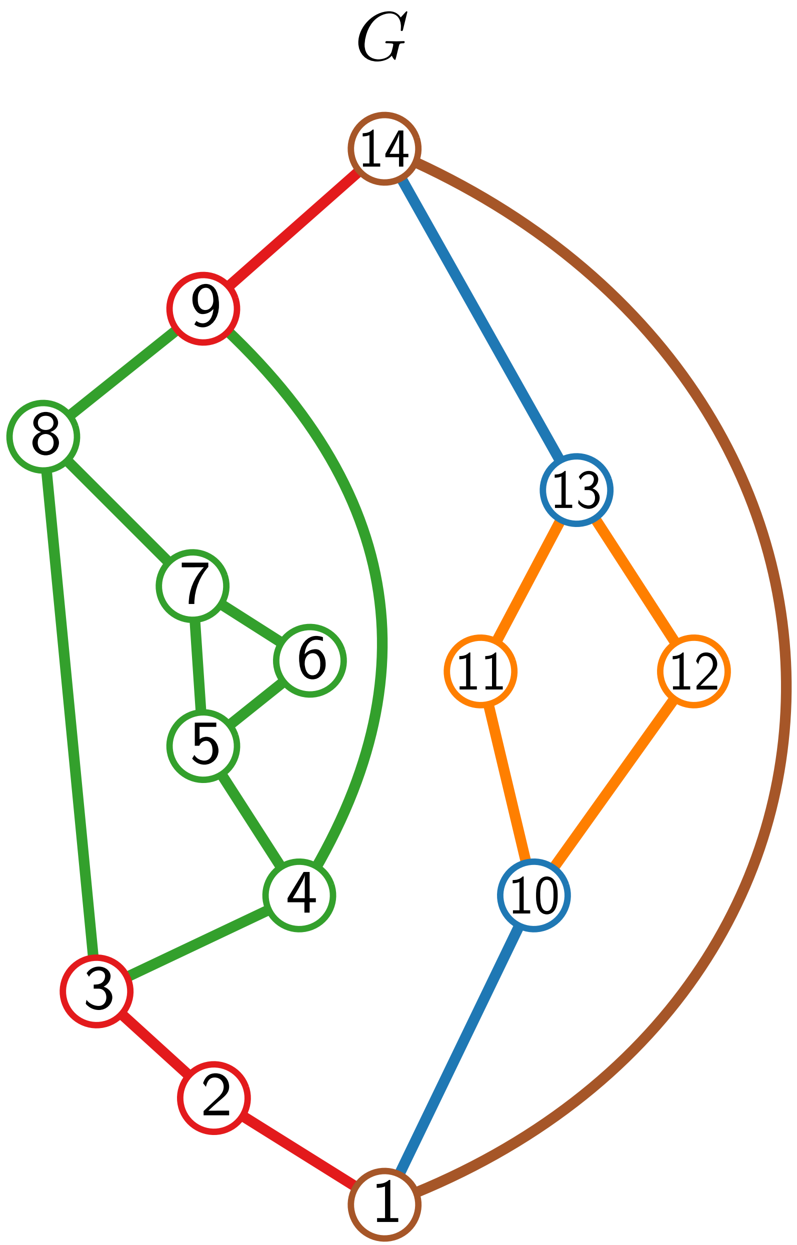


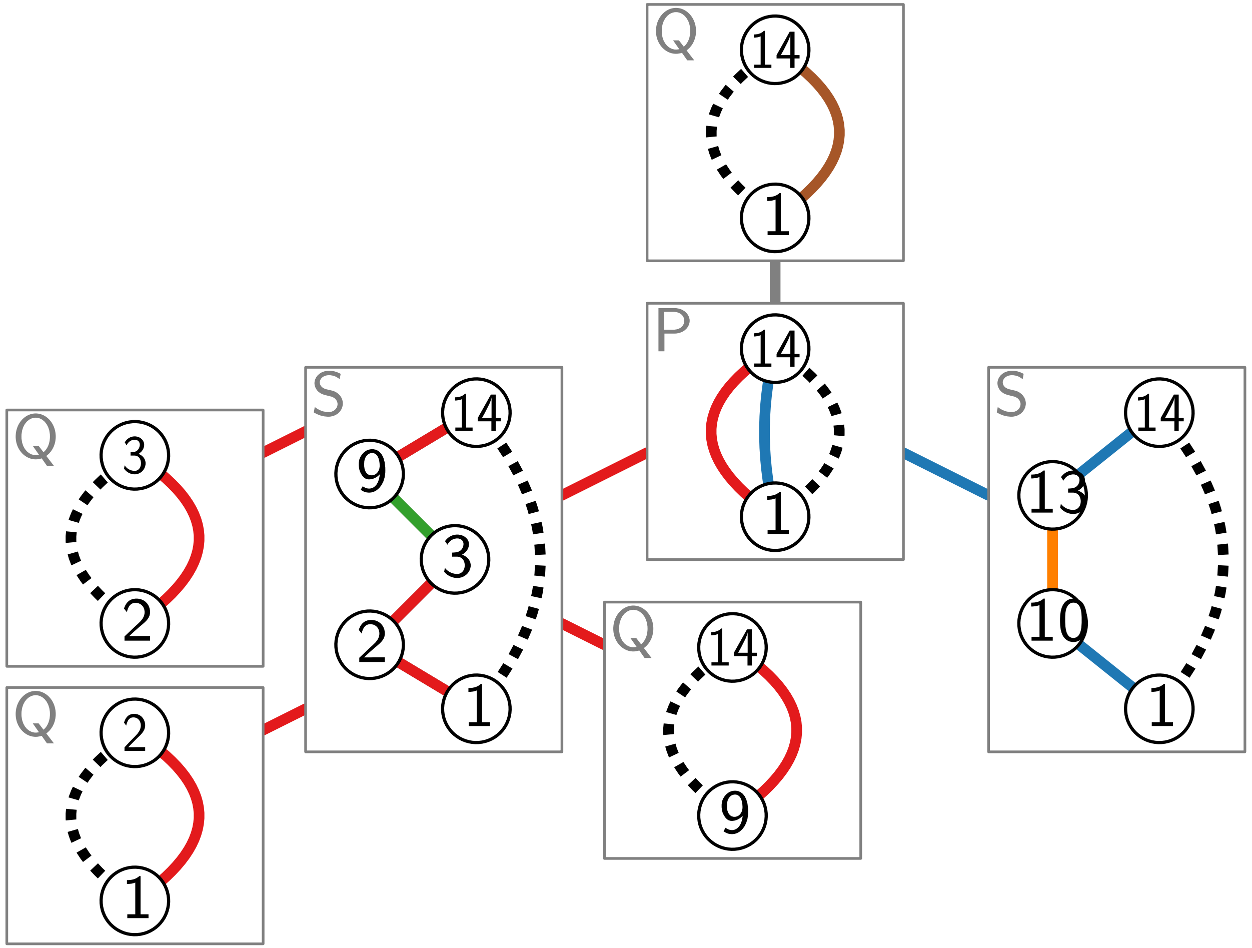
SPQR-Tree – Example



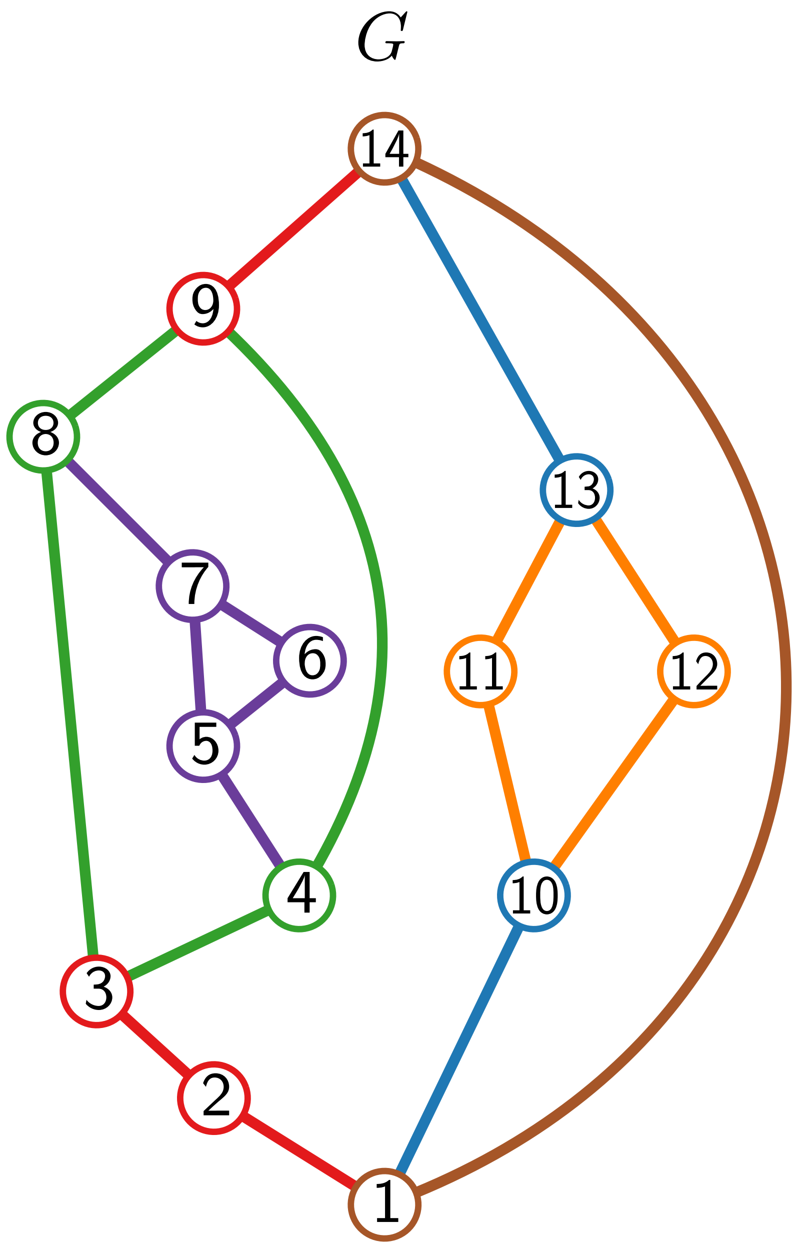


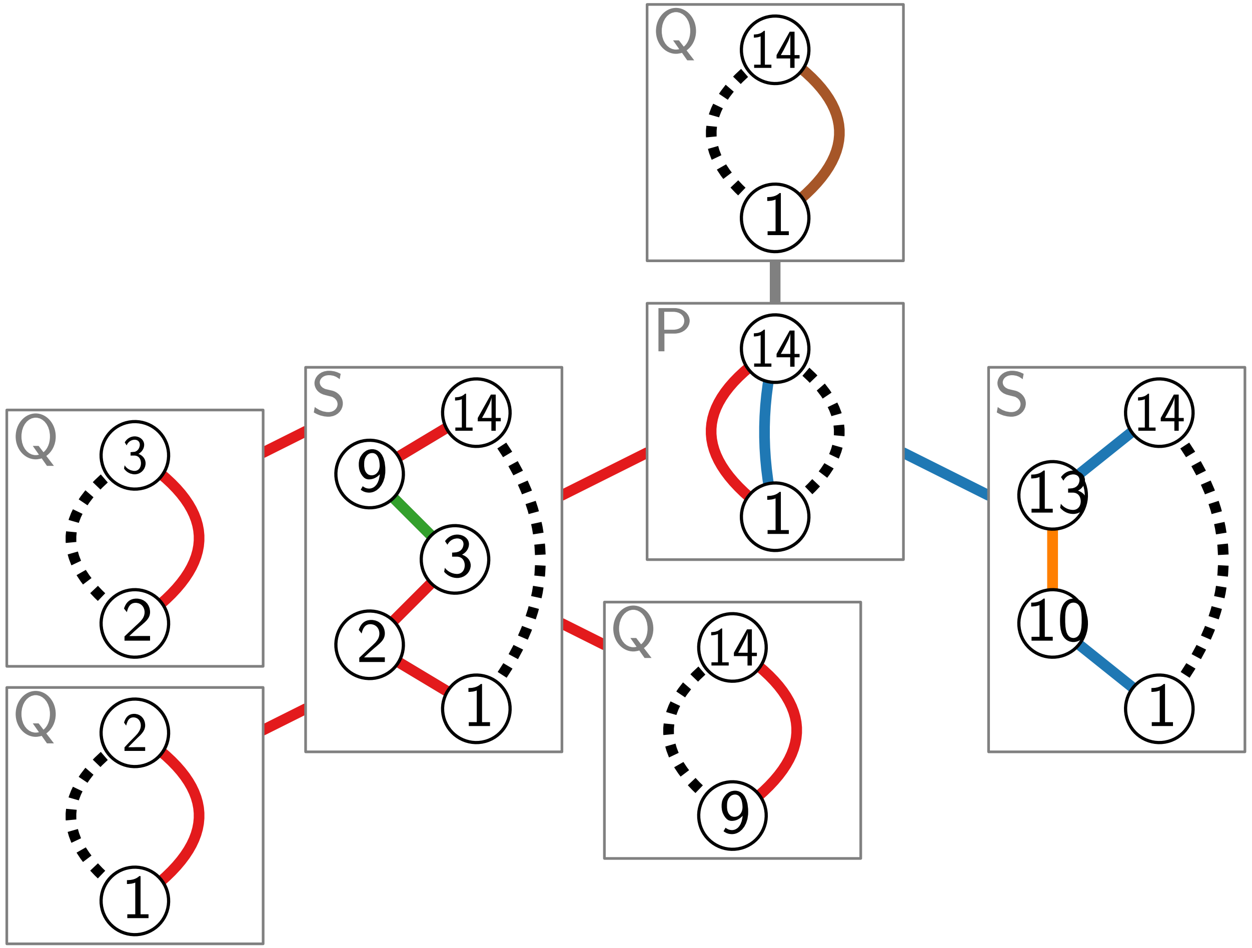
SPQR-Tree – Example



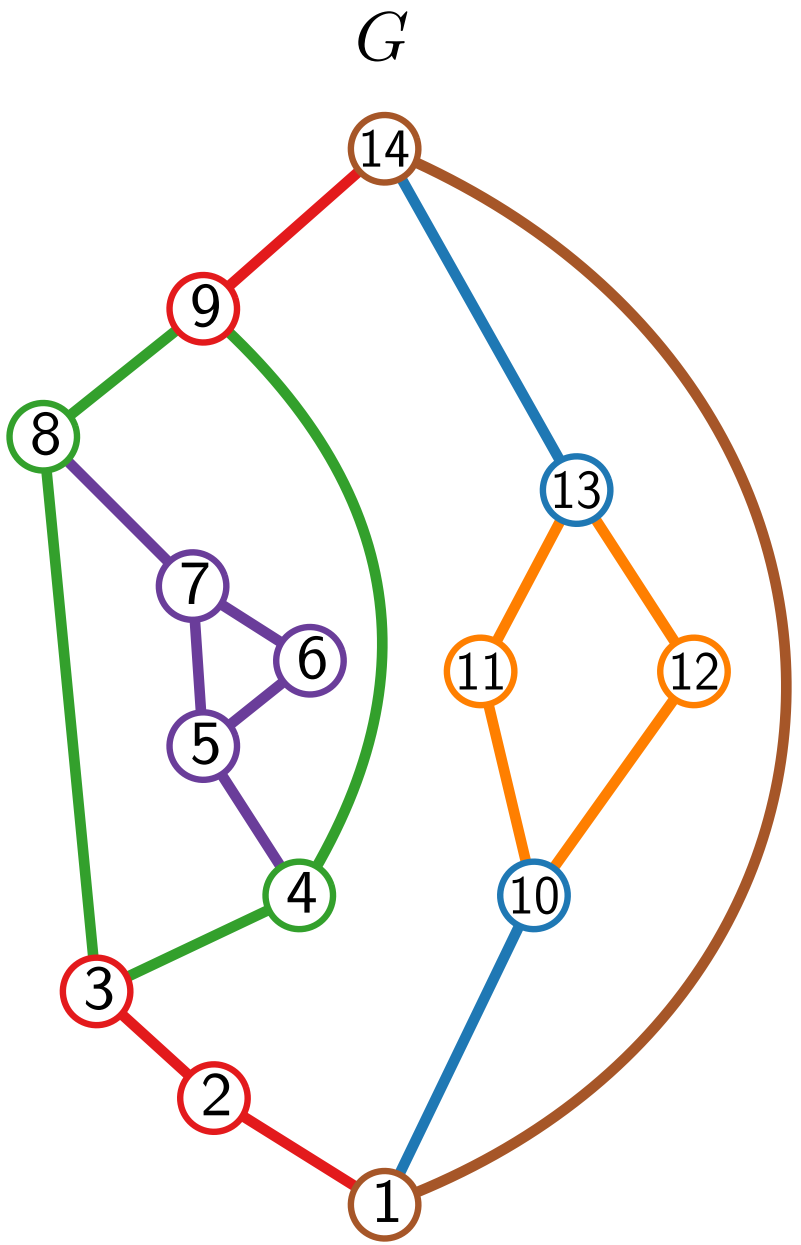


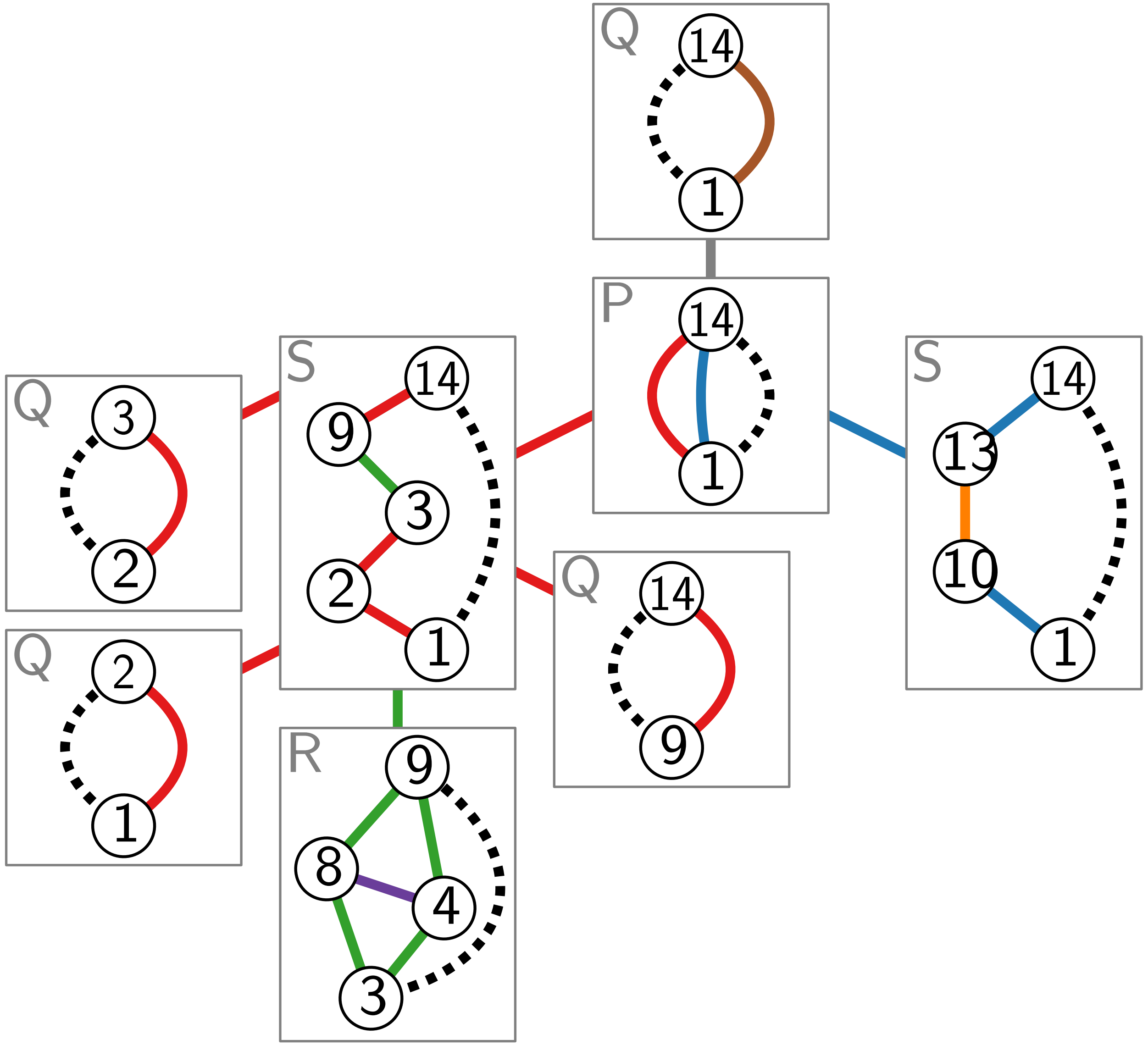
SPQR-Tree – Example



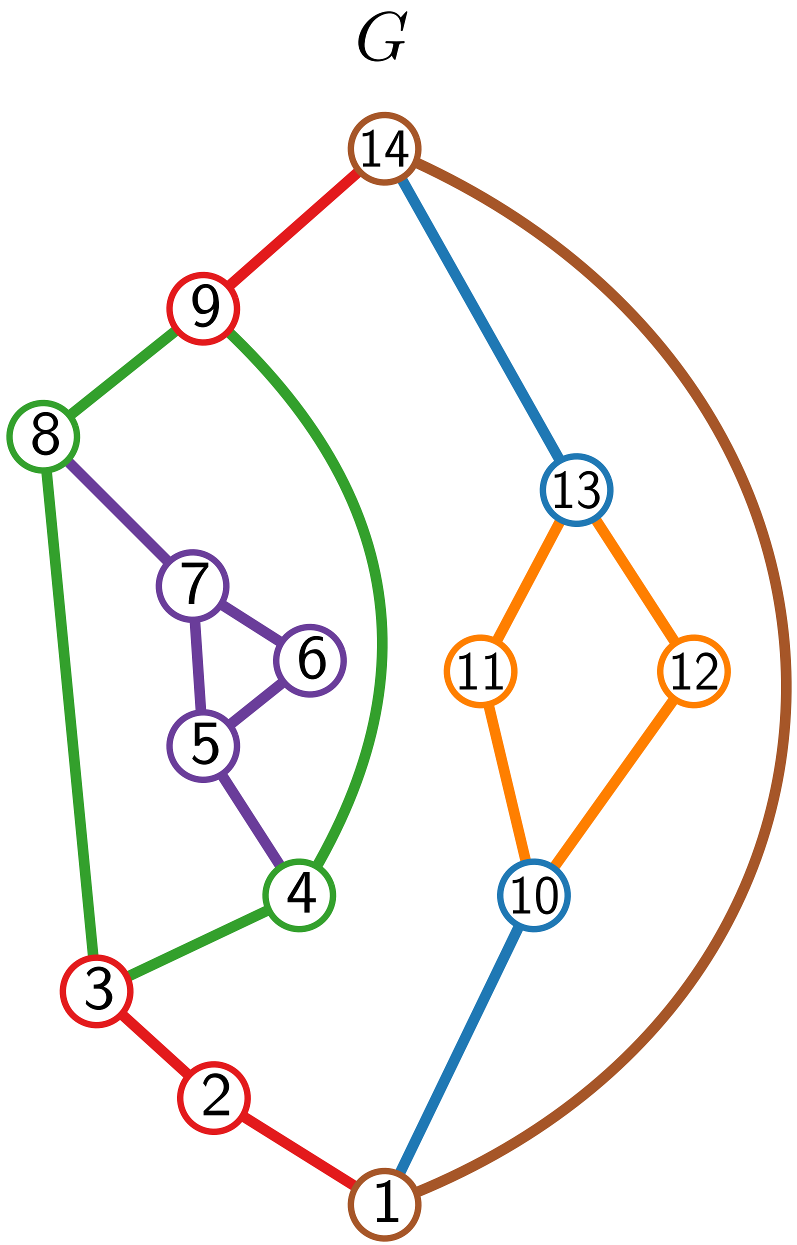


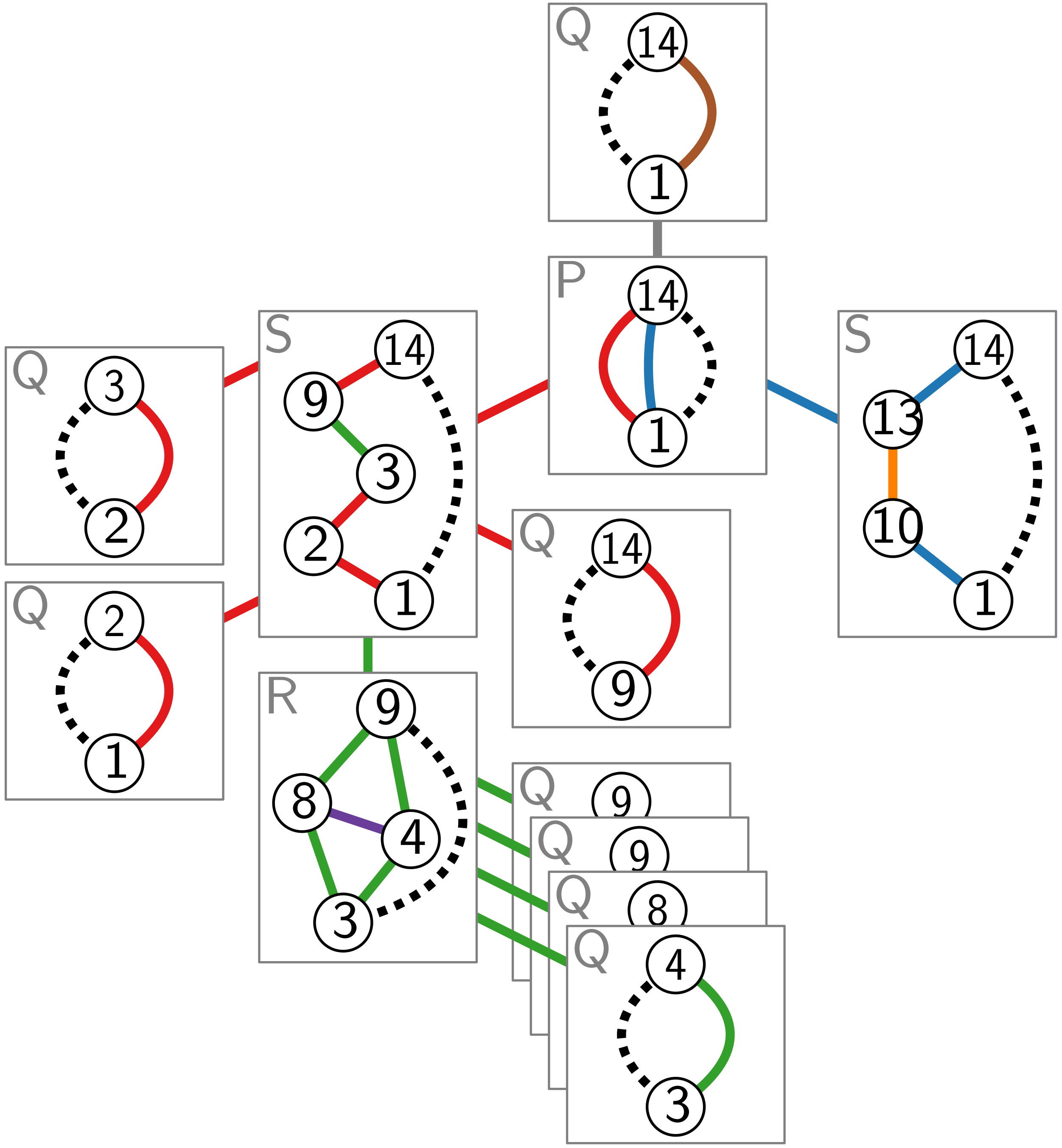
SPQR-Tree – Example





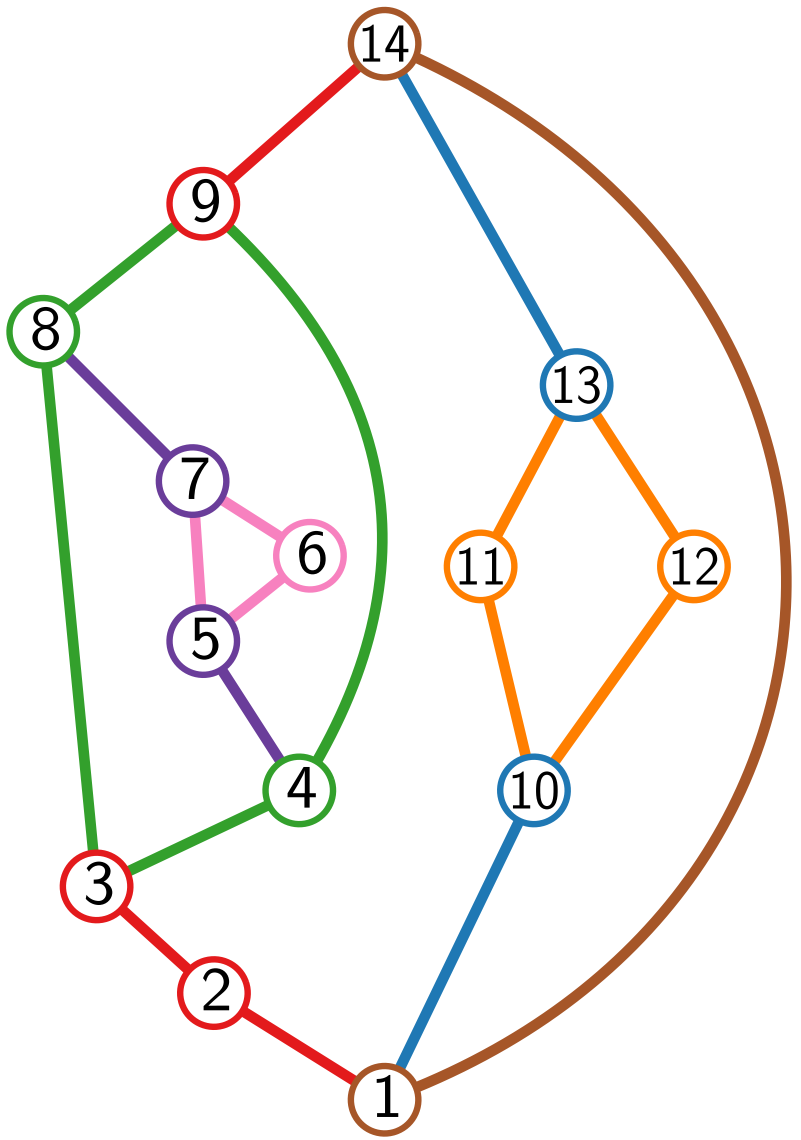
SPQR-Tree – Example

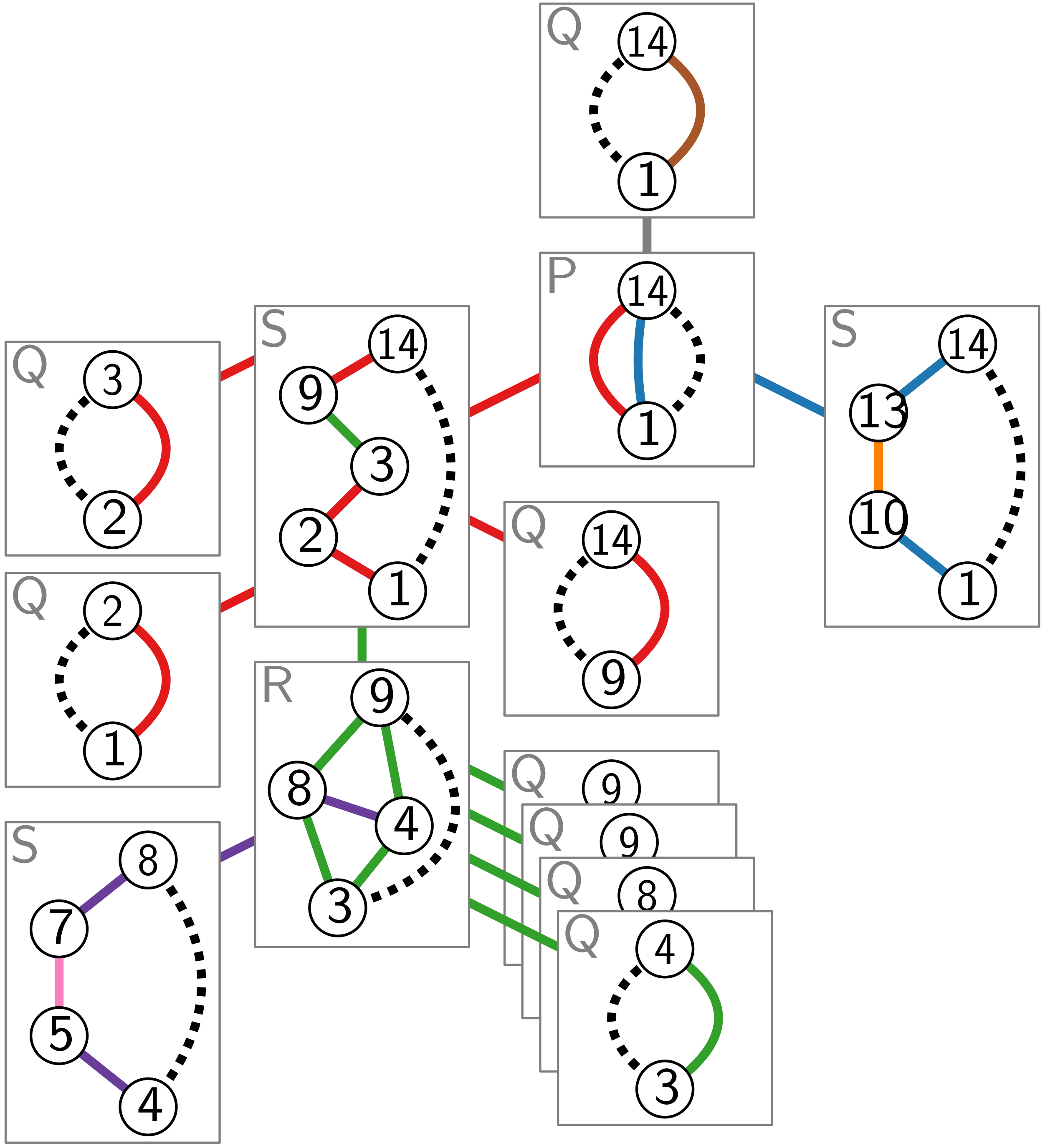




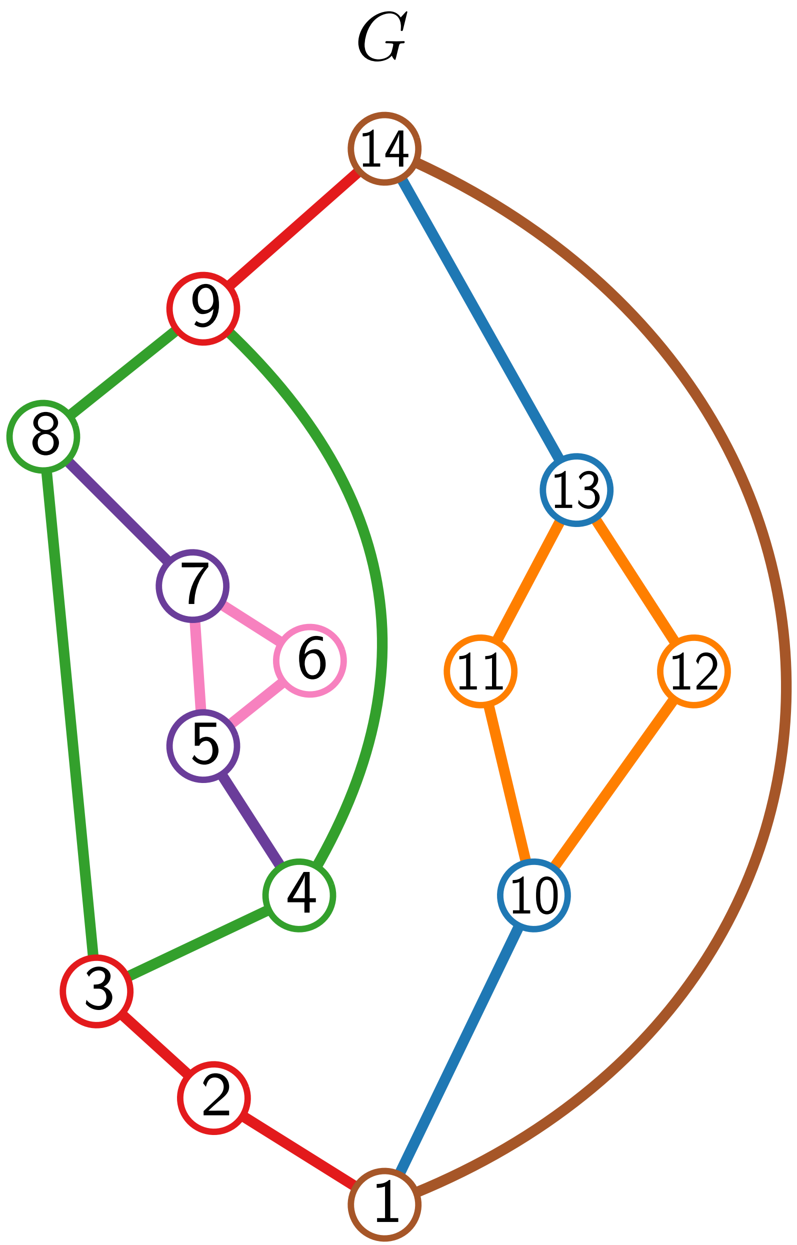
SPQR-Tree – Example

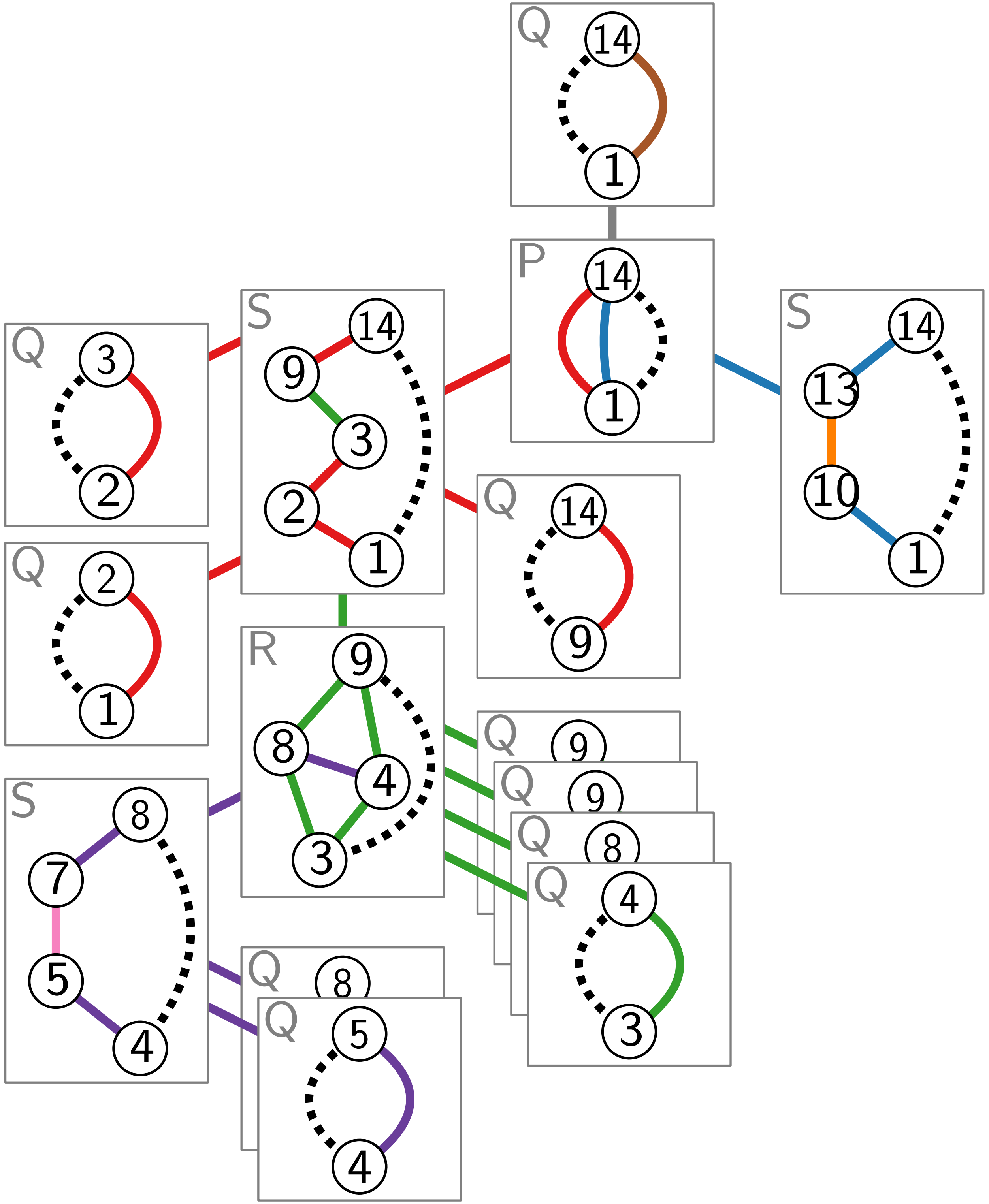
G



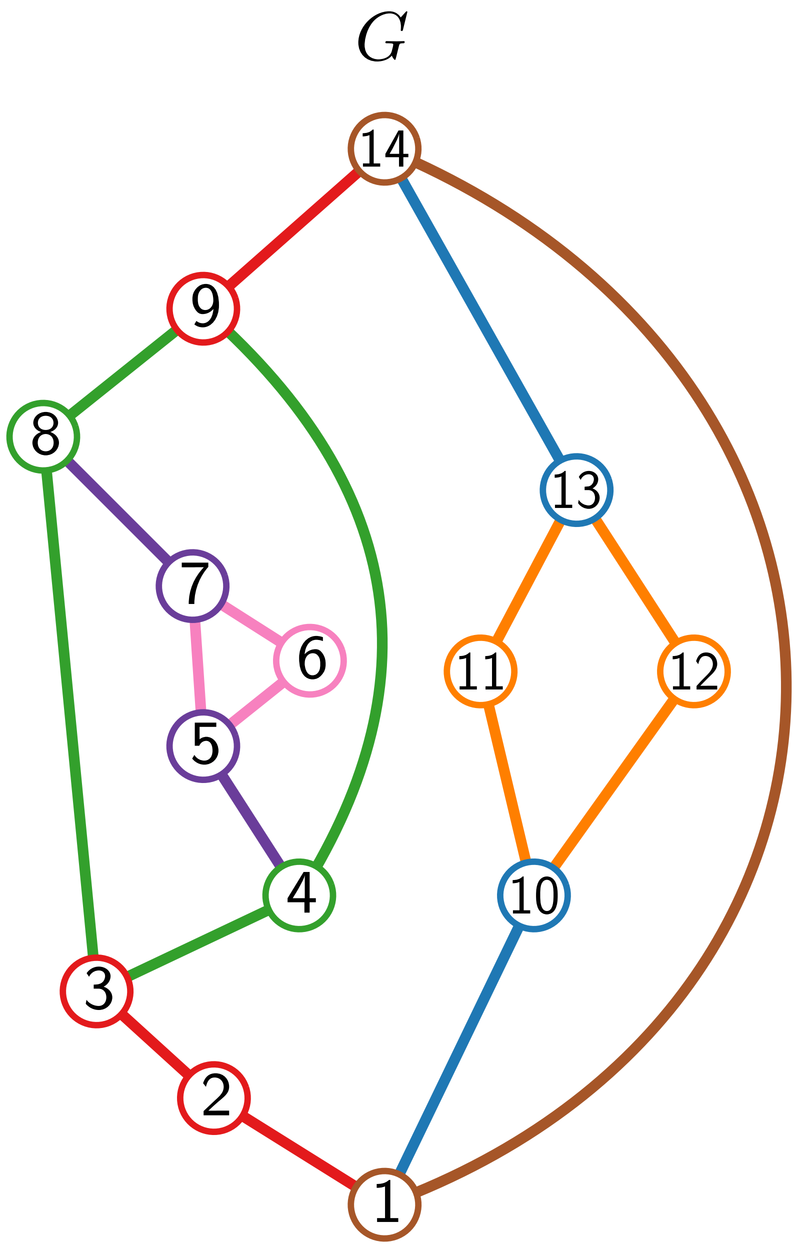


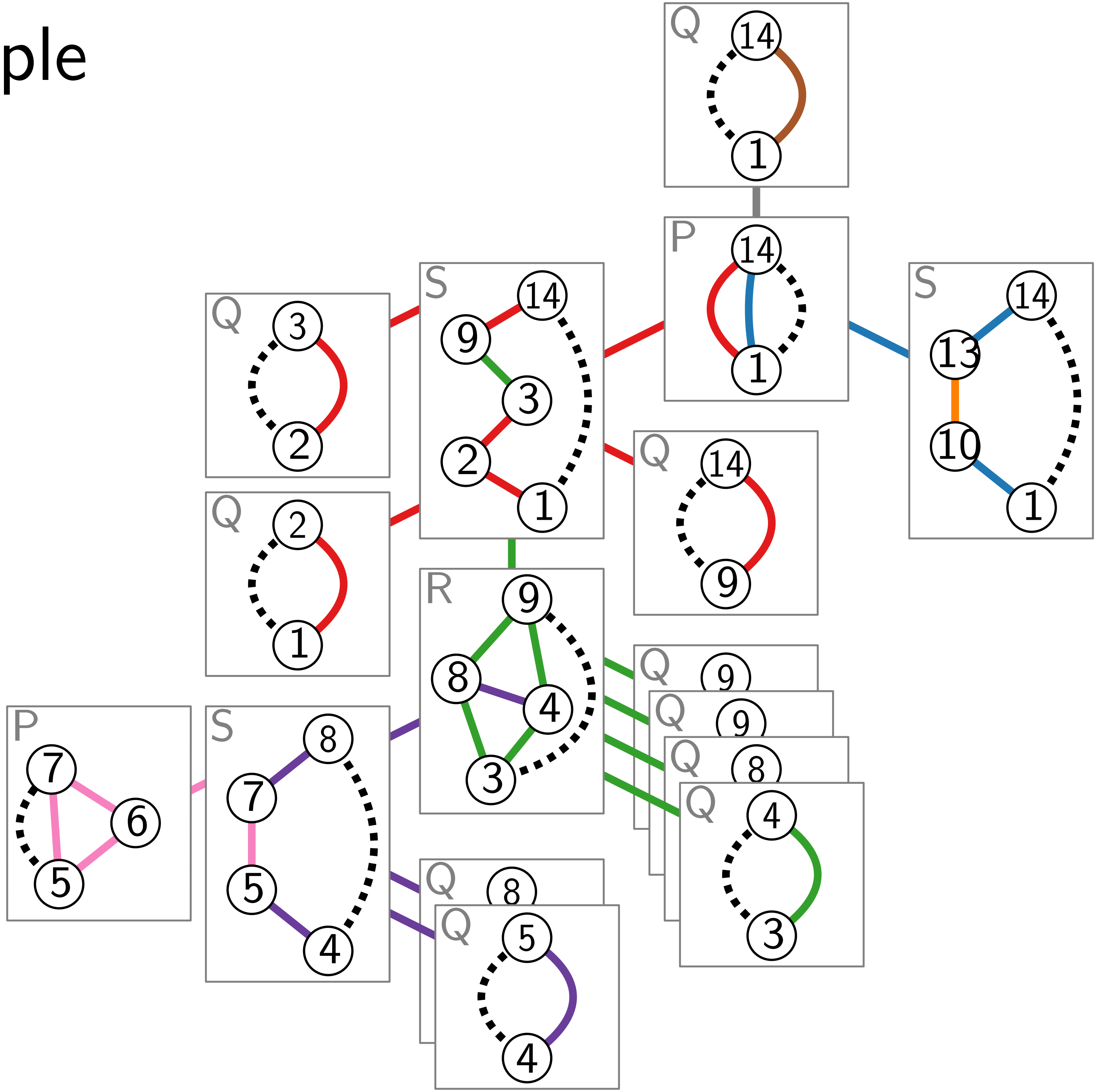
SPQR-Tree – Example



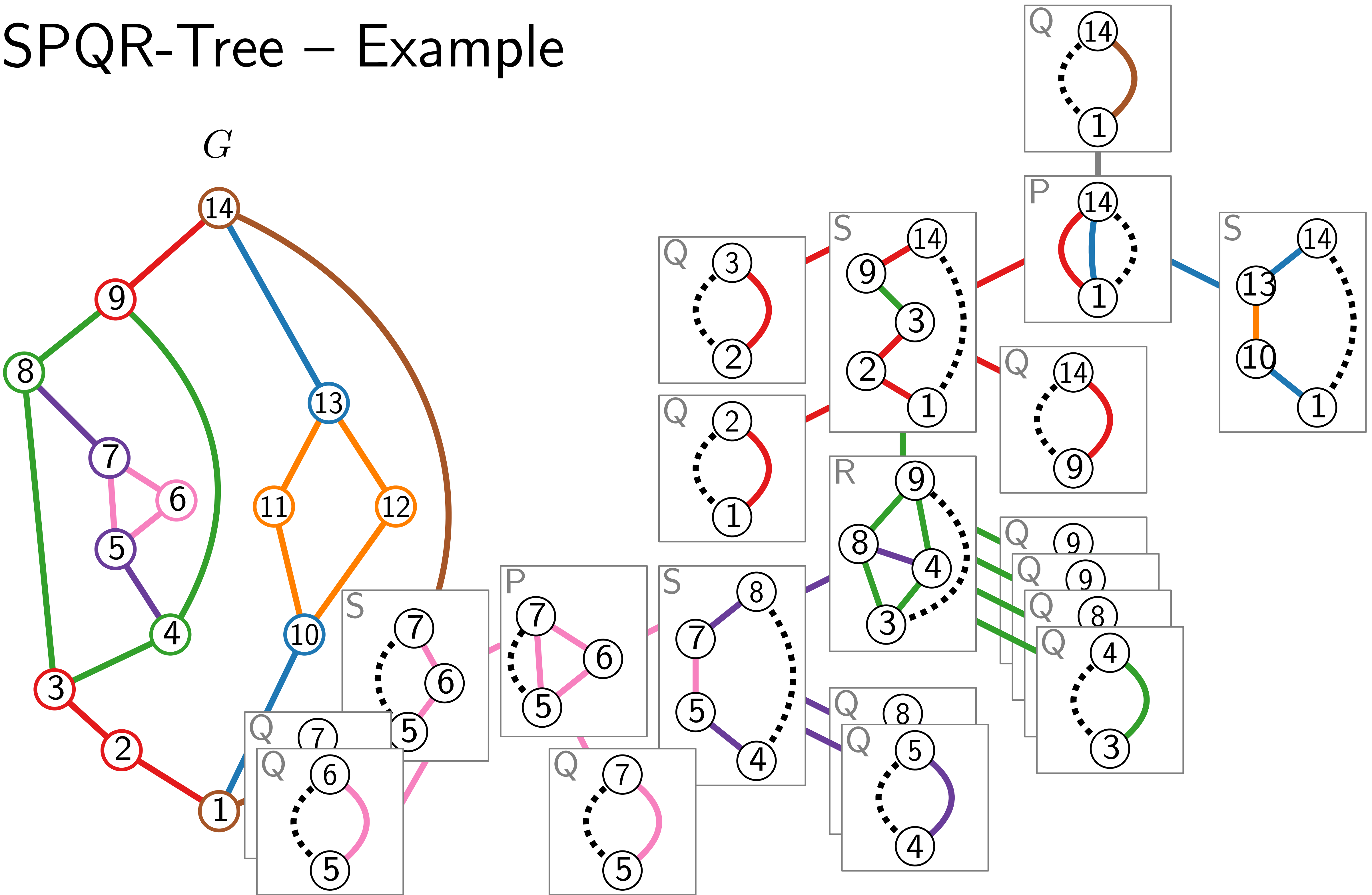


SPQR-Tree – Example

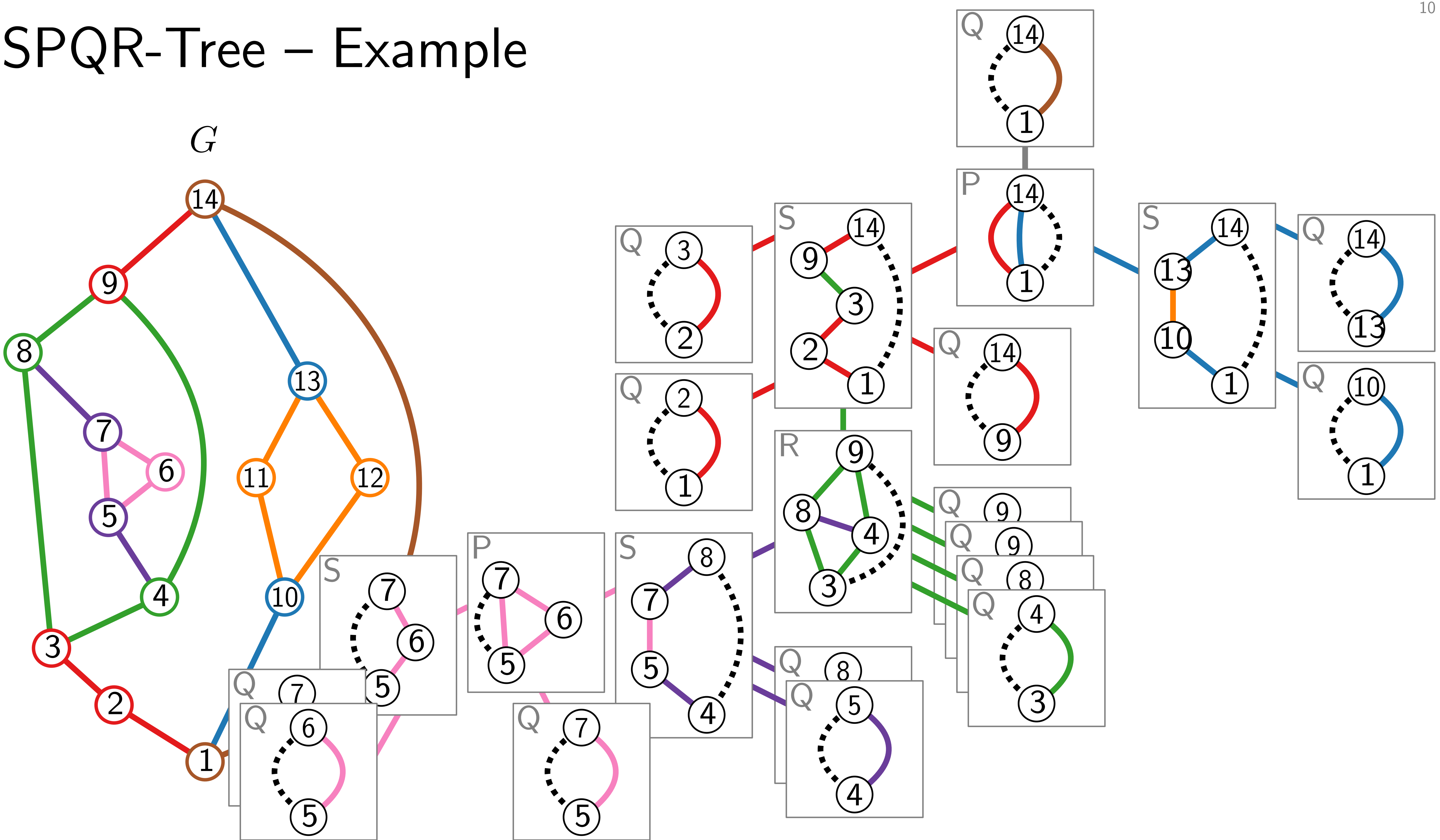




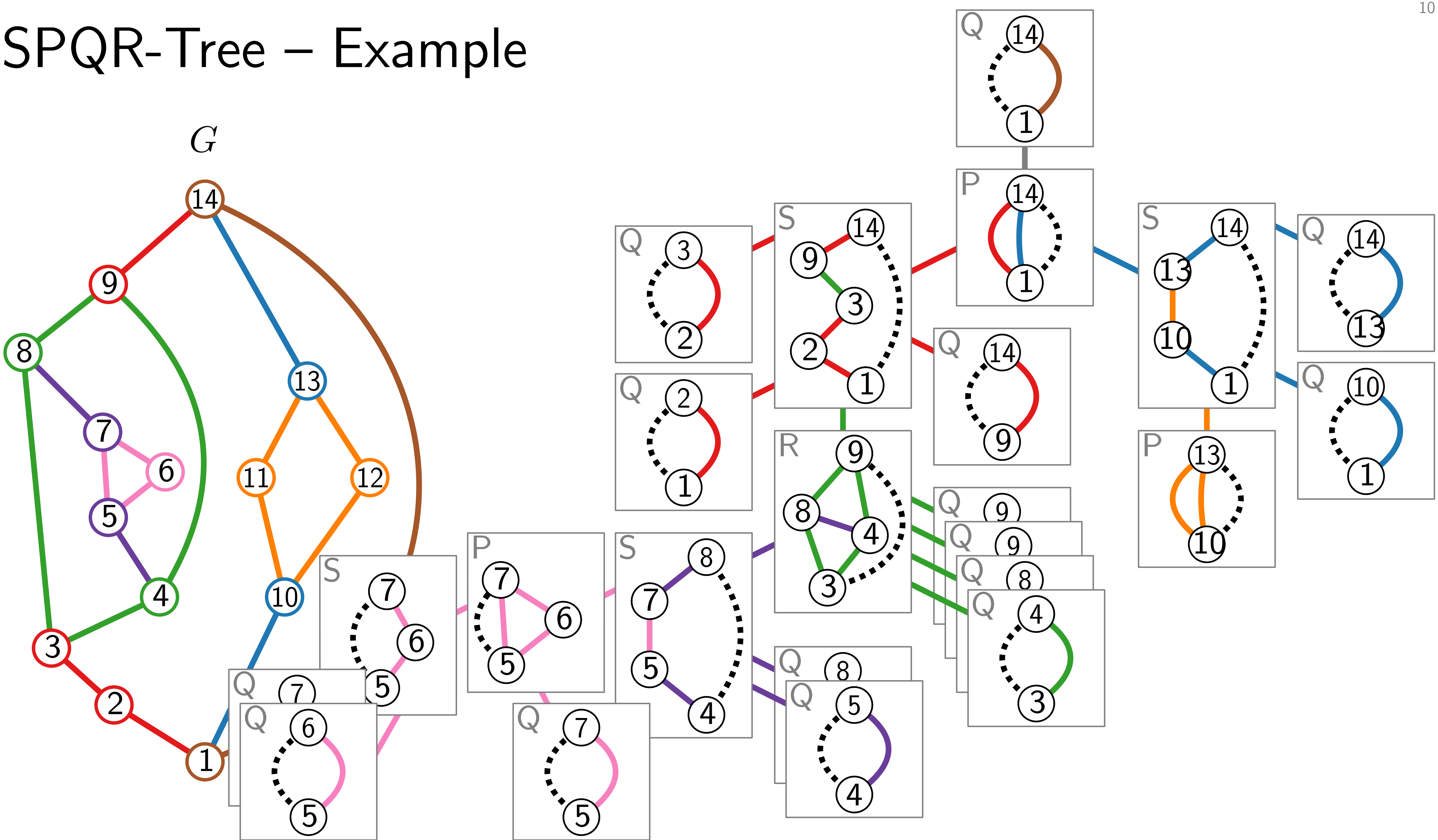
SPQR-Tree – Example



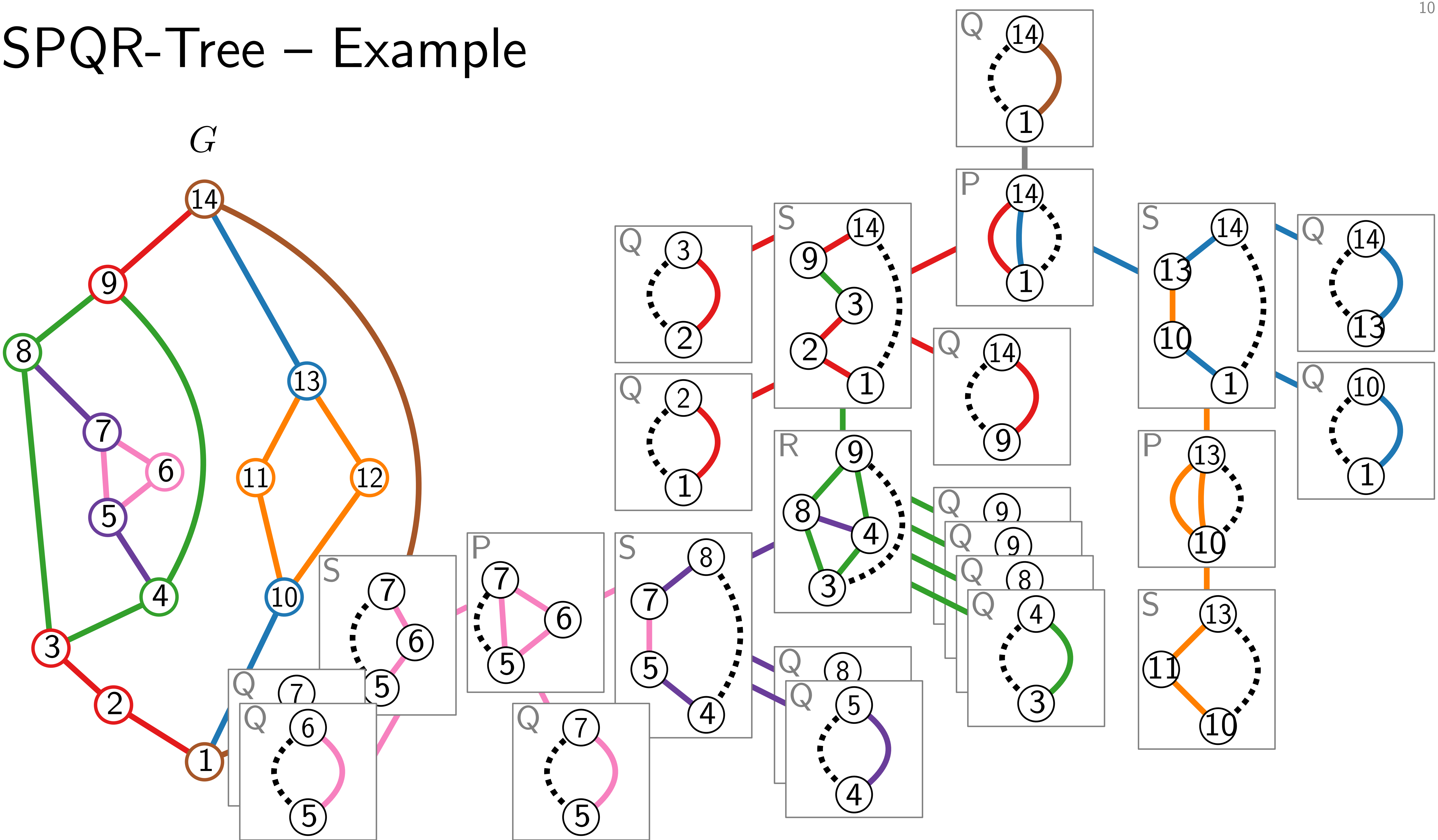
SPQR-Tree – Example



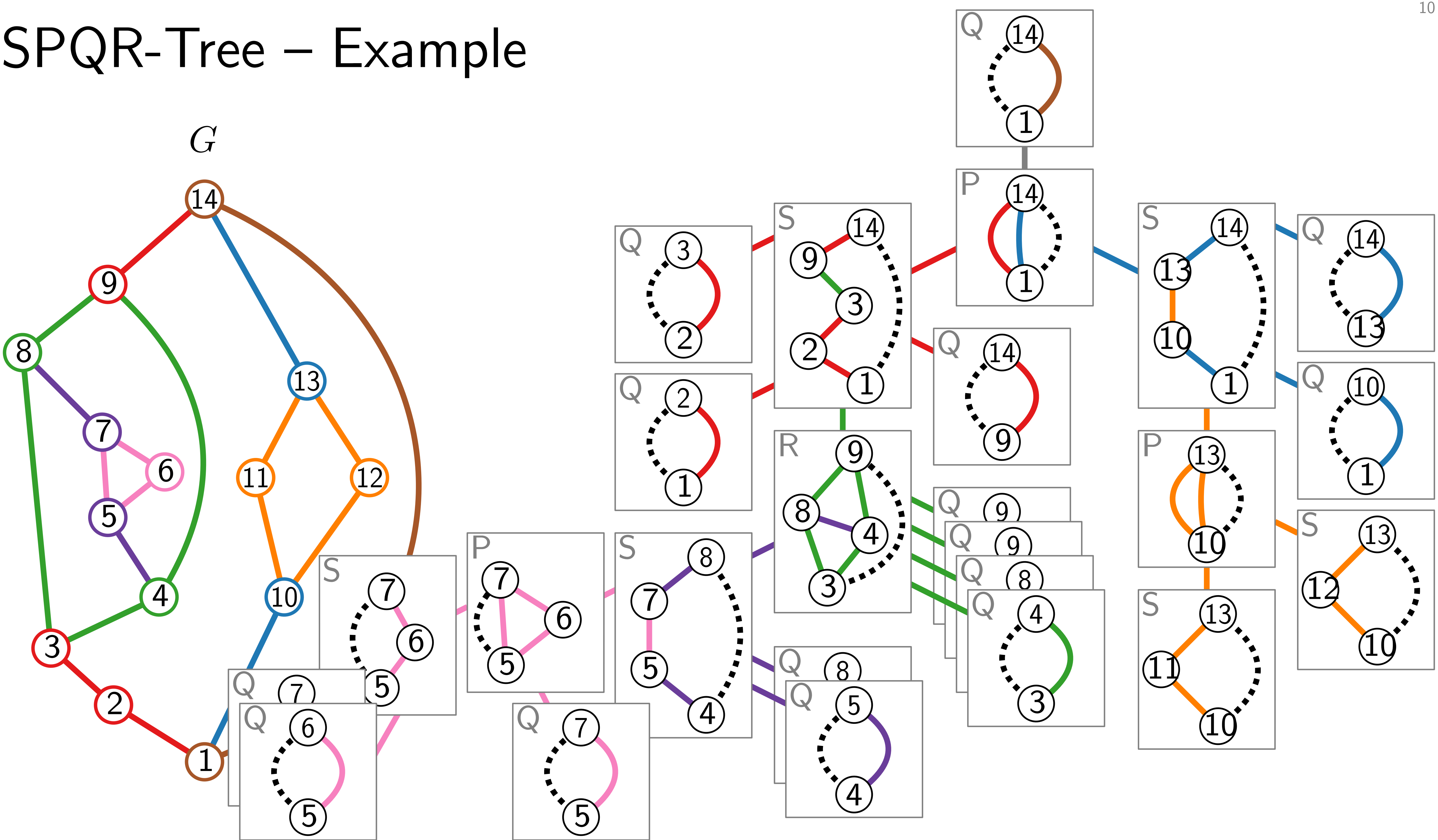
SPQR-Tree – Example



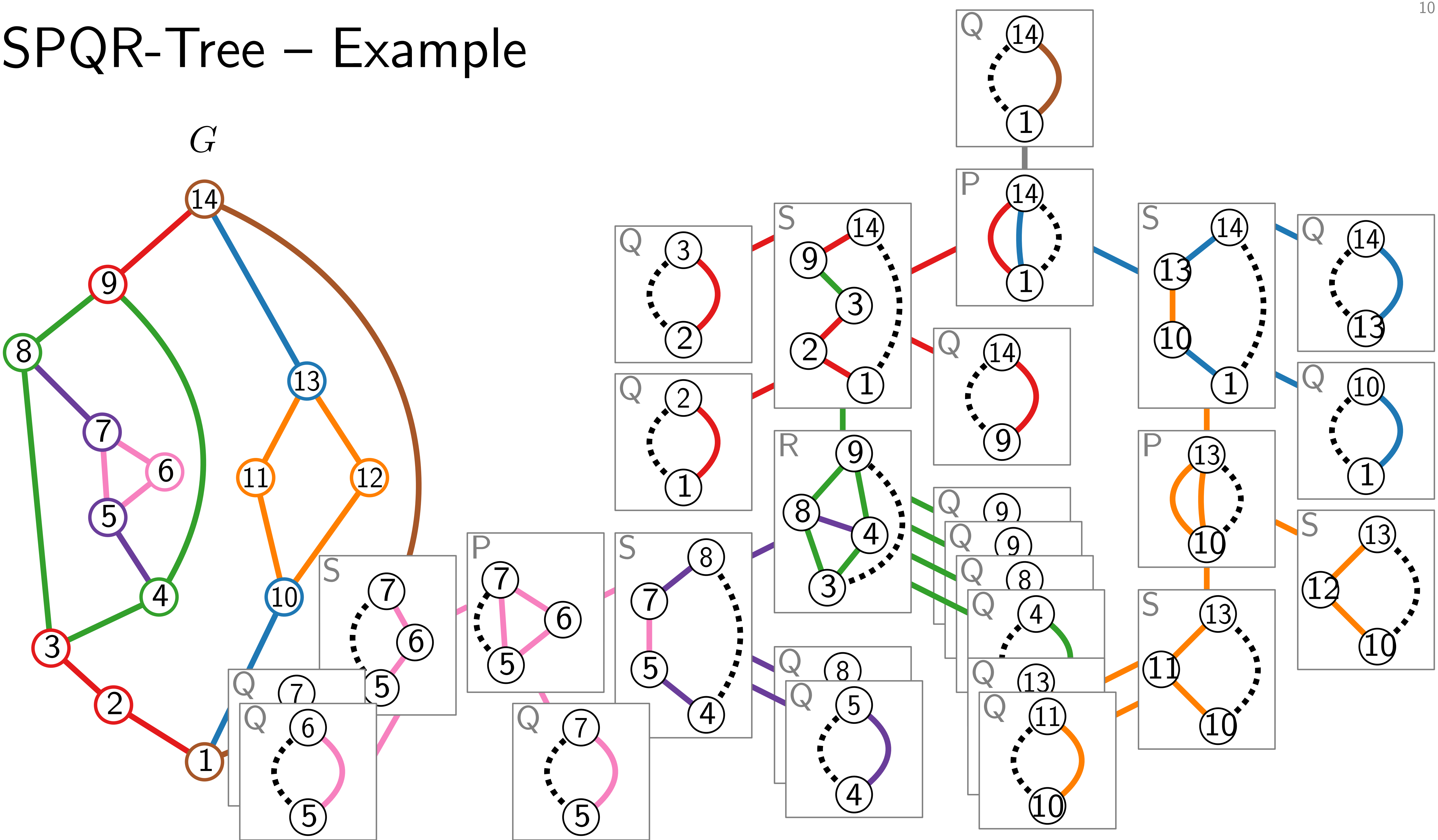
SPQR-Tree – Example



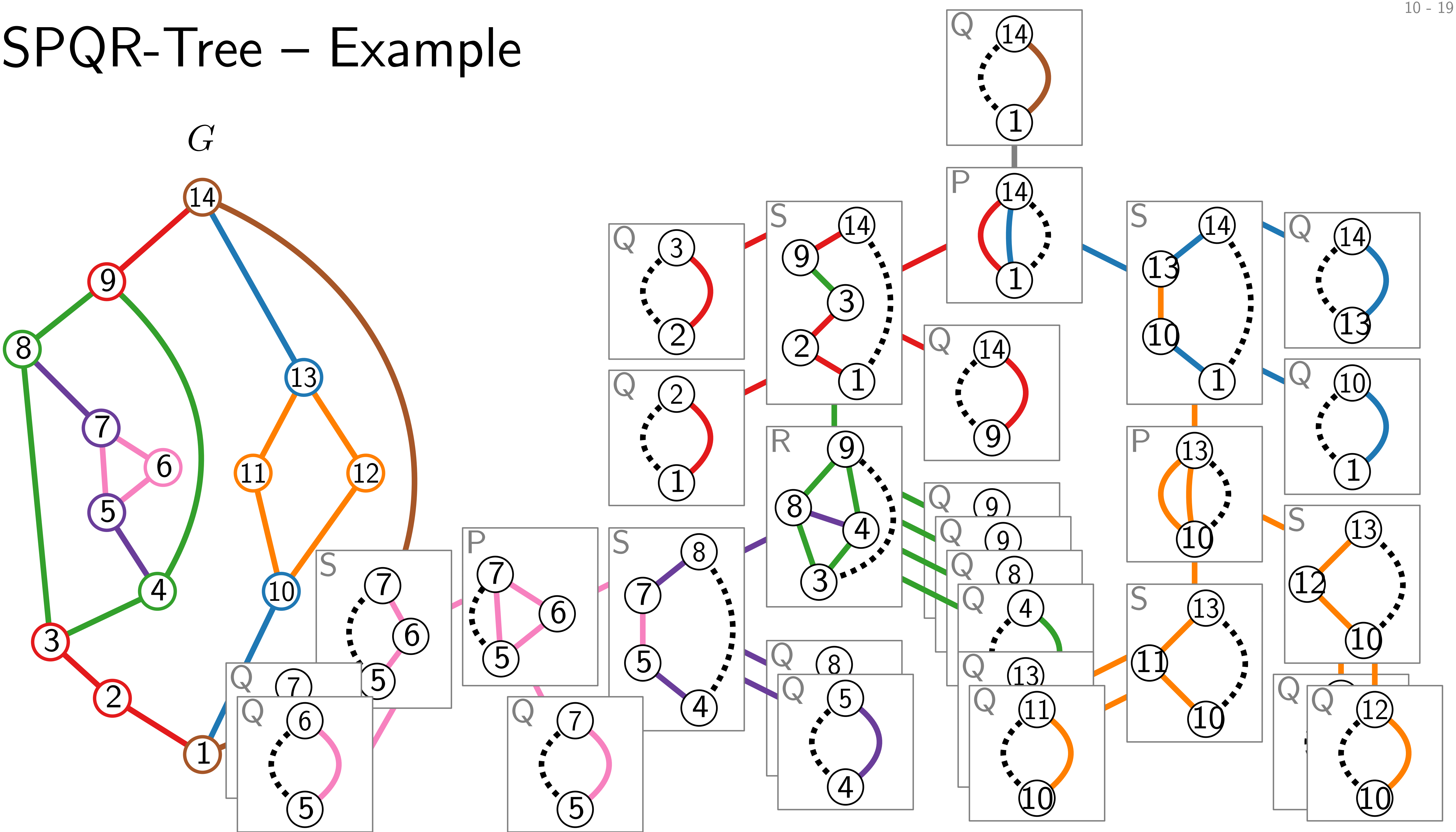
SPQR-Tree – Example



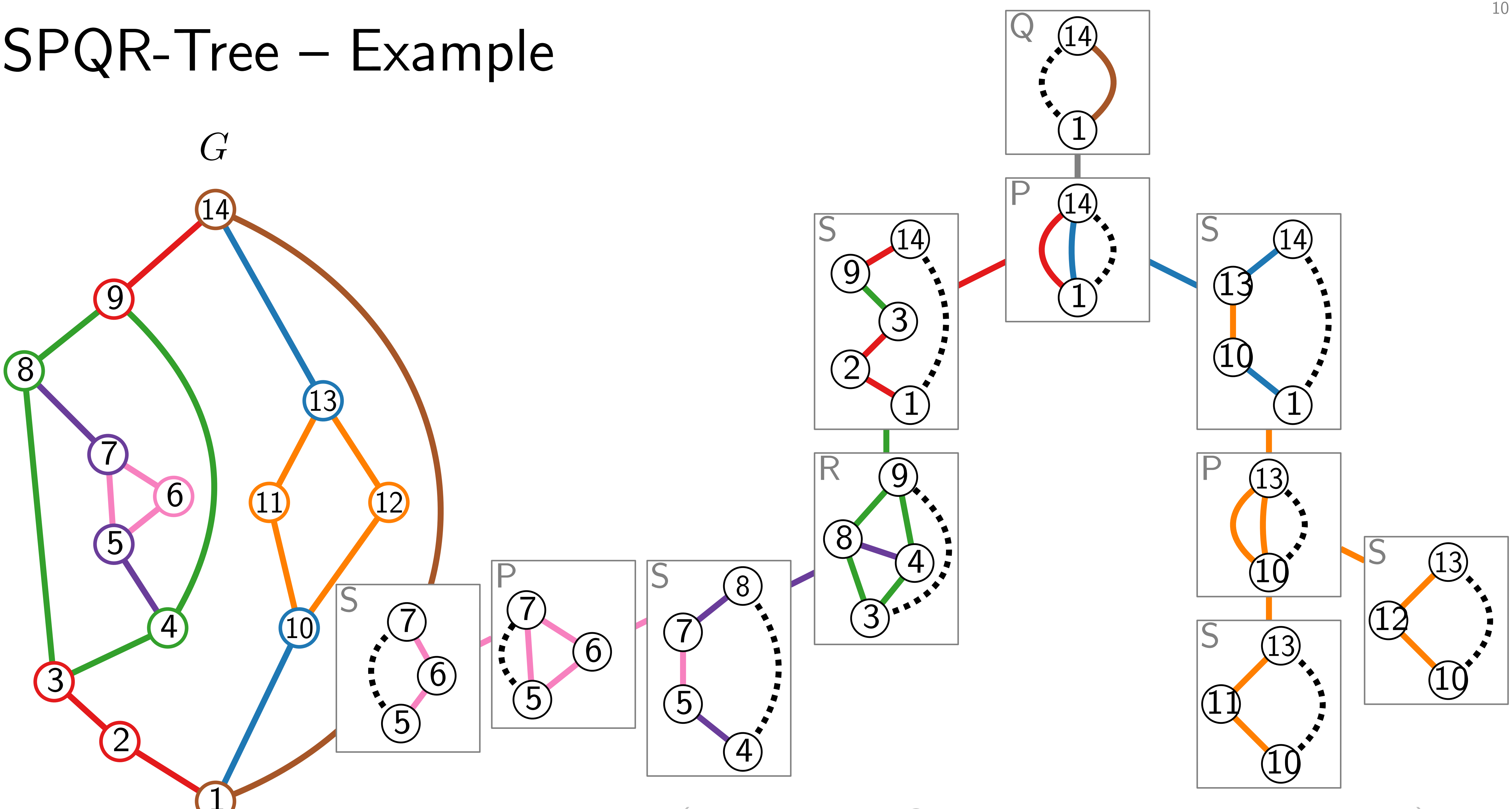
SPQR-Tree – Example



SPQR-Tree – Example



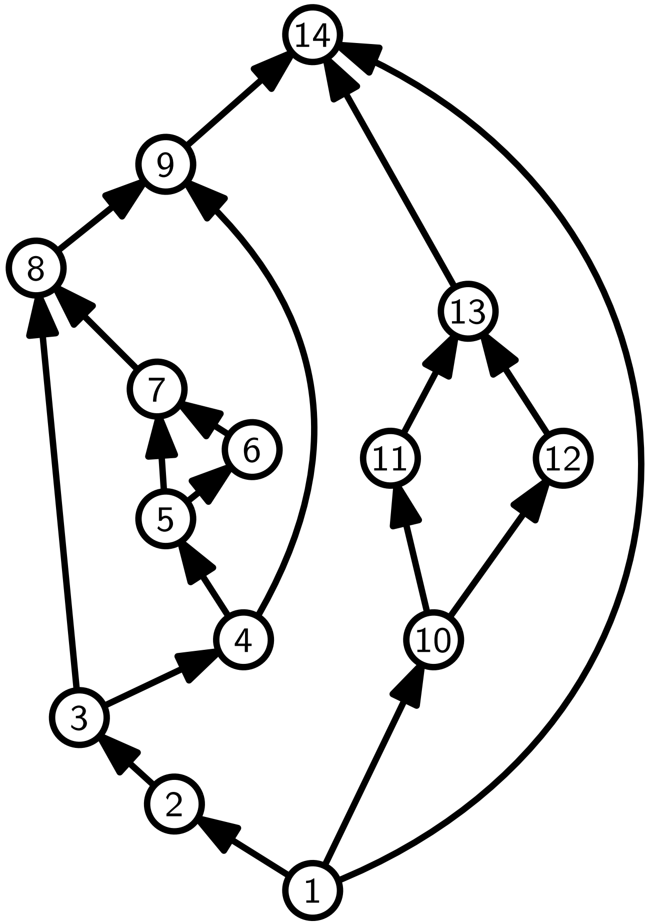
SPQR-Tree – Example



(tree without Q-nodes, but including the root)

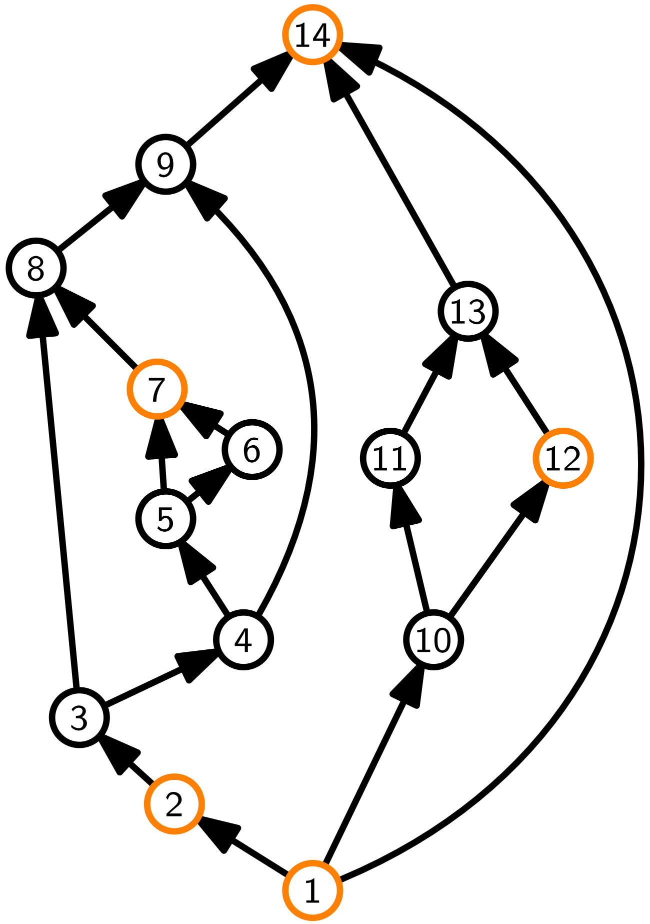
Representation Extension for st-Graphs

Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.



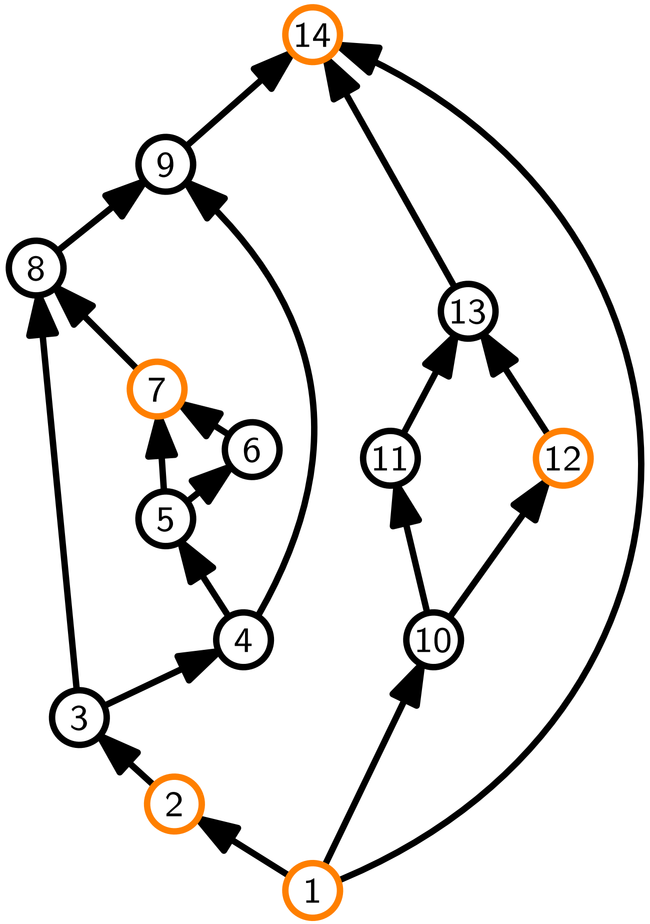
Representation Extension for st-Graphs

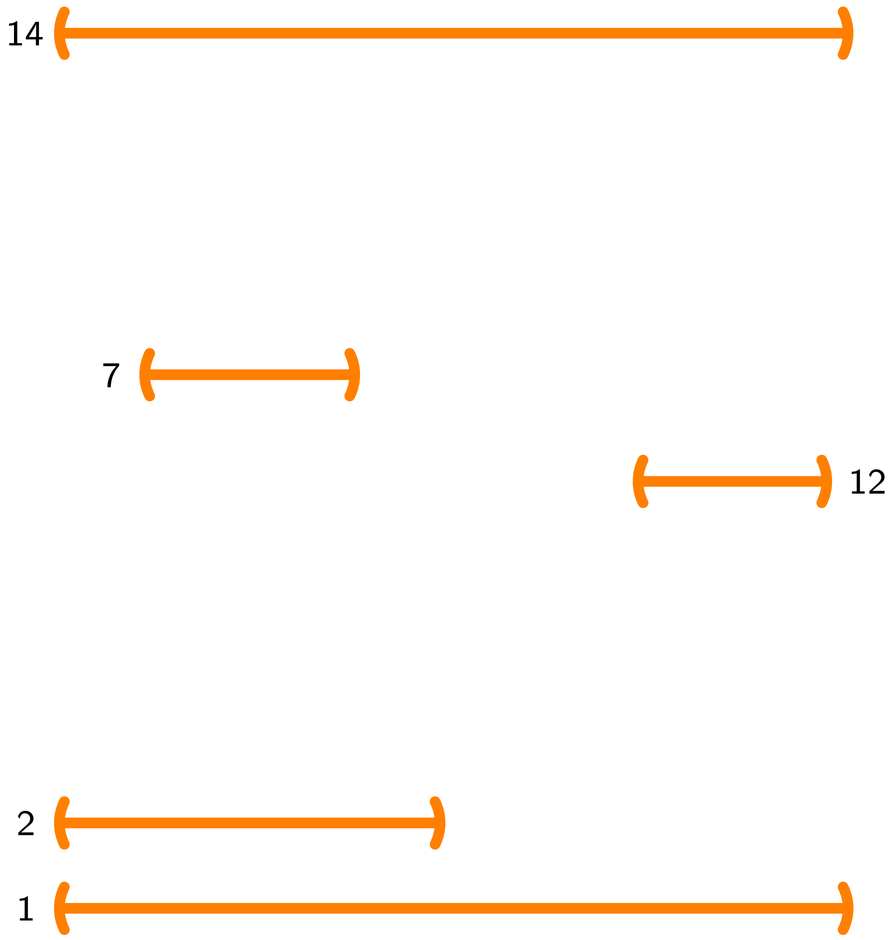
Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.



Representation Extension for st-Graphs

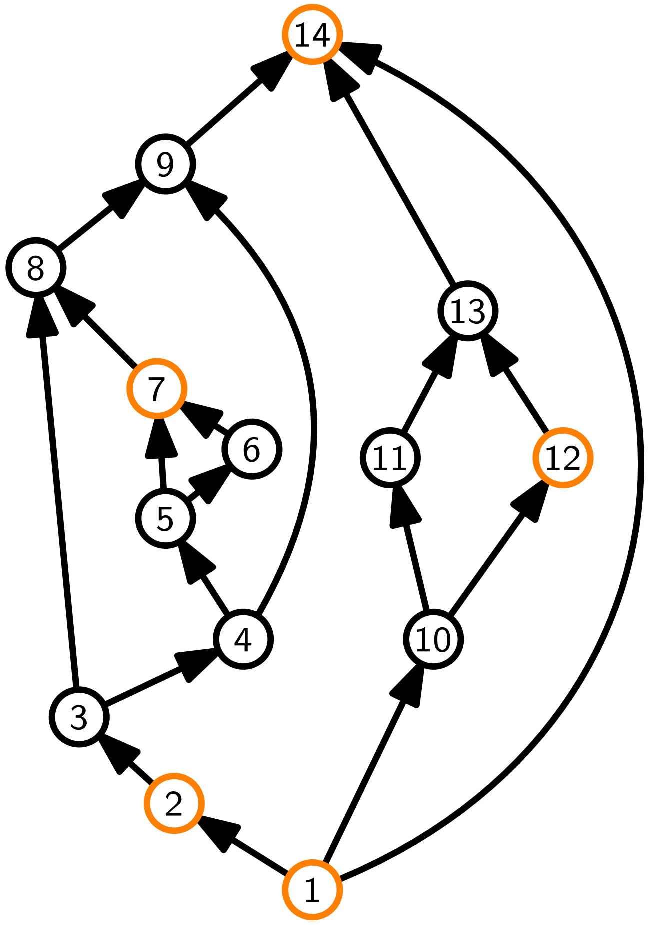
Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.

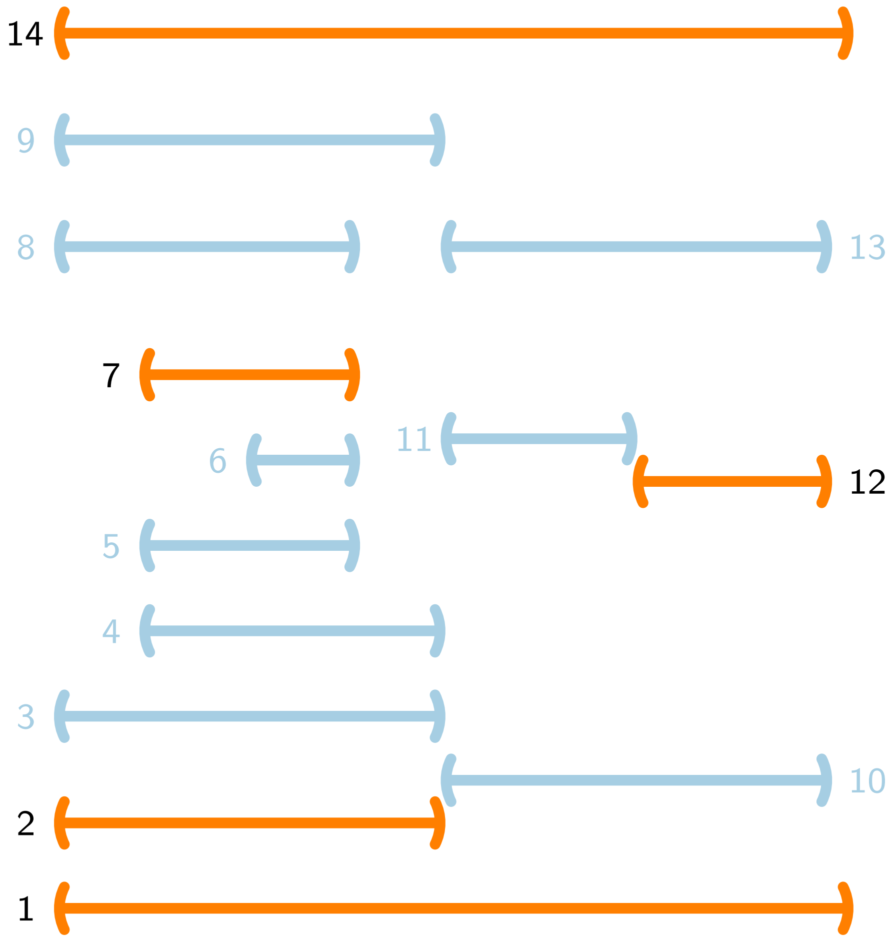




Representation Extension for st-Graphs

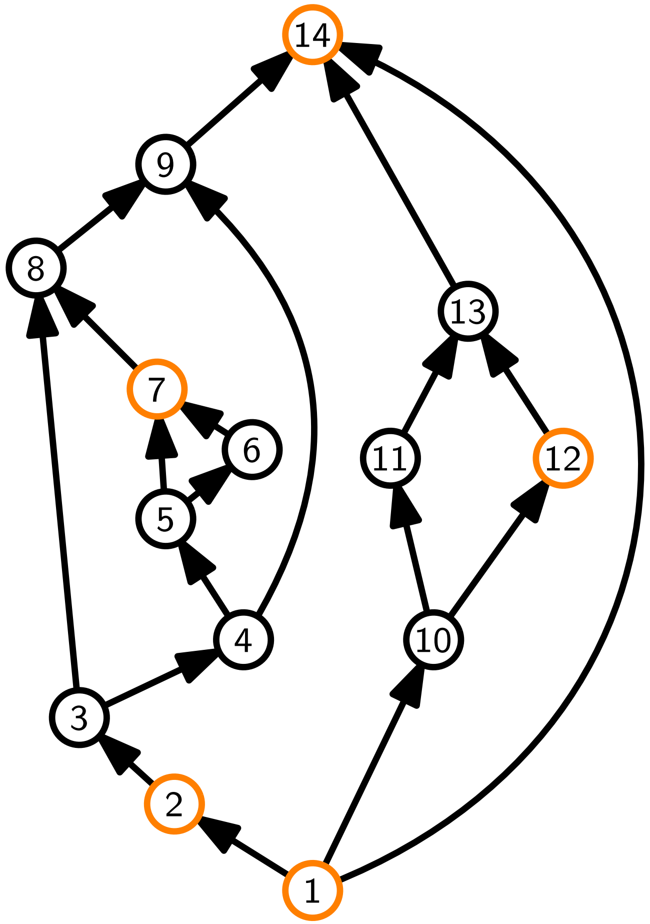
Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.

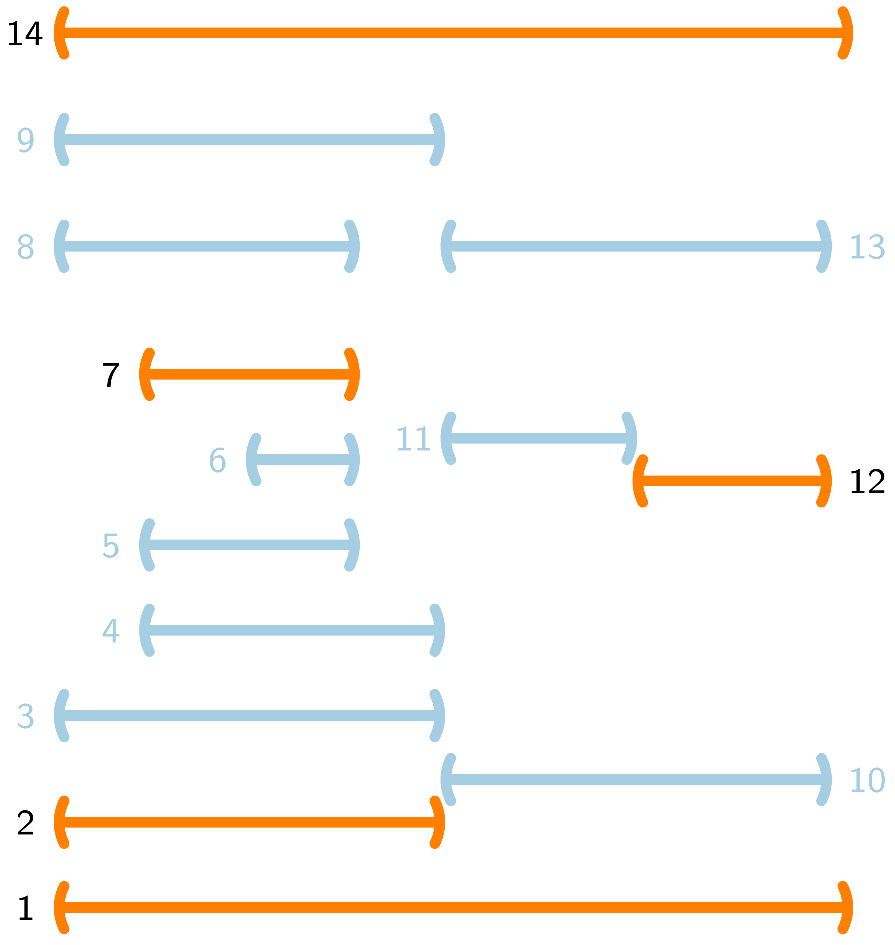




Representation Extension for st-Graphs

Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.

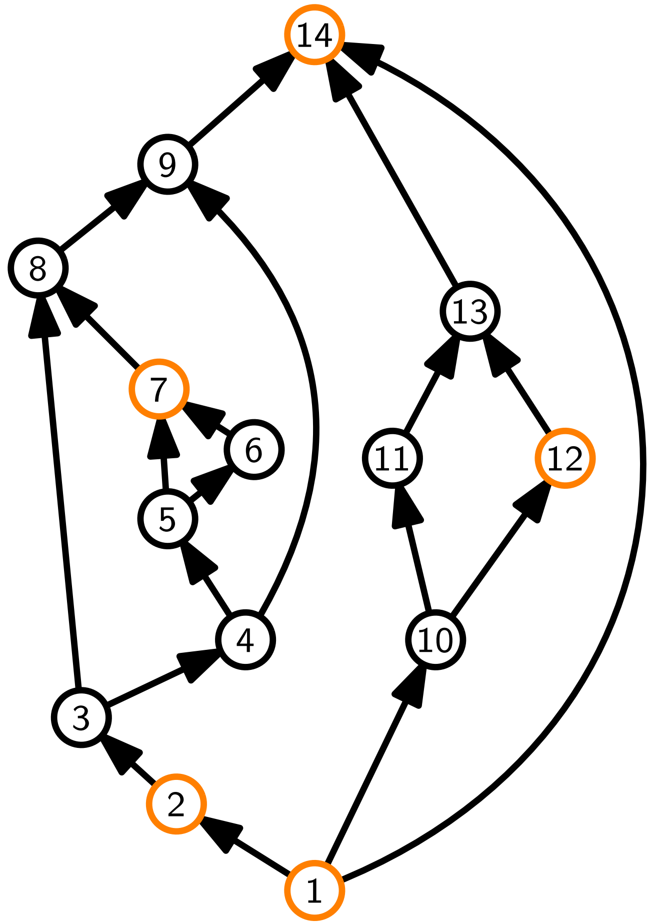


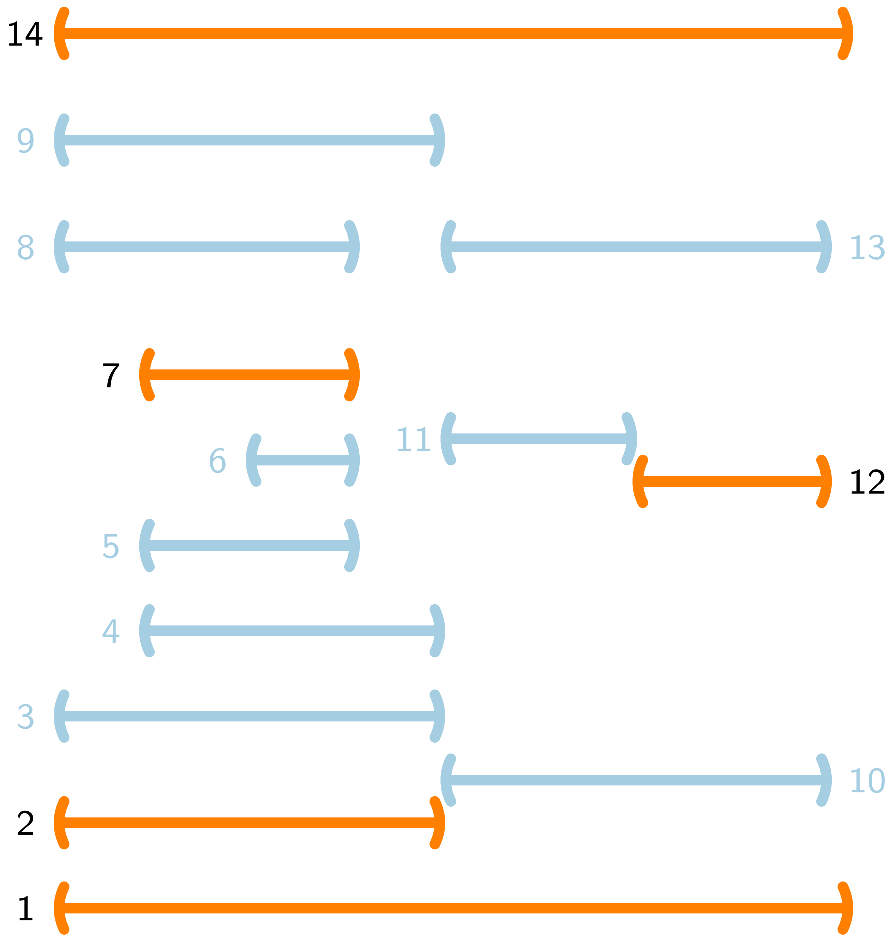


- Simplify problem via assumption regarding y-coordinates

Representation Extension for st-Graphs

Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.

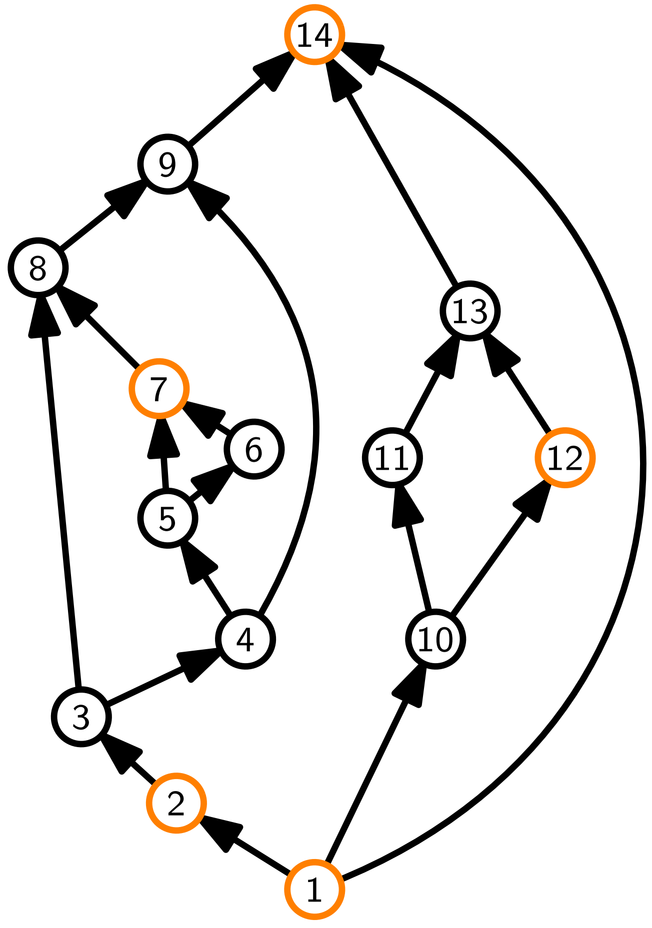


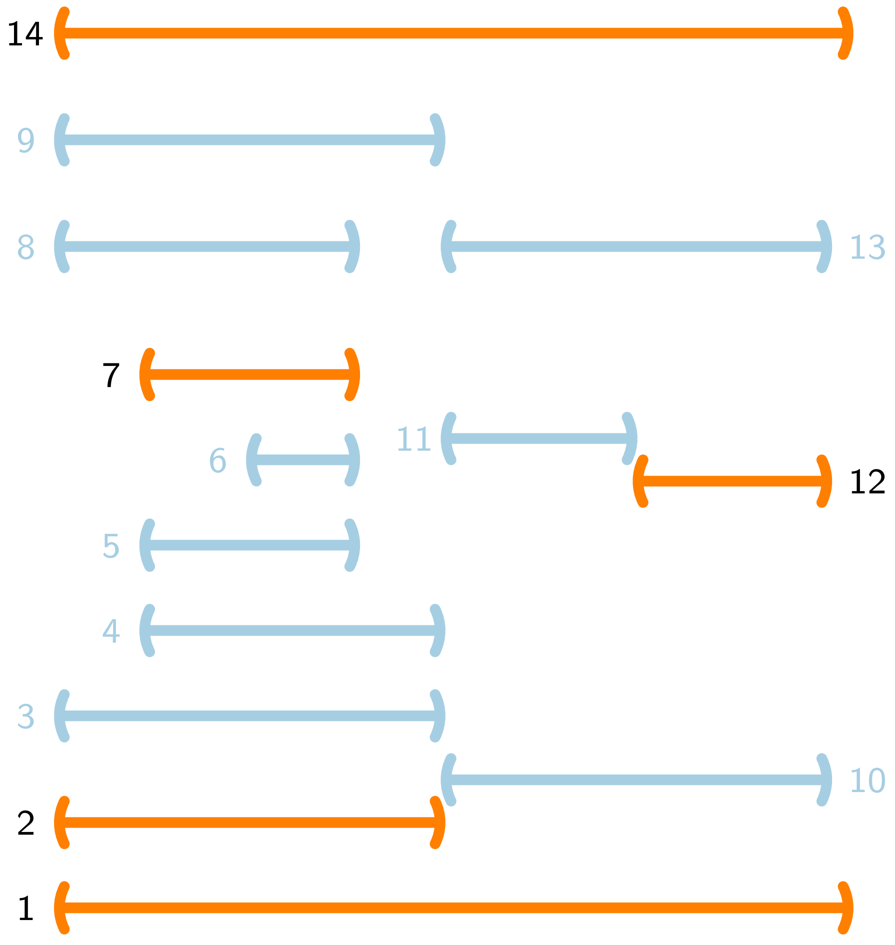


- Simplify problem via assumption regarding y-coordinates
- Exploit connection between SPQR-trees and rectangle tiling

Representation Extension for st-Graphs

Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.

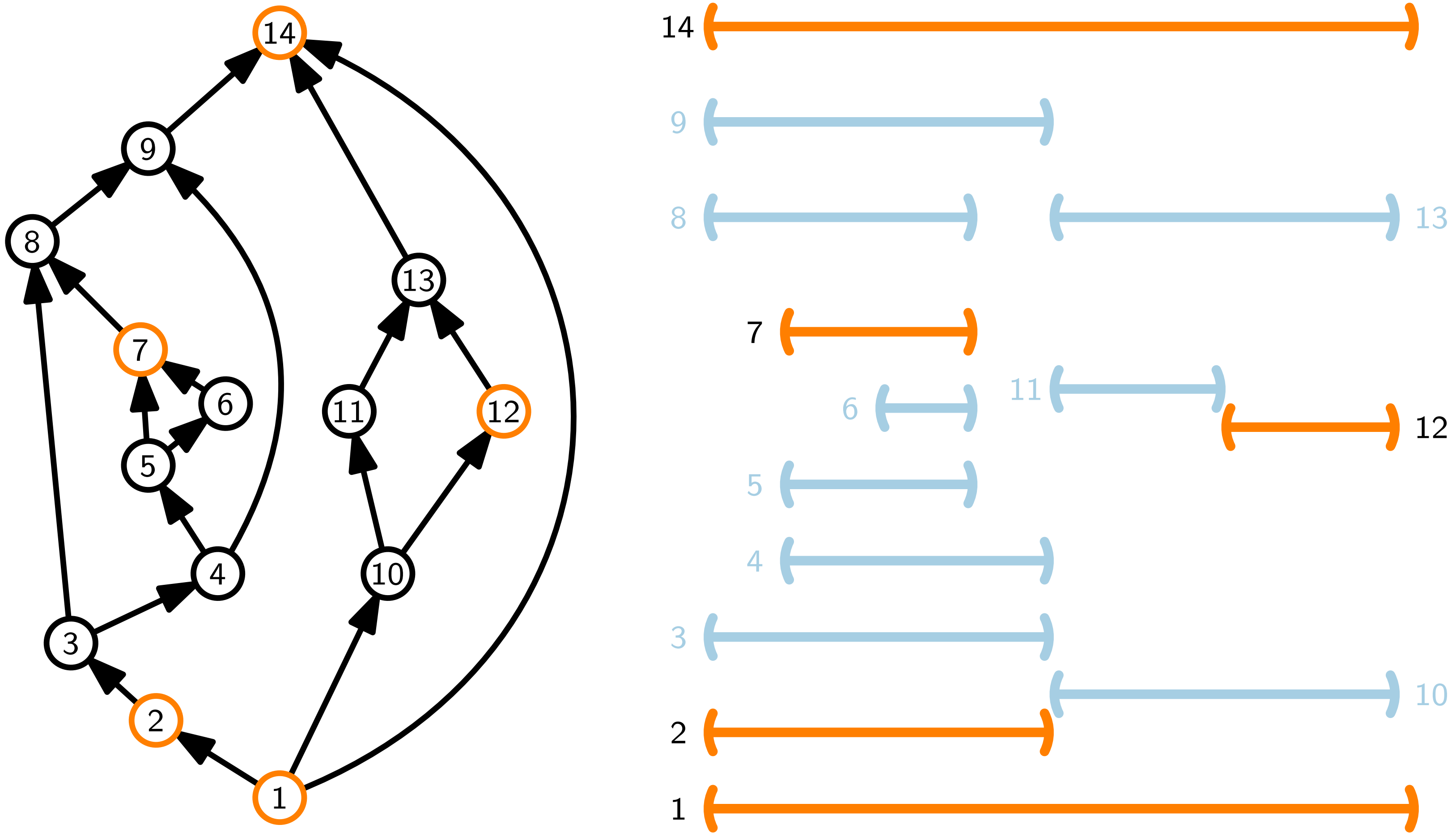




- Simplify problem via assumption regarding y-coordinates
- Exploit connection between SPQR-trees and rectangle tiling
- Solve problems for **S**-, **P**-, and **R**-nodes

Representation Extension for st-Graphs

Theorem 1'.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.



- Simplify problem via assumption regarding y-coordinates
- Exploit connection between SPQR-trees and rectangle tiling
- Solve problems for **S**-, **P**-, and **R**-nodes
- Dynamic program via structure of SPQR-tree

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y\colon V(G) \to \mathbb{R}$ such that

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y \colon V(G) \to \mathbb{R}$ such that
  - for every $v \in V'$, $y(v) =$ y-coordinate of $\psi'(v)$.

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y\colon V(G) \to \mathbb{R}$ such that
  - for every $v \in V'$, $y(v) =$ y-coordinate of $\psi'(v)$.
  - for every $(u, v) \in E(G)$, it holds that $y(u) < y(v)$.

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y \colon V(G) \to \mathbb{R}$ such that
  - for every $v \in V'$, $y(v) =$ y-coordinate of $\psi'(v)$.
  - for every $(u, v) \in E(G)$, it holds that $y(u) < y(v)$.

Lemma 1.
G has a representation extending ψ' $\Leftrightarrow$
G has a representation extending ψ'
where the y-coordinates of the bars are as in y.

y-Coordinate Invariant

- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y\colon V(G) \to \mathbb{R}$ such that
  - for every $v \in V'$, $y(v) =$ y-coordinate of $\psi'(v)$.
  - for every $(u, v) \in E(G)$, it holds that $y(u) < y(v)$.

Lemma 1.
G has a representation extending ψ' $\Leftrightarrow$
G has a representation extending ψ'
where the y-coordinates of the bars are as in y.

Proof idea. The relative positions of **adjacent** bars must match the order given by y.
So, we can adjust the y-coordinates of any solution to be as in y by sweeping from bottom to top.

y-Coordinate Invariant

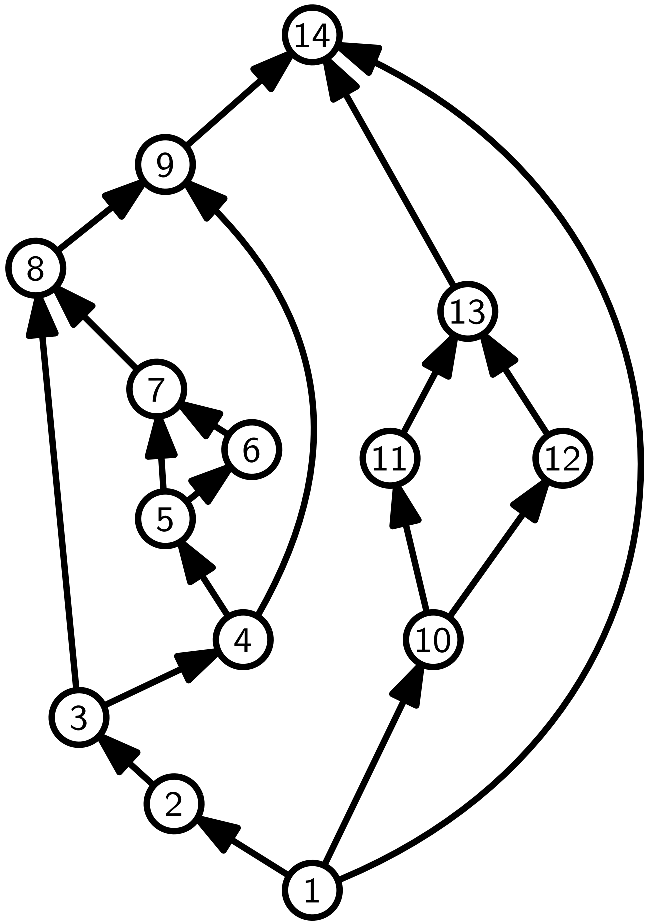
- Let G be an st-graph, and let ψ' be a representation of $V' \subseteq V(G)$.
- Let $y \colon V(G) \to \mathbb{R}$ such that
  - for every $v \in V'$, $y(v) =$ y-coordinate of $\psi'(v)$.
  - for every $(u, v) \in E(G)$, it holds that $y(u) < y(v)$.

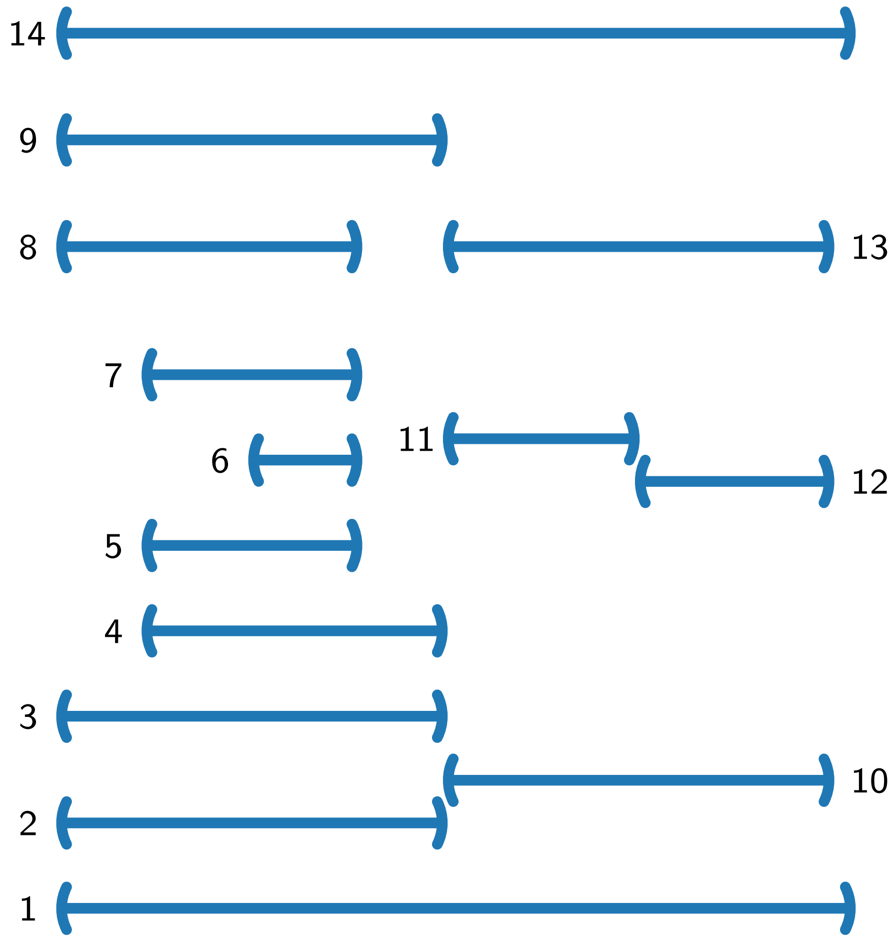
Lemma 1.
G has a representation extending ψ' $\Leftrightarrow$
G has a representation extending ψ'
where the y-coordinates of the bars are as in y.

We can now assume that all y-coordinates are given!

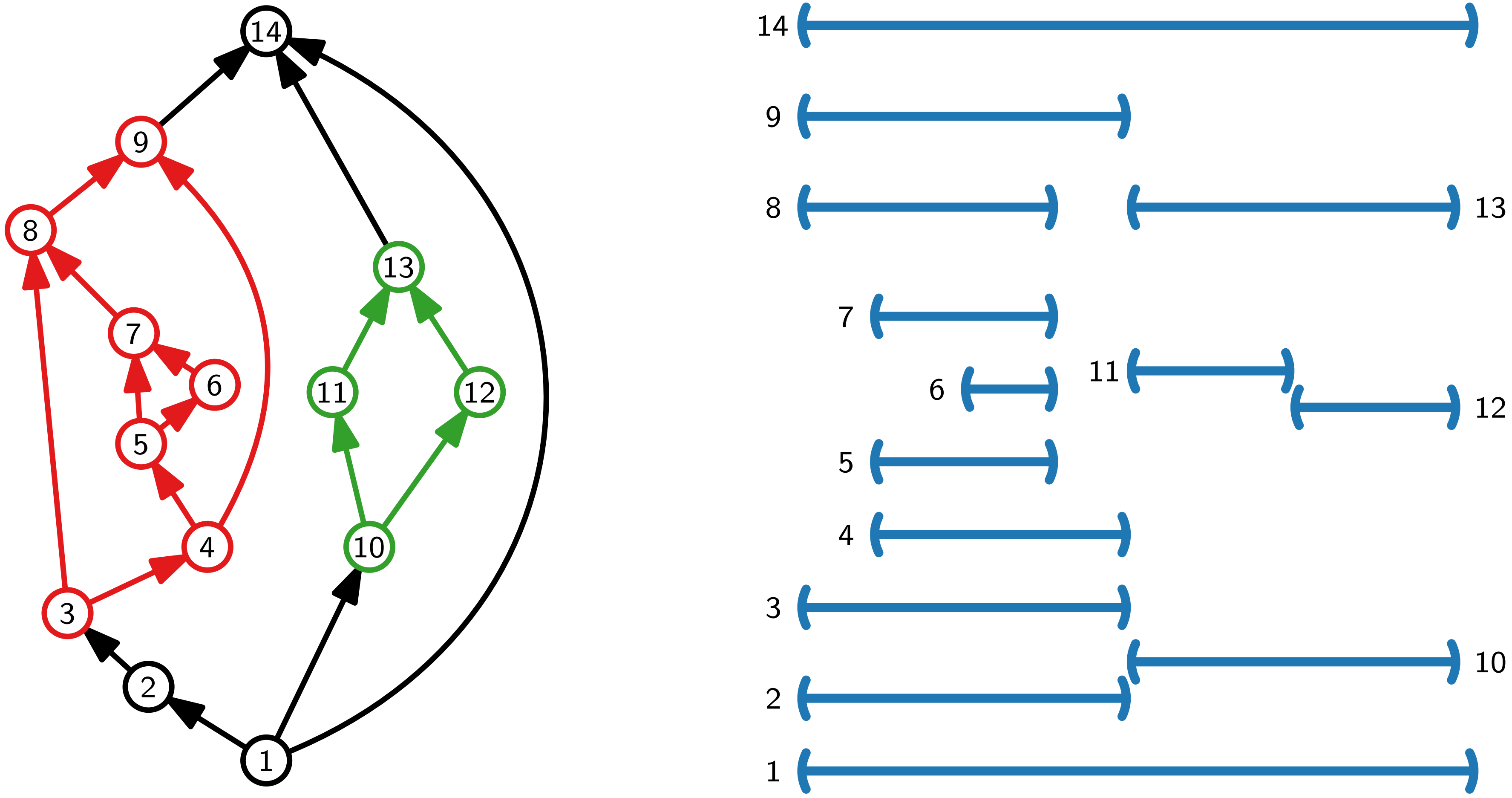
Proof idea. The relative positions of **adjacent** bars must match the order given by y.
So, we can adjust the y-coordinates of any solution to be as in y by sweeping from bottom to top.

But Why Do SPQR-Trees Help?

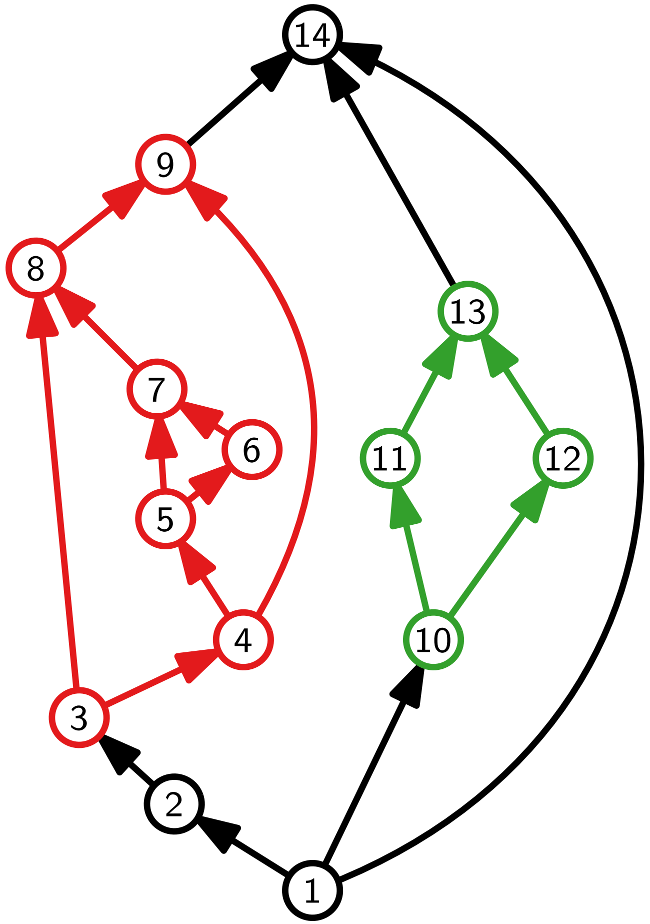


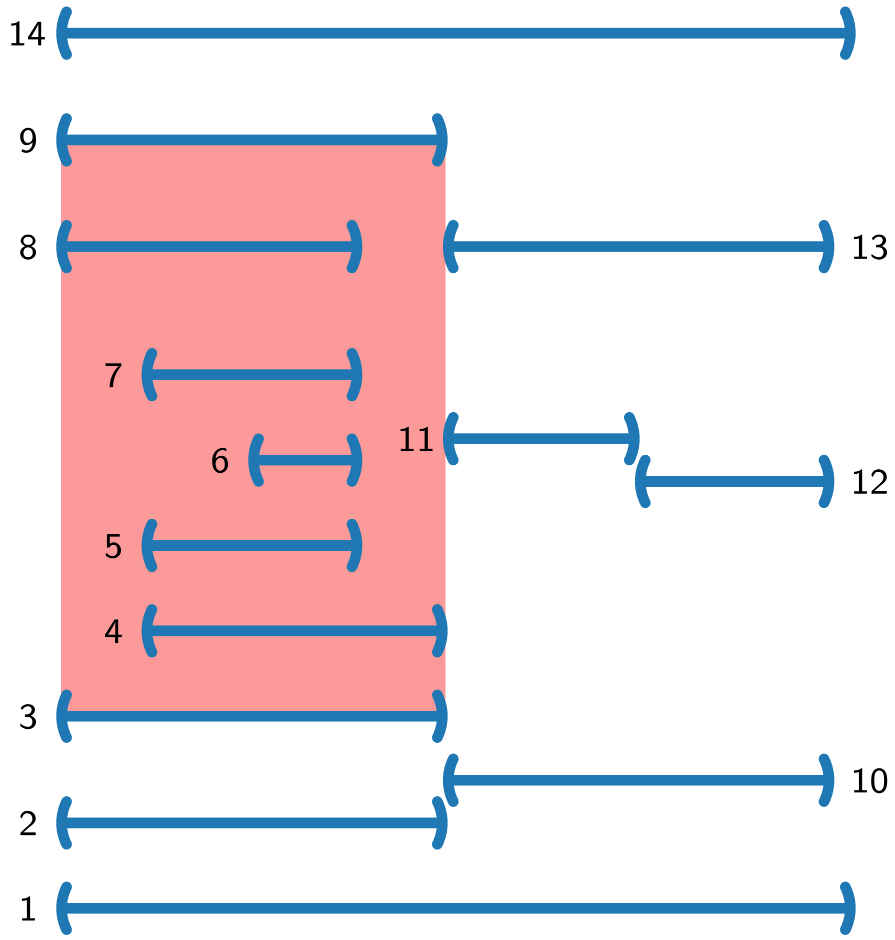


But Why Do SPQR-Trees Help?

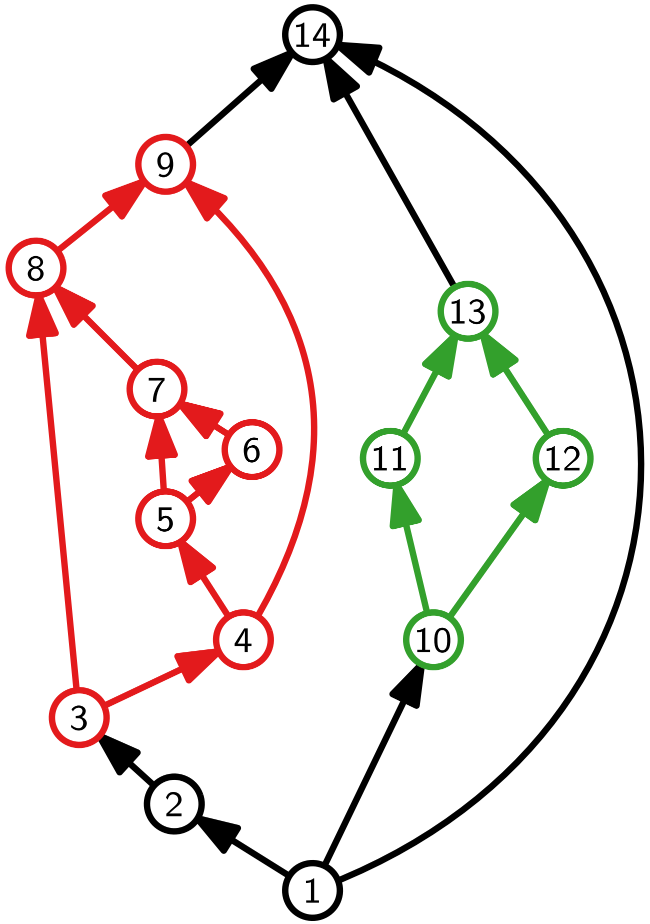


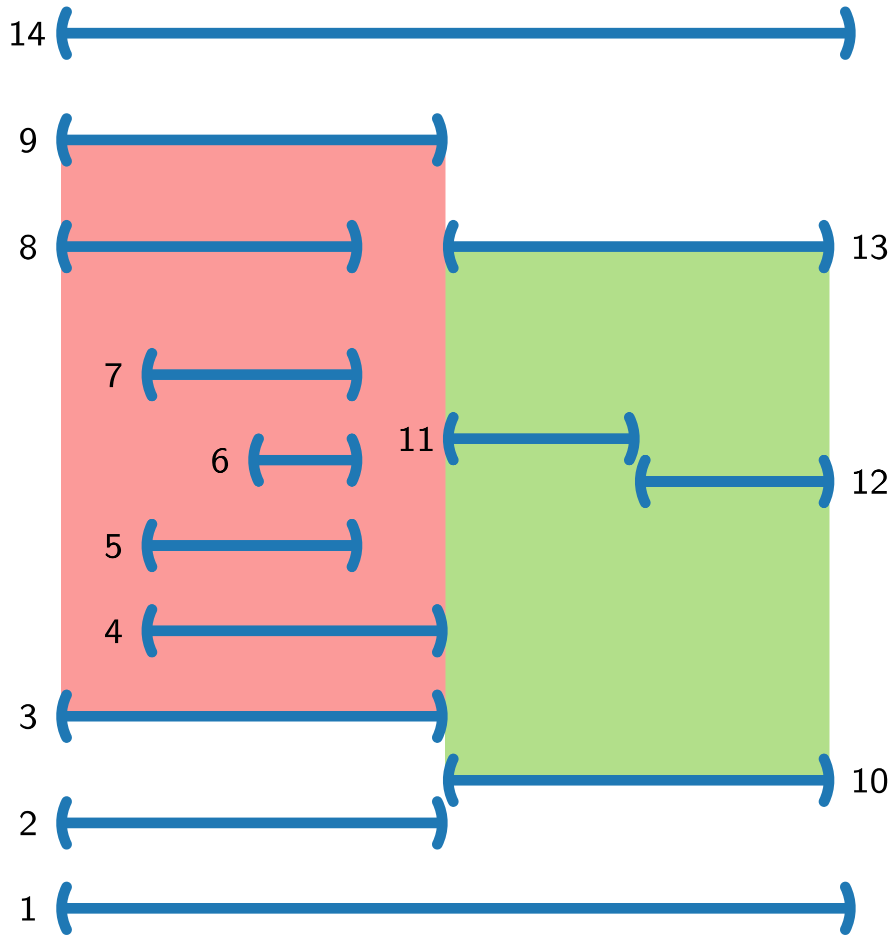
But Why Do SPQR-Trees Help?





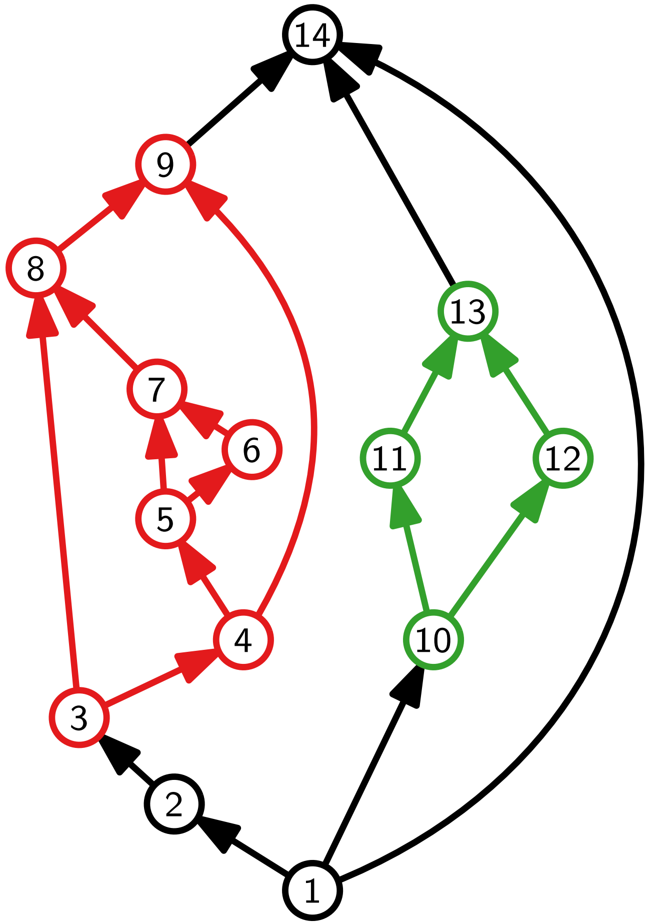
But Why Do SPQR-Trees Help?

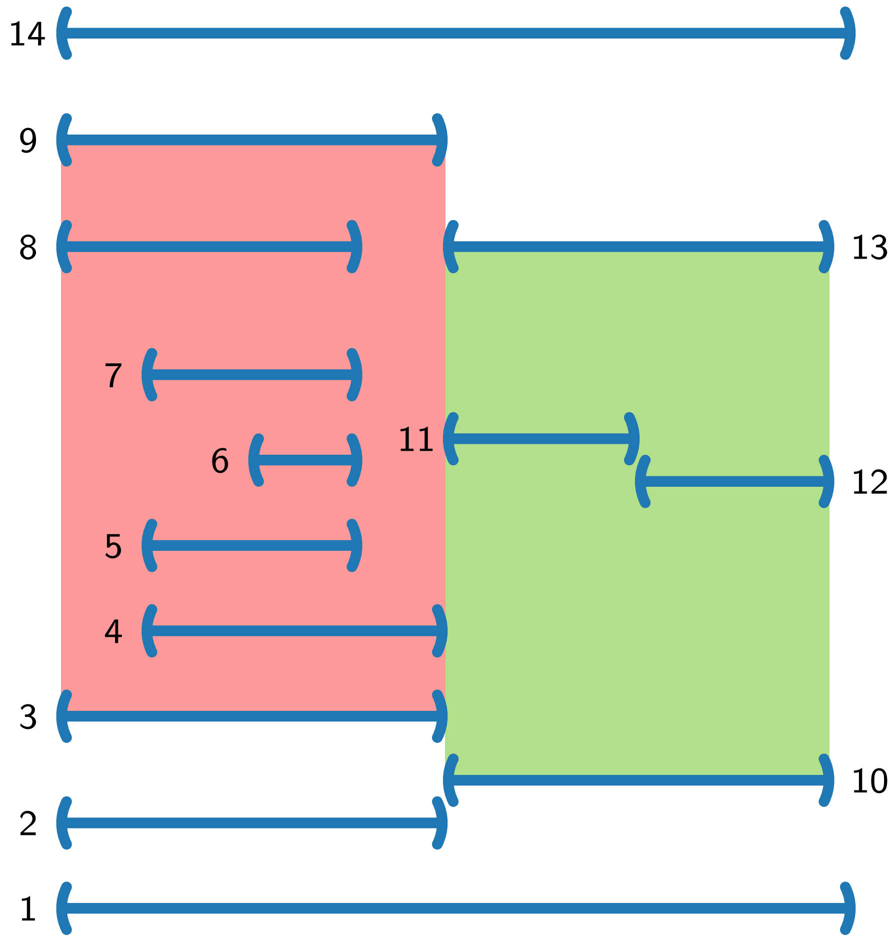




But Why Do SPQR-Trees Help?

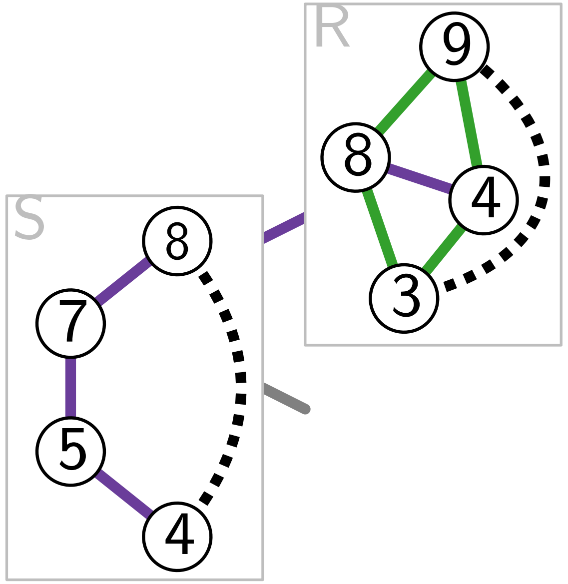
Lemma 2.
The SPQR-tree of an st-graph G induces a recursive **tiling** of any ε-bar visibility representation of G.



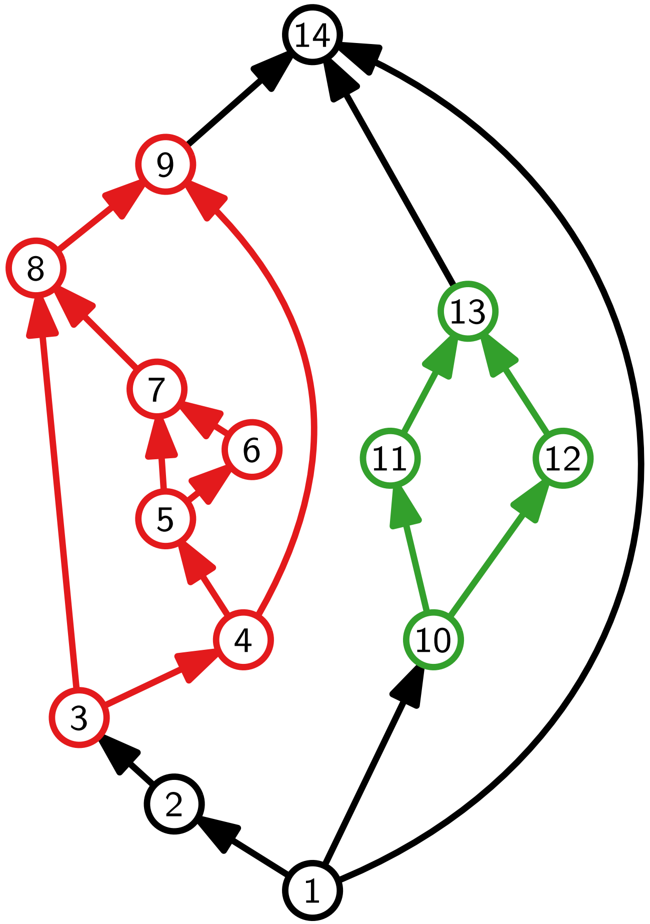


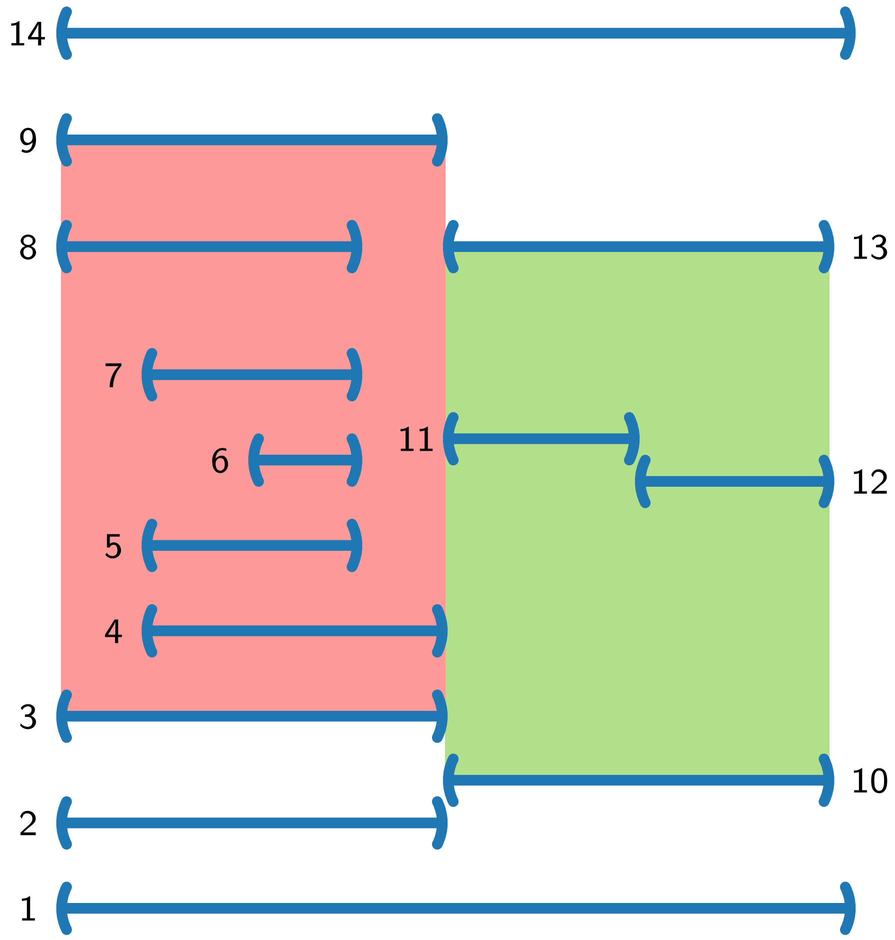
But Why Do SPQR-Trees Help?

Lemma 2.
The SPQR-tree of an st-graph G induces a recursive **tiling** of any ε-bar visibility representation of G.



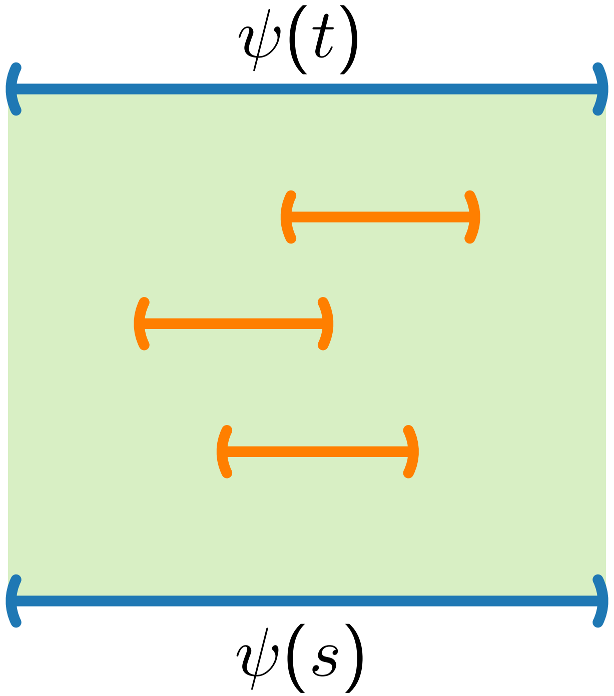
Solve tiles bottom-up.





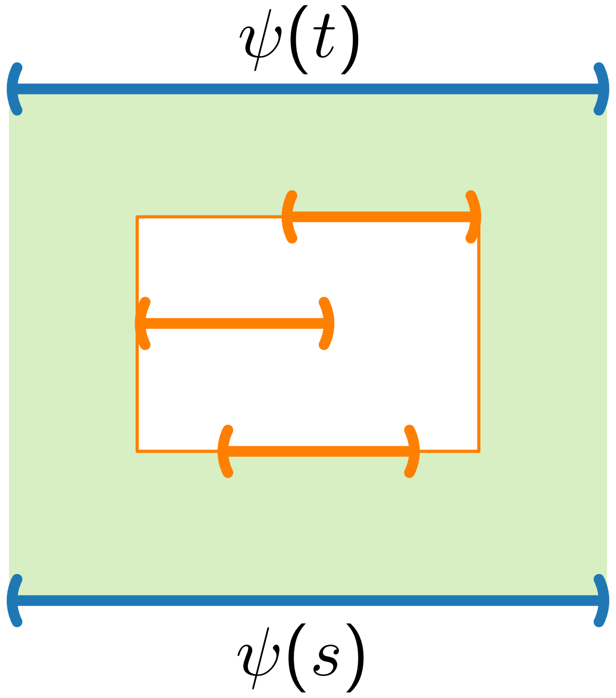
Tiles

Convention. **Orange** bars are from the given partial representation.



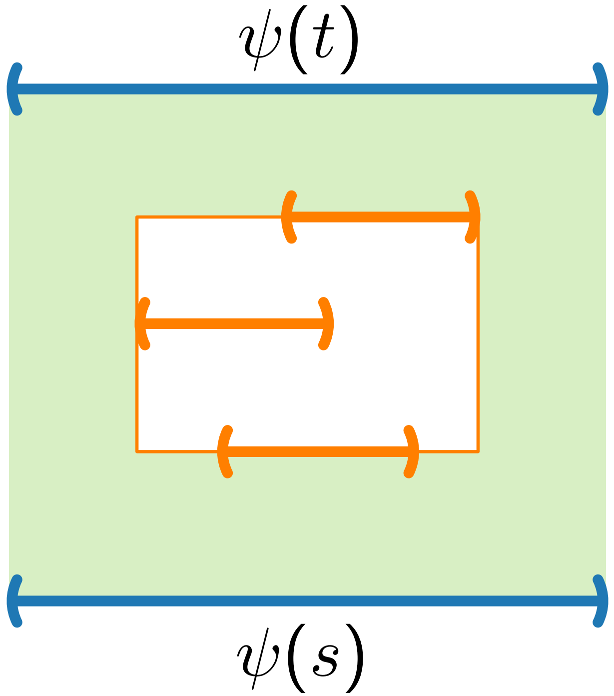
Tiles

Convention. **Orange** bars are from the given partial representation.



Tiles

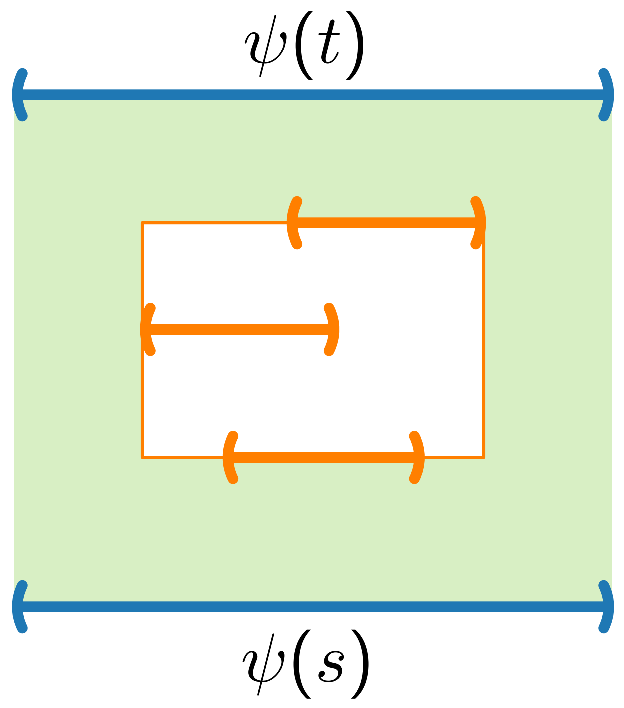
Convention. **Orange** bars are from the given partial representation.



Observation.
The bounding box (tile) of any solution ψ **contains** the bounding box of the partial representation.

Tiles

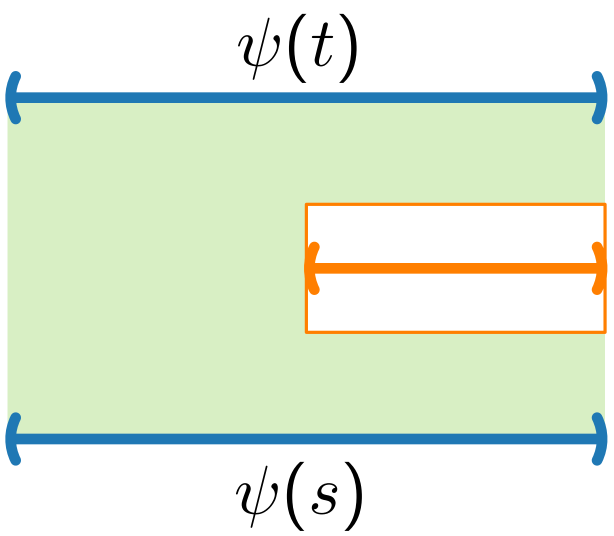
Convention. **Orange** bars are from the given partial representation.



Observation.
The bounding box (tile) of any solution ψ **contains** the bounding box of the partial representation.

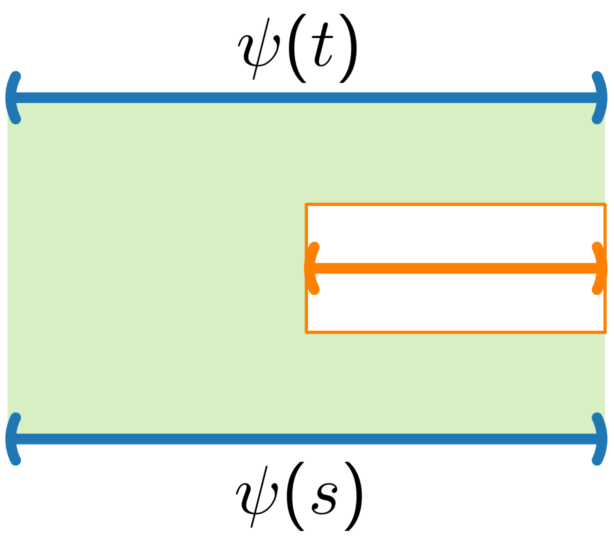
How many different **types** of tiles are there?

Types of Tiles



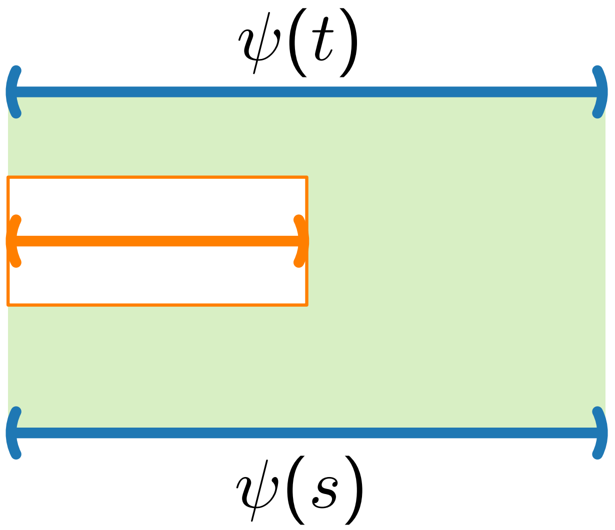
- Left **L**oose
- Right **F**ixed

Types of Tiles

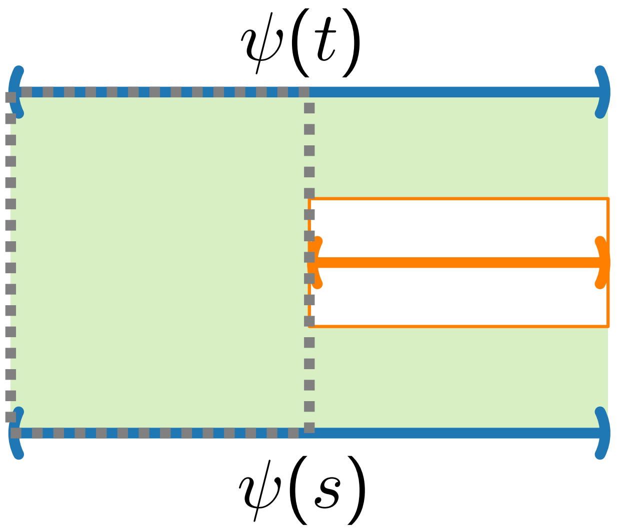


- Left **L**oose
- Right **F**ixed

- Left **F**ixed
- Right **L**oose

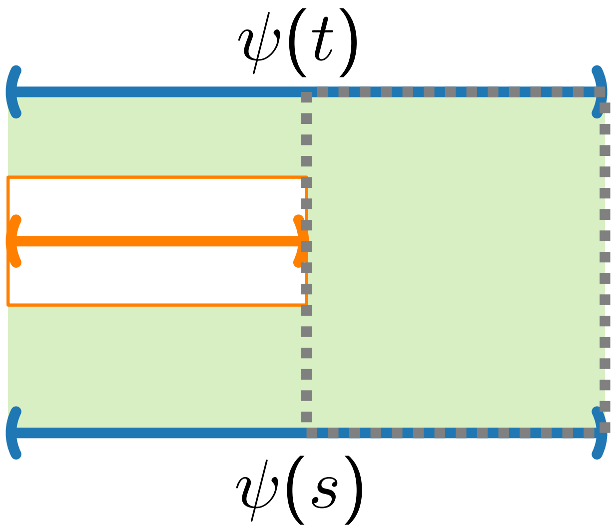


Types of Tiles

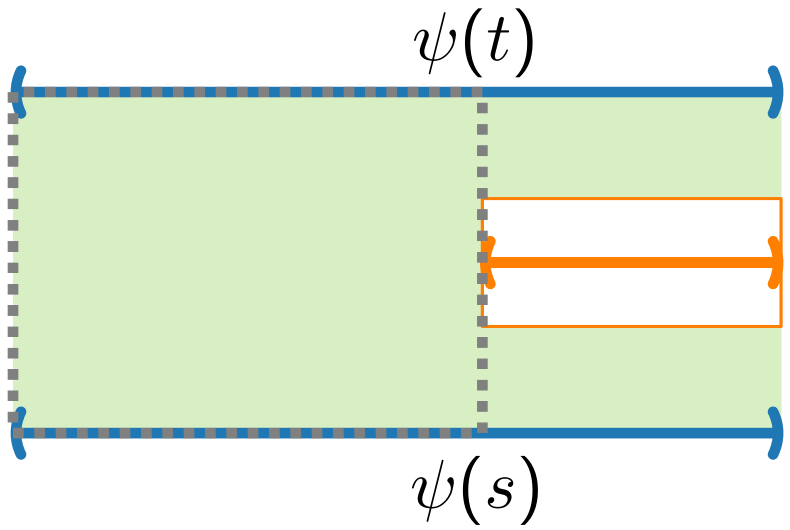


- Left **L**oose
- Right **F**ixed

- Left **F**ixed
- Right **L**oose

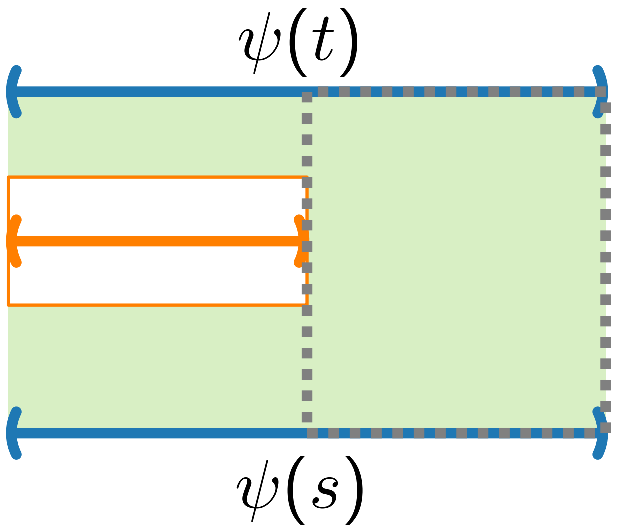


Types of Tiles

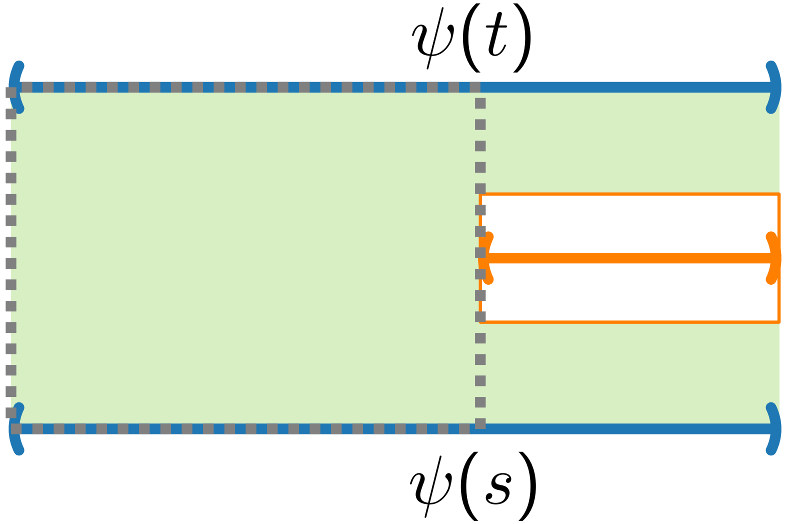


- Left **L**oose
- Right **F**ixed

- Left **F**ixed
- Right **L**oose

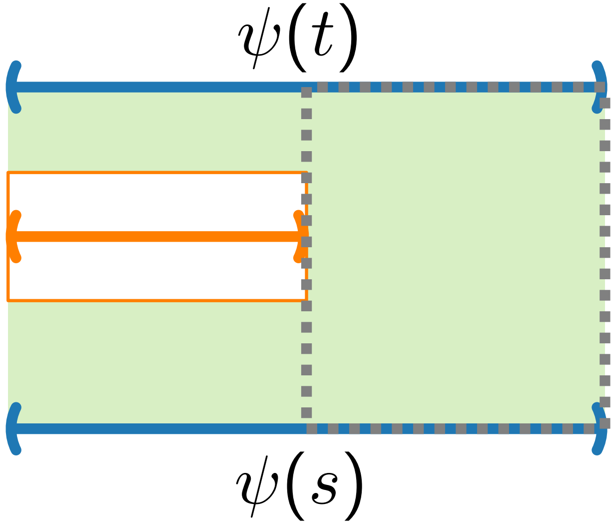


Types of Tiles



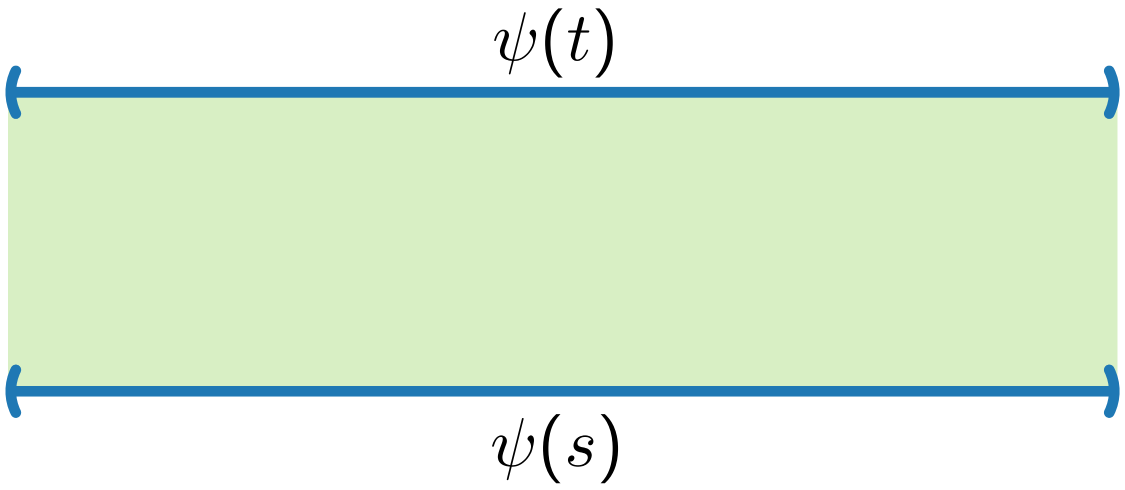
- Left **L**oose
- Right **F**ixed

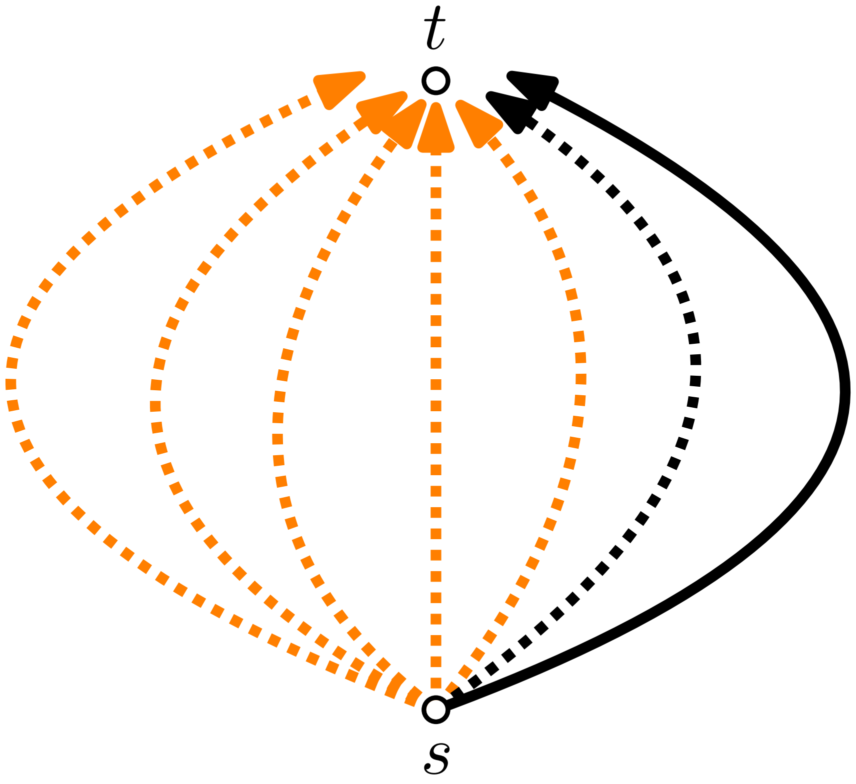
- Left **F**ixed
- Right **L**oose



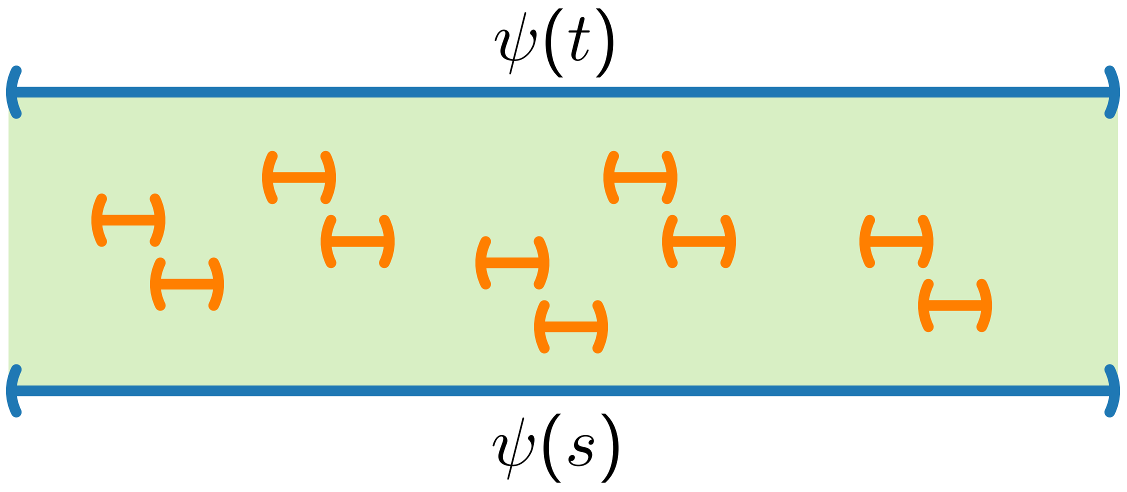
Four different types: **FF, FL, LF, LL**

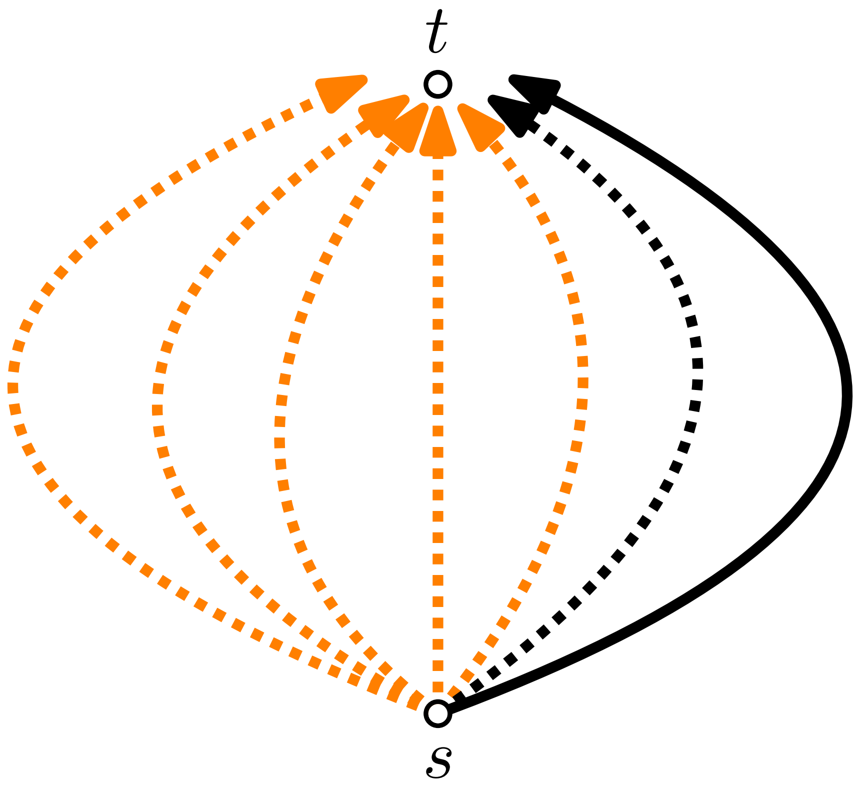
P-Nodes



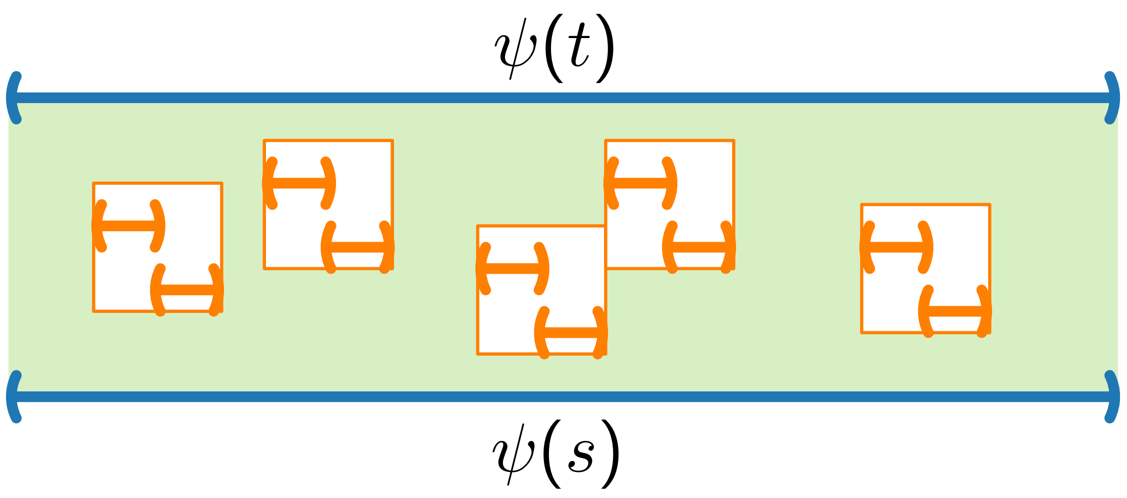


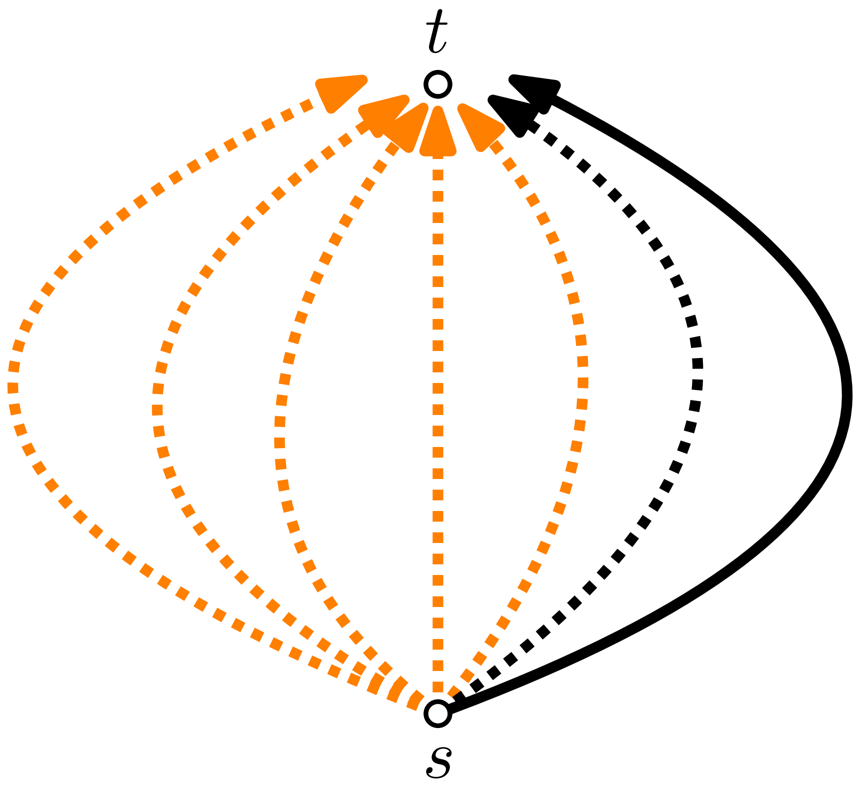
P-Nodes



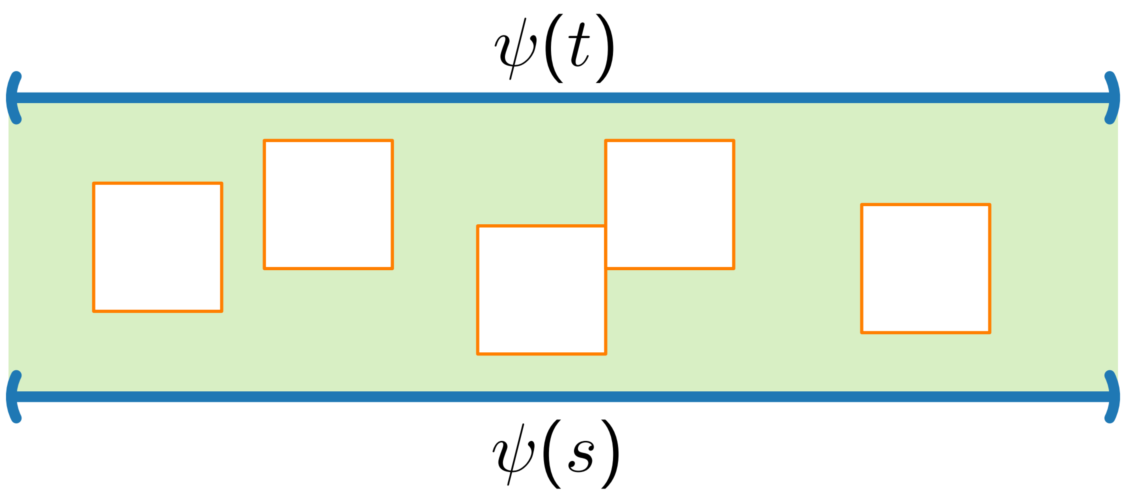


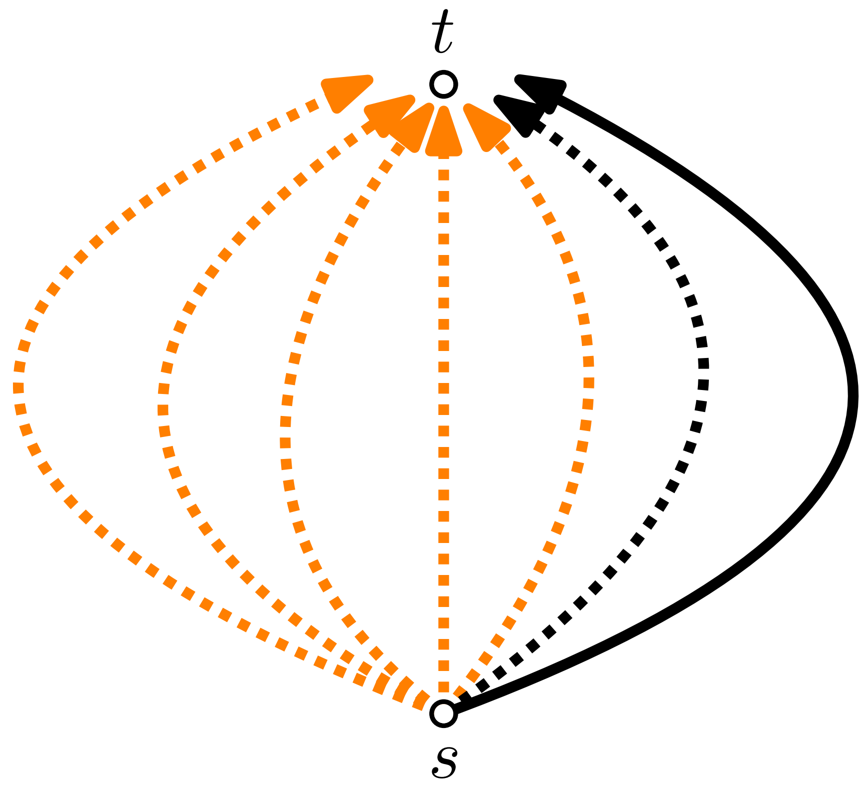
P-Nodes



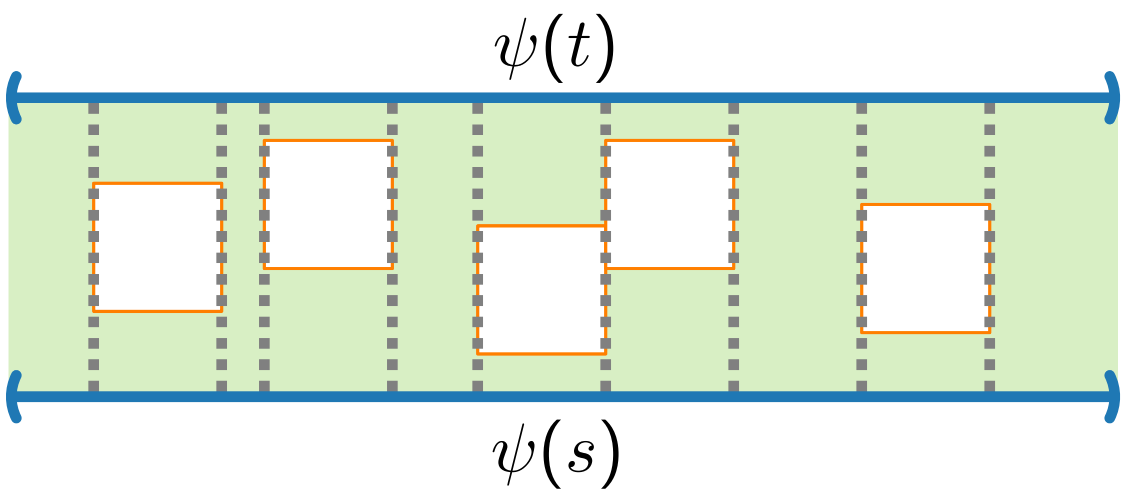


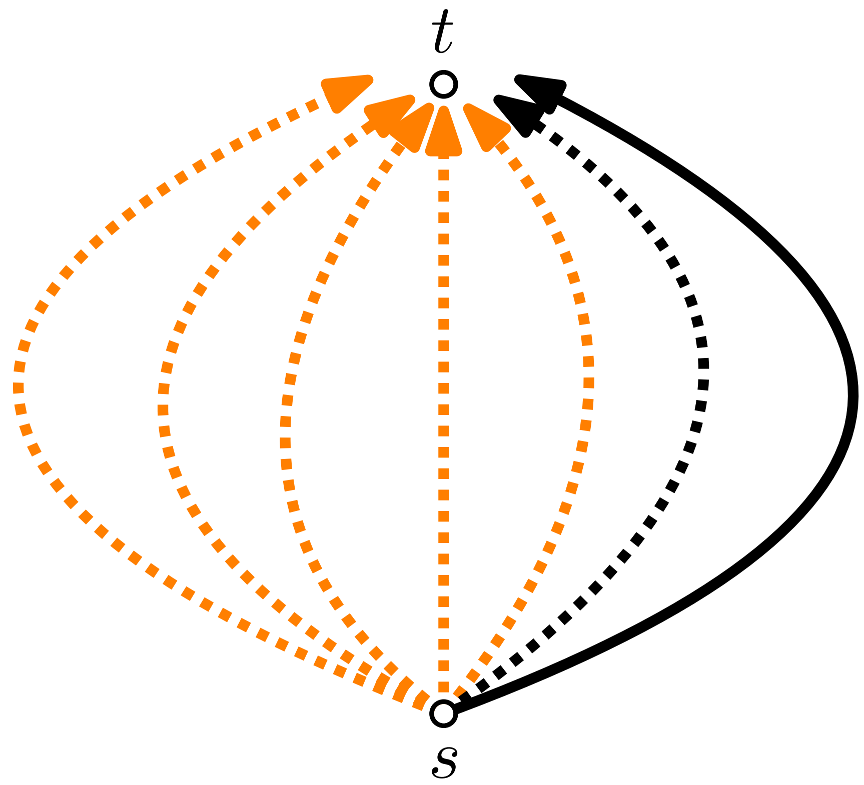
P-Nodes



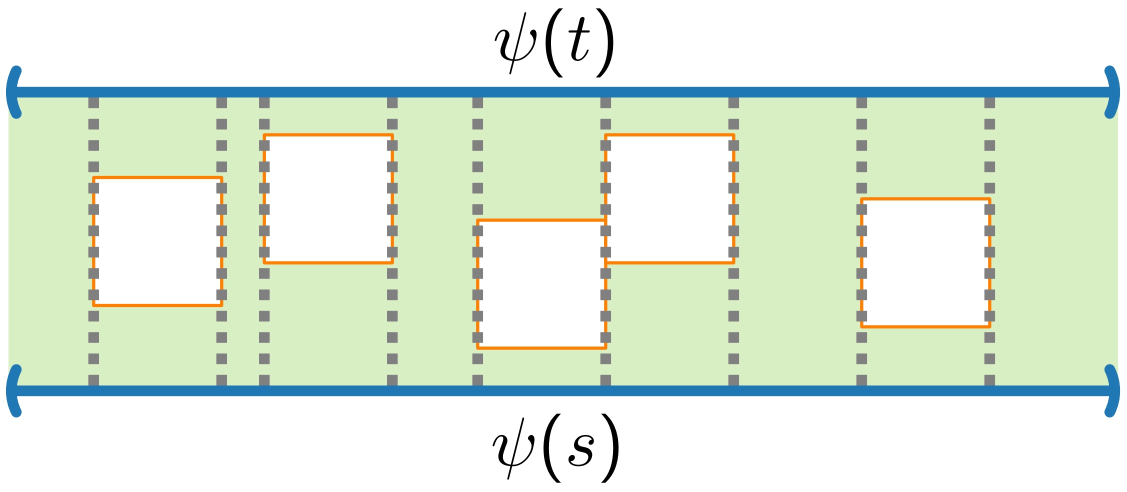


P-Nodes

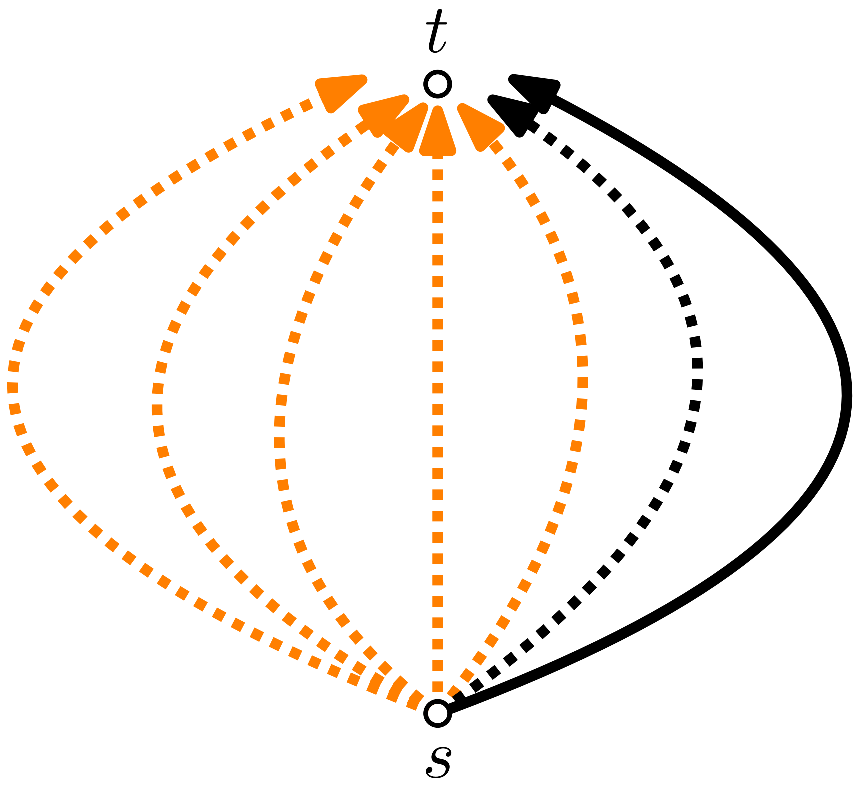




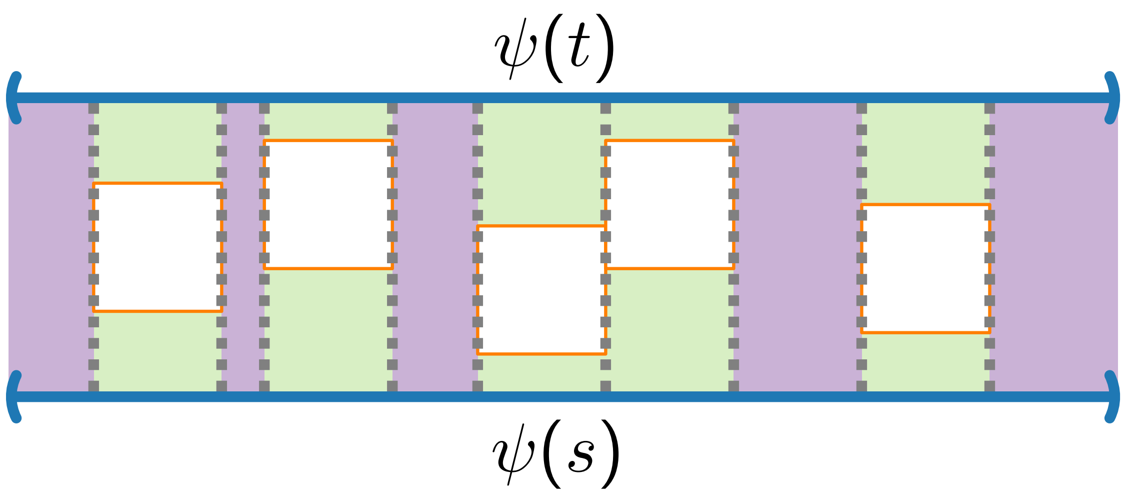
P-Nodes



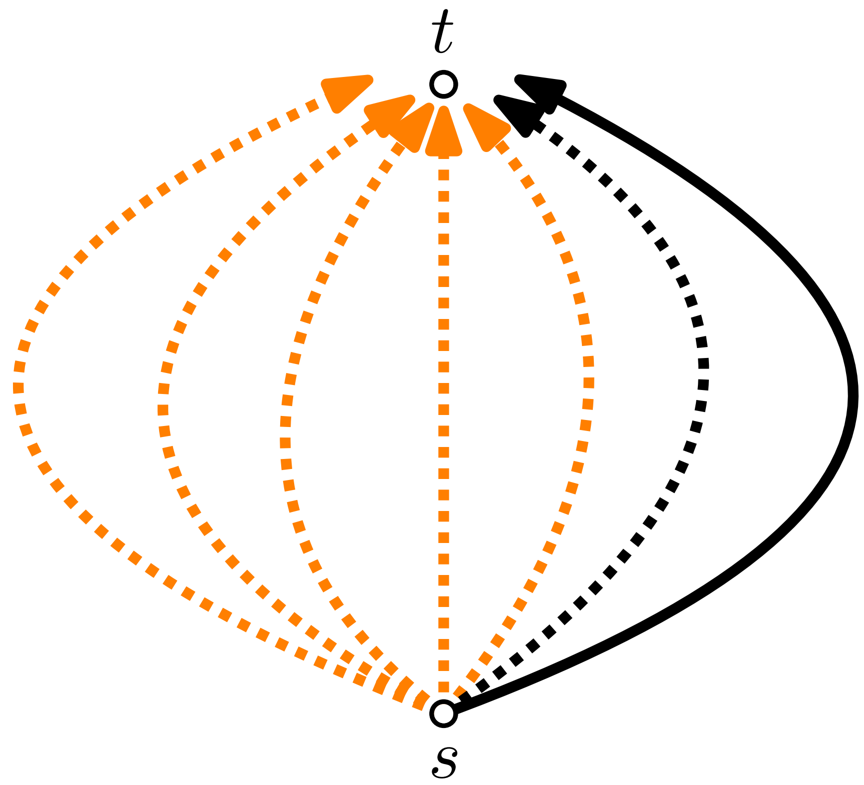
- Children of **P**-node with prescribed bars occur in given left-to-right order



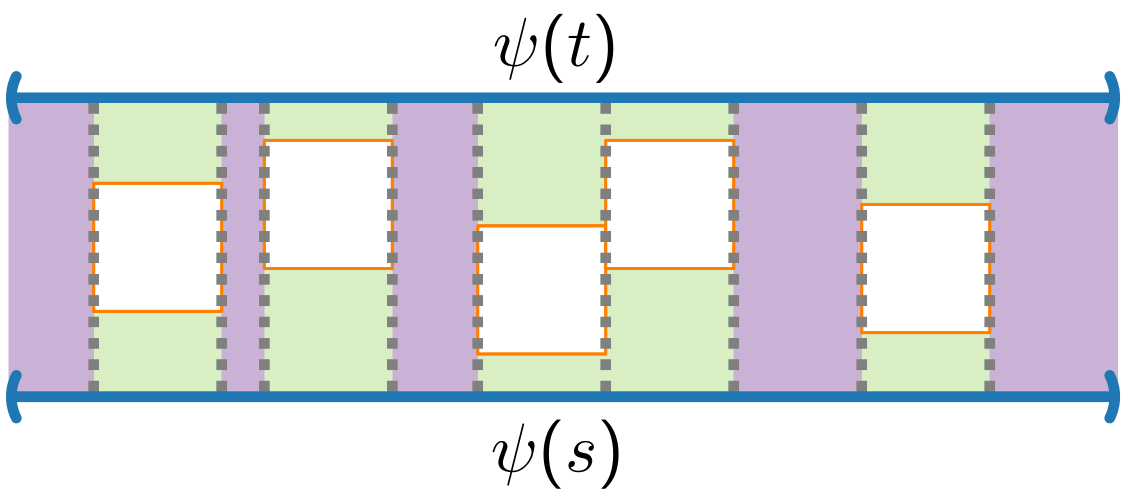
P-Nodes

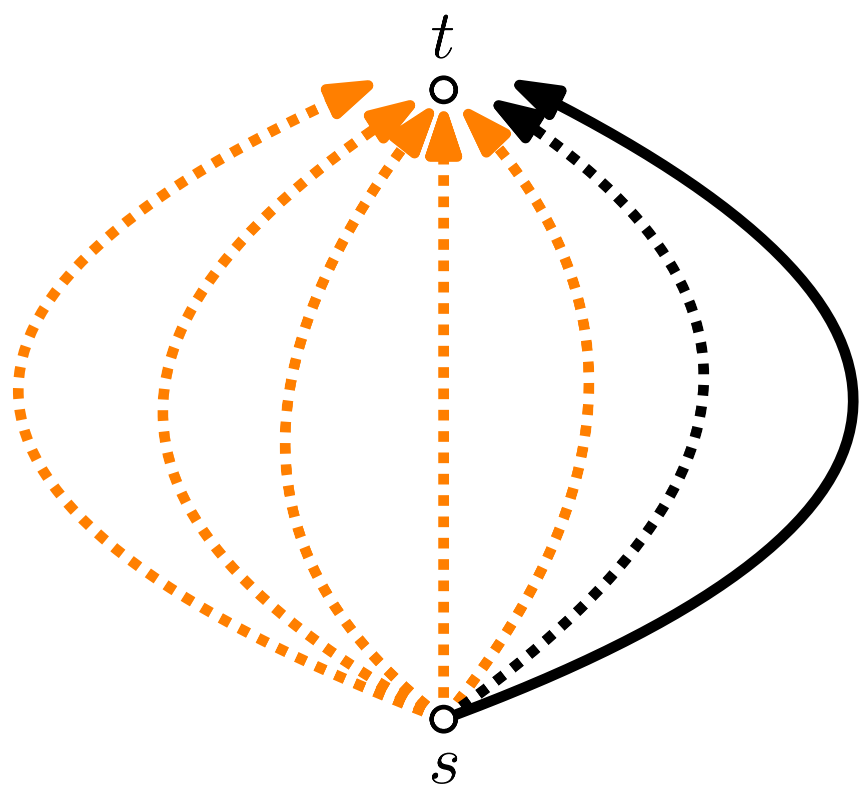


- Children of **P**-node with prescribed bars occur in given left-to-right order
- But there might be some gaps. . .



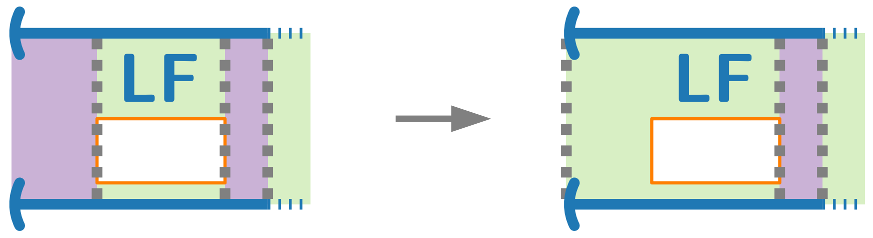
P-Nodes



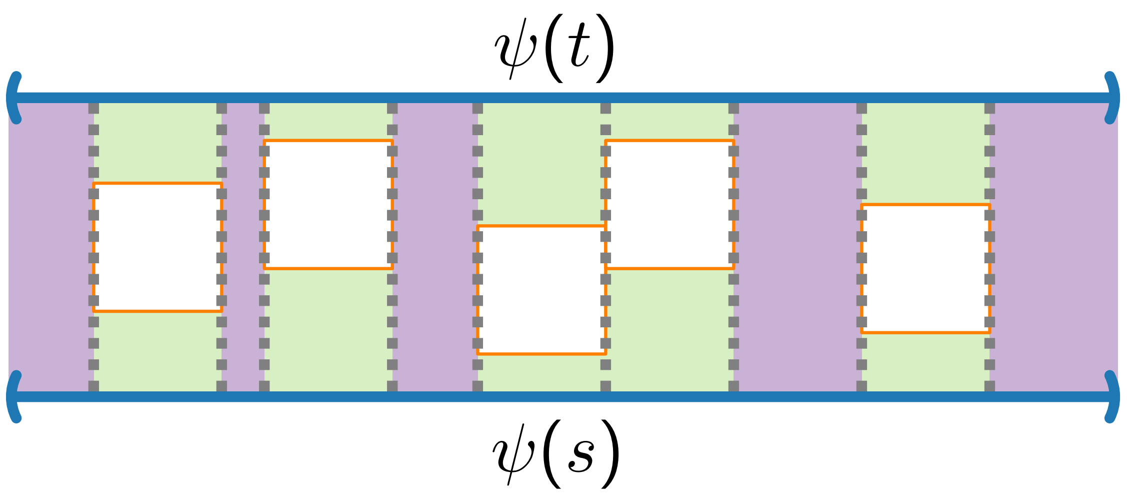


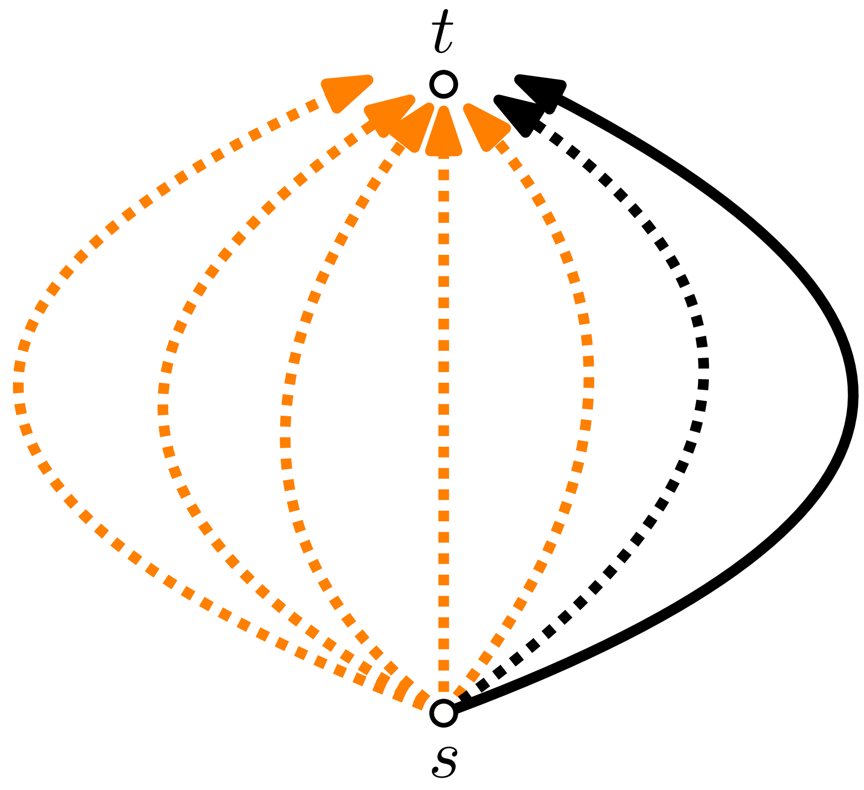
- Children of **P**-node with prescribed bars occur in given left-to-right order
- But there might be some gaps...

Idea.
Greedily *fill* the gaps by preferring to "stretch" the children with prescribed bars.



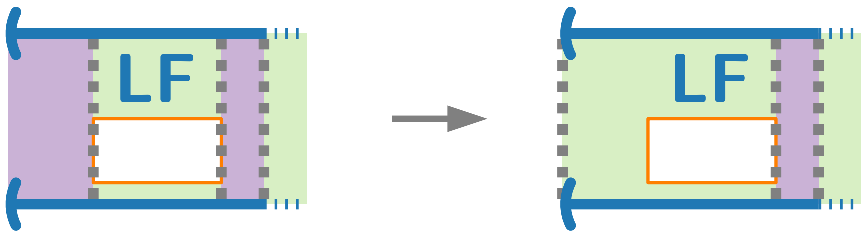
P-Nodes





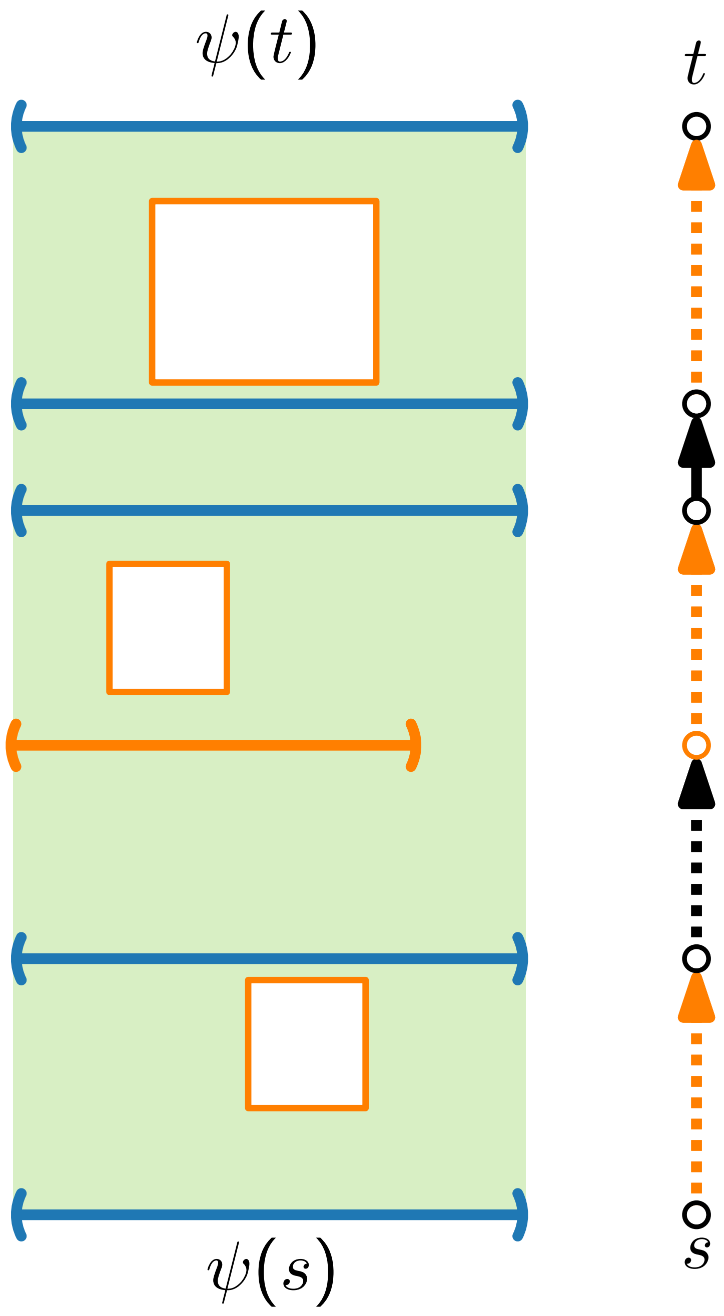
- Children of **P**-node with prescribed bars occur in given left-to-right order
- But there might be some gaps...

Idea.
Greedily *fill* the gaps by preferring to "stretch" the children with prescribed bars.

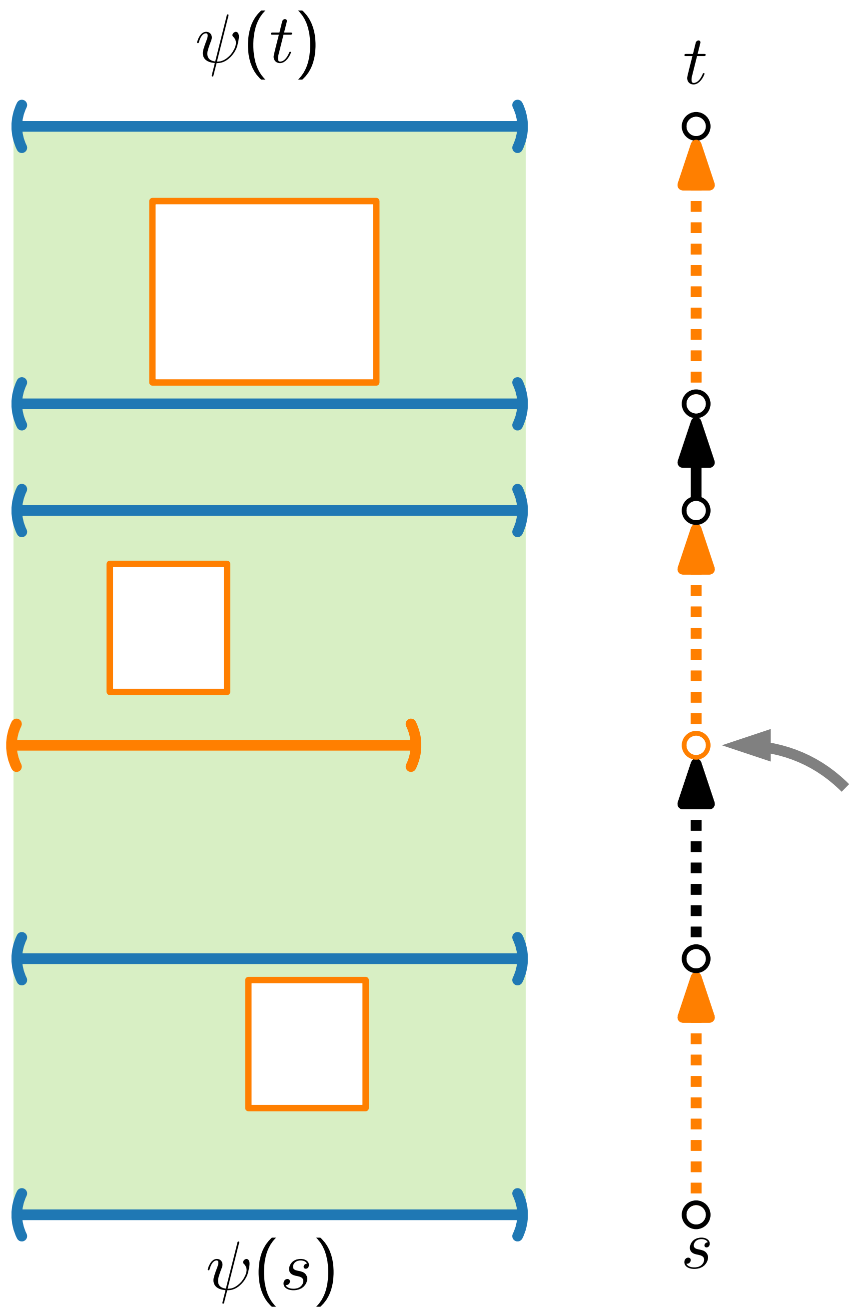


Outcome.
After processing, we must know the valid types for the corresponding subgraphs.

S-Nodes

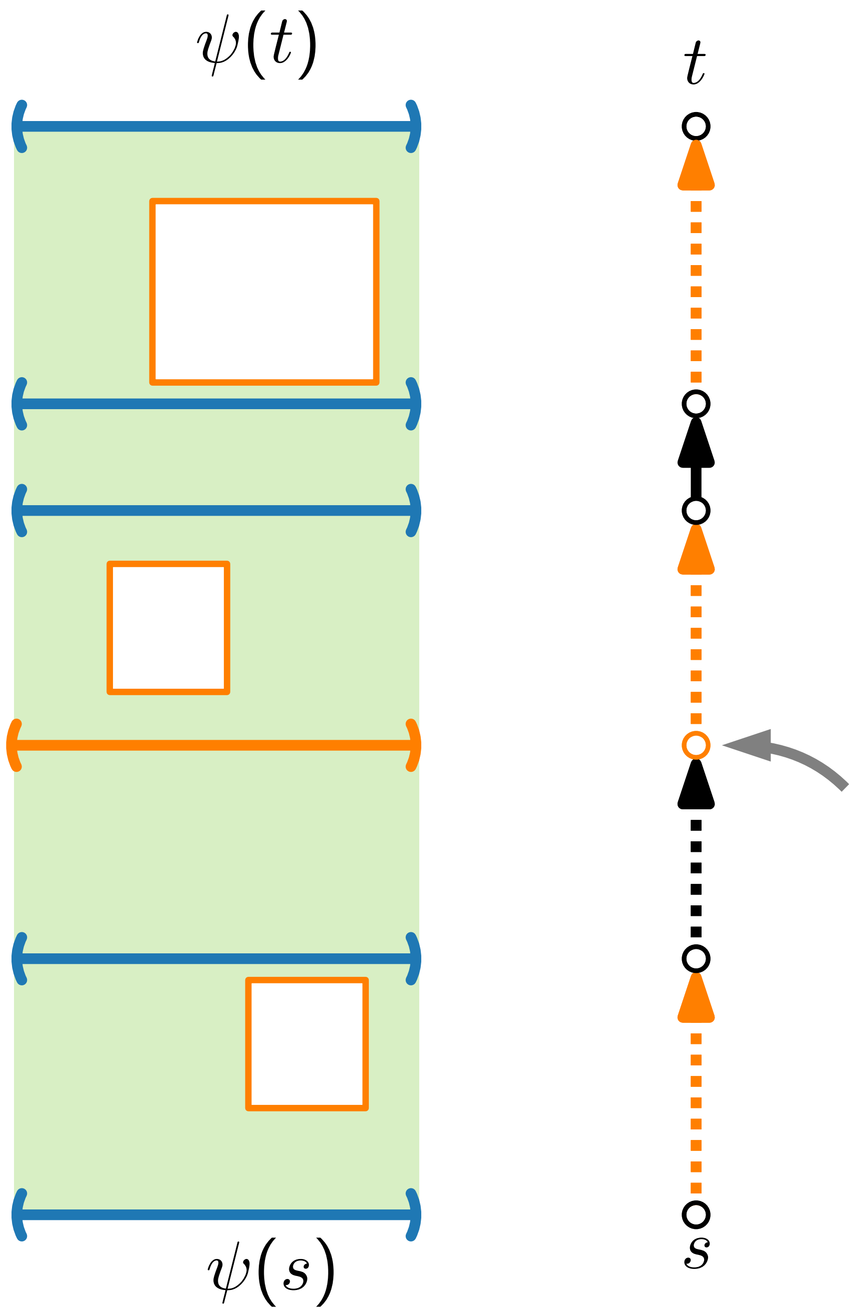


S-Nodes



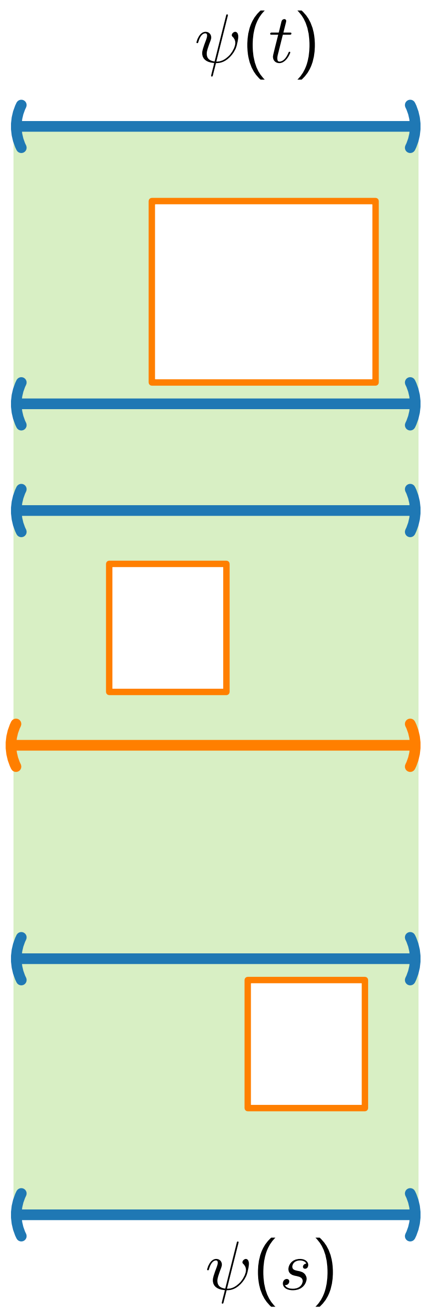
This fixed vertex means we can only make a Fixed-Fixed representation!

S-Nodes

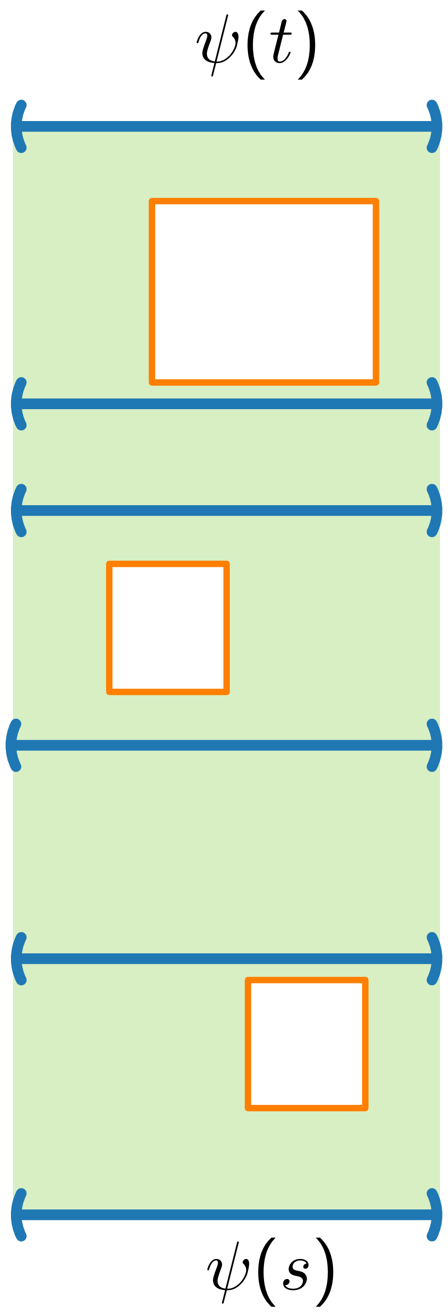


This fixed vertex means we can only make a **F**ixed-**F**ixed representation!

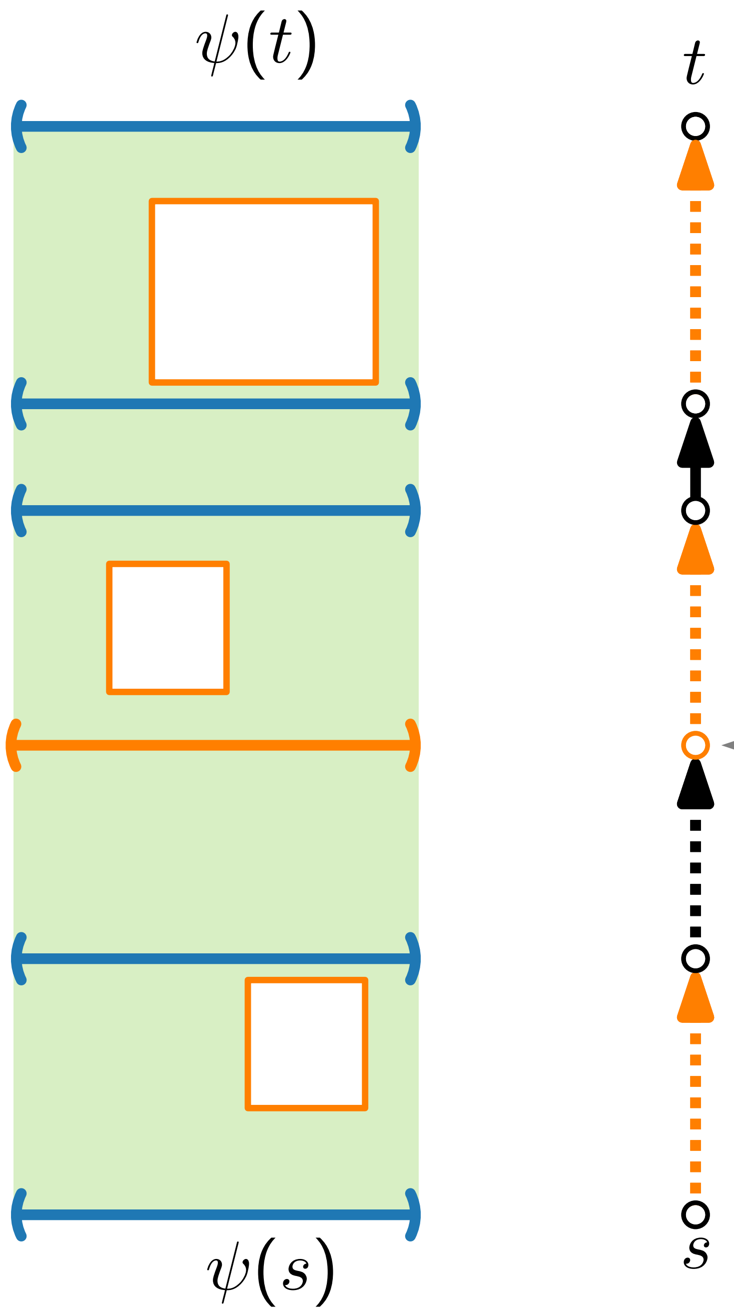
S-Nodes



This fixed vertex means we can only make a **F**ixed-**F**ixed representation!

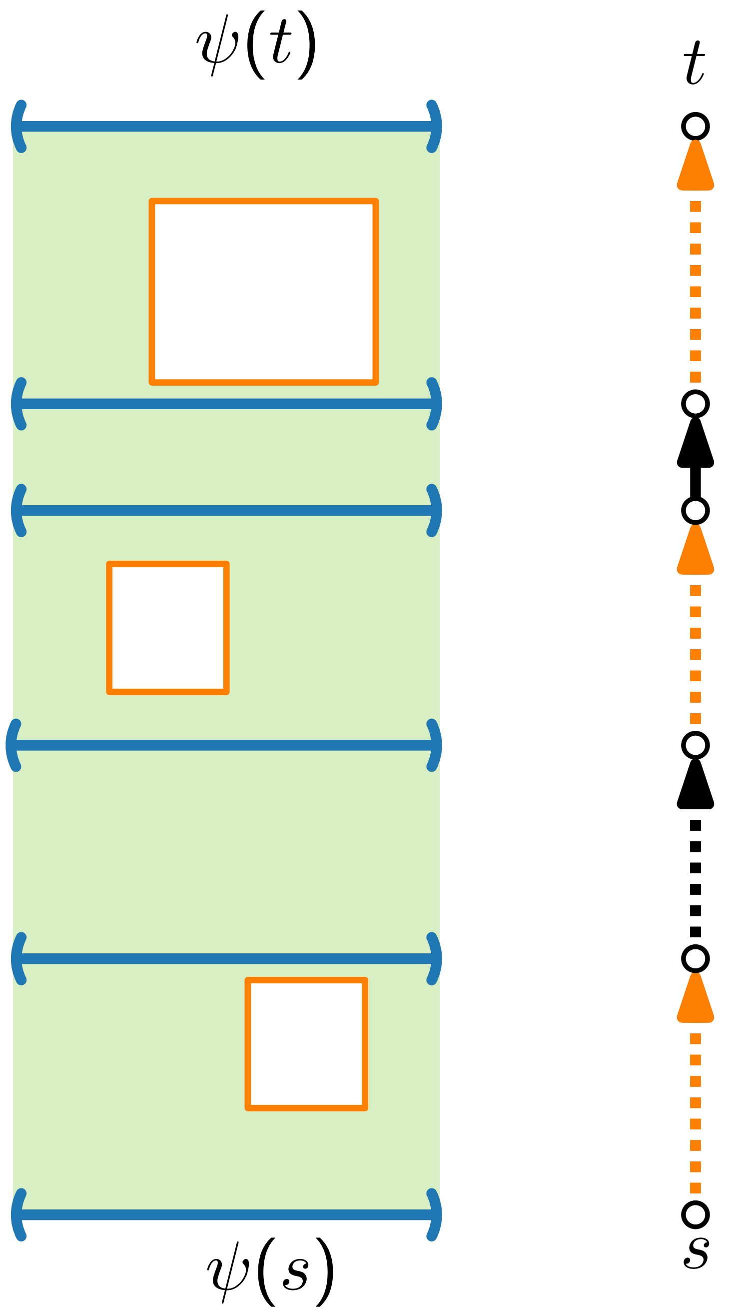


S-Nodes

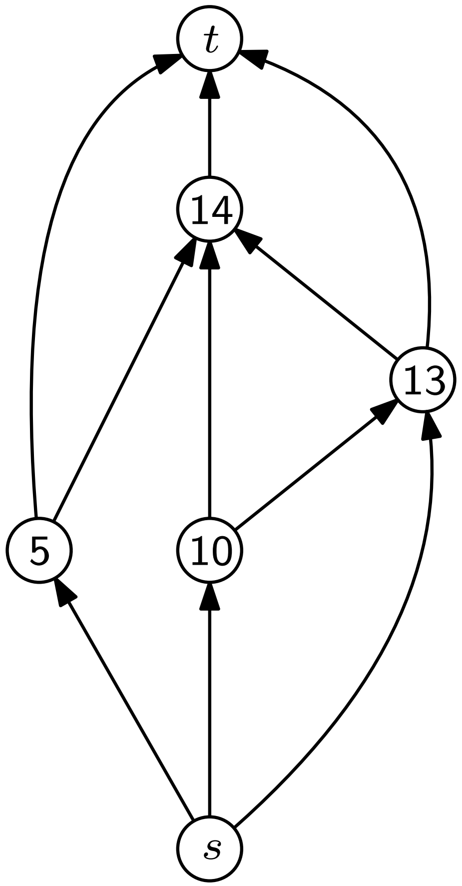


Here we have a chance to make all (**LL, FL, LF, FF**) types.

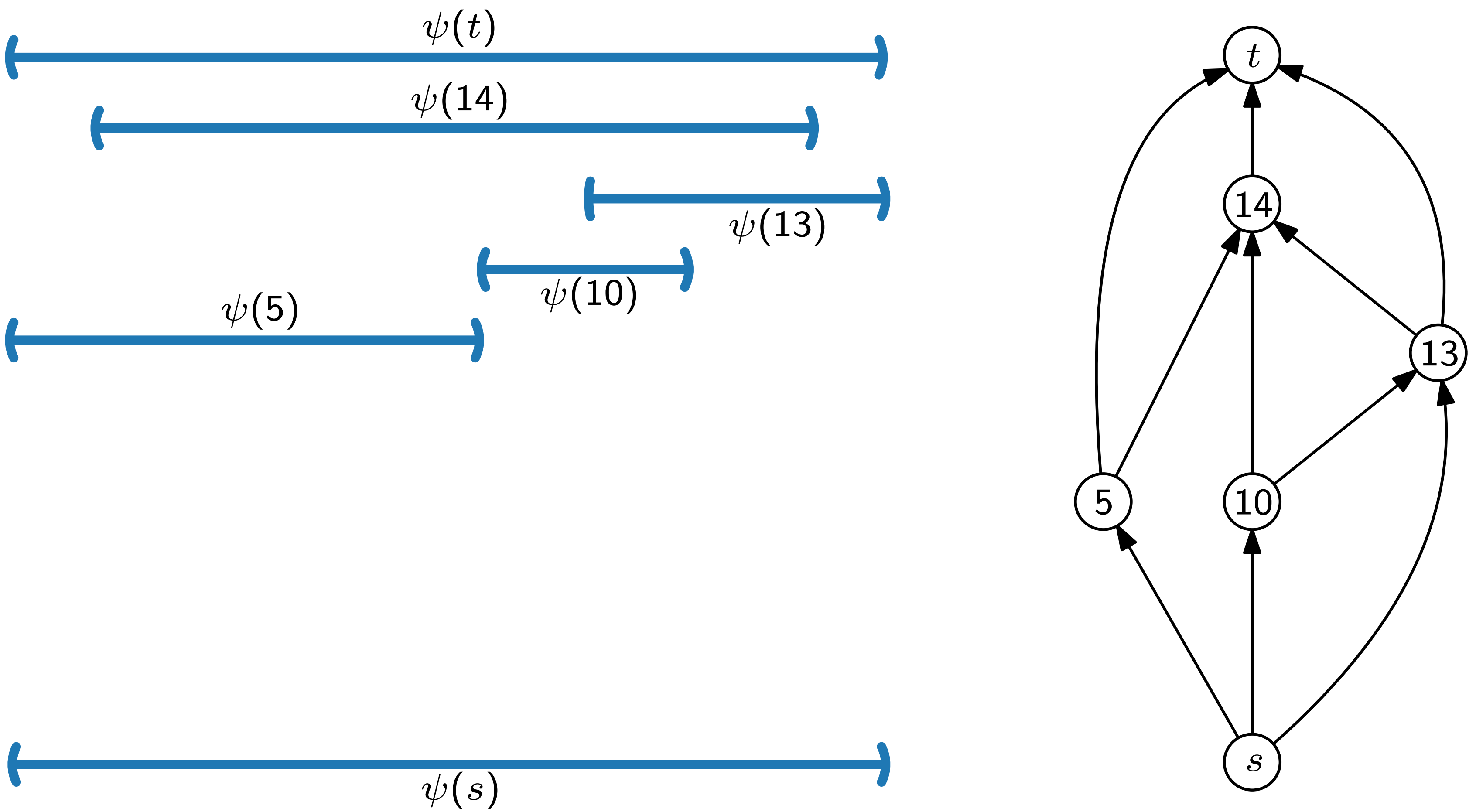
This fixed vertex means we can only make a **F**ixed-**F**ixed representation!



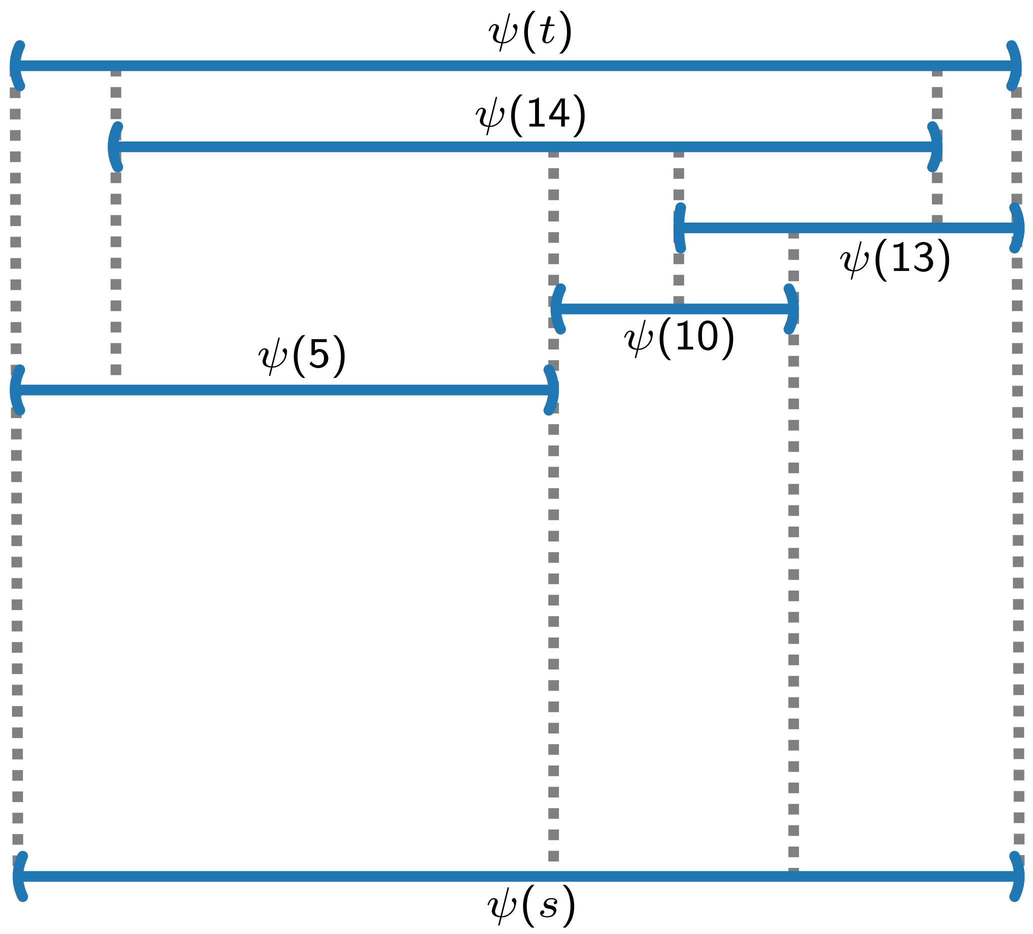
R-Nodes

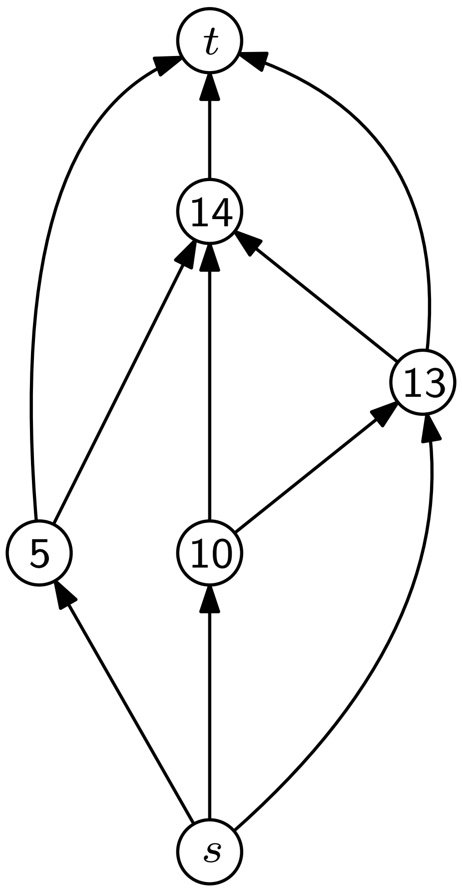


R-Nodes

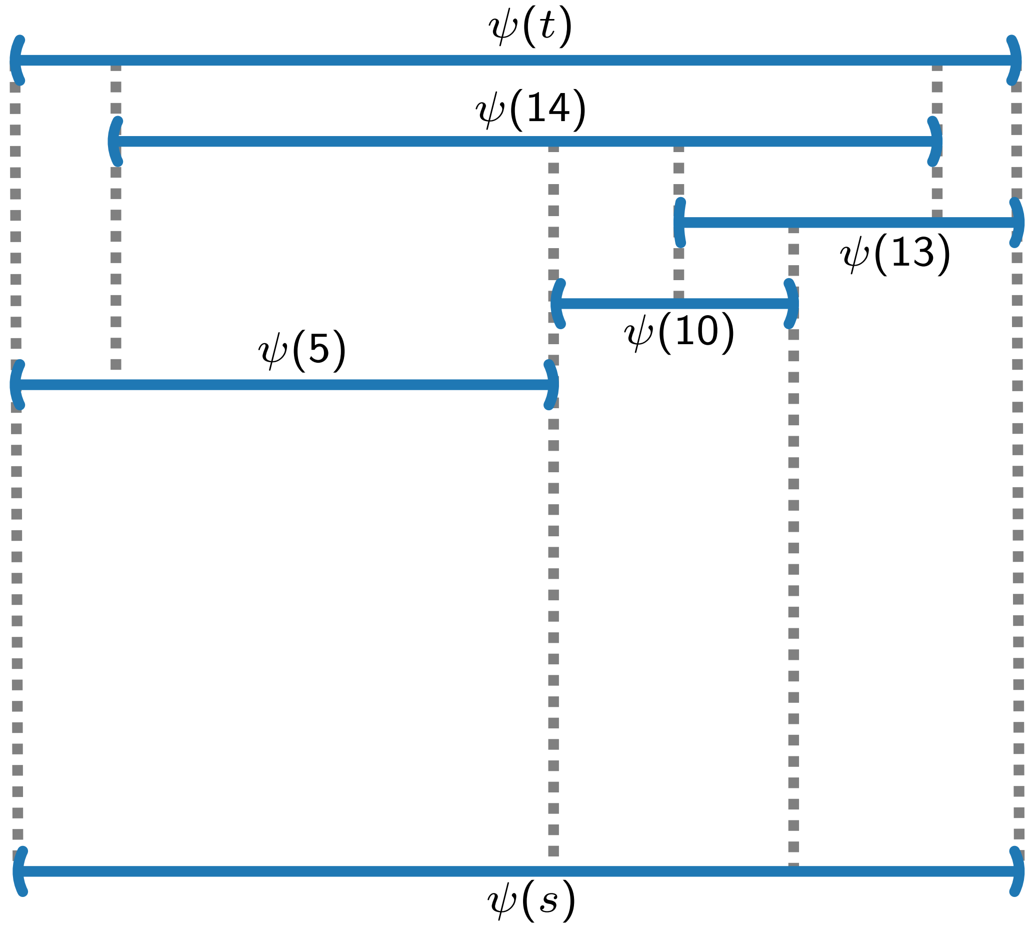


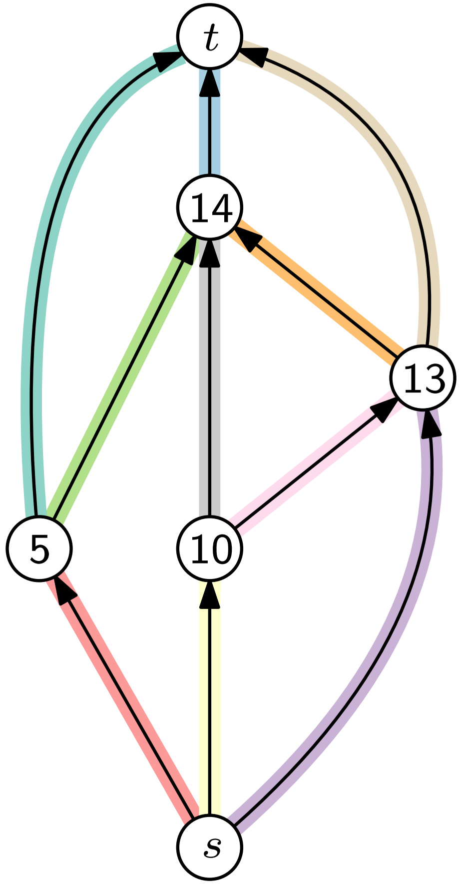
R-Nodes



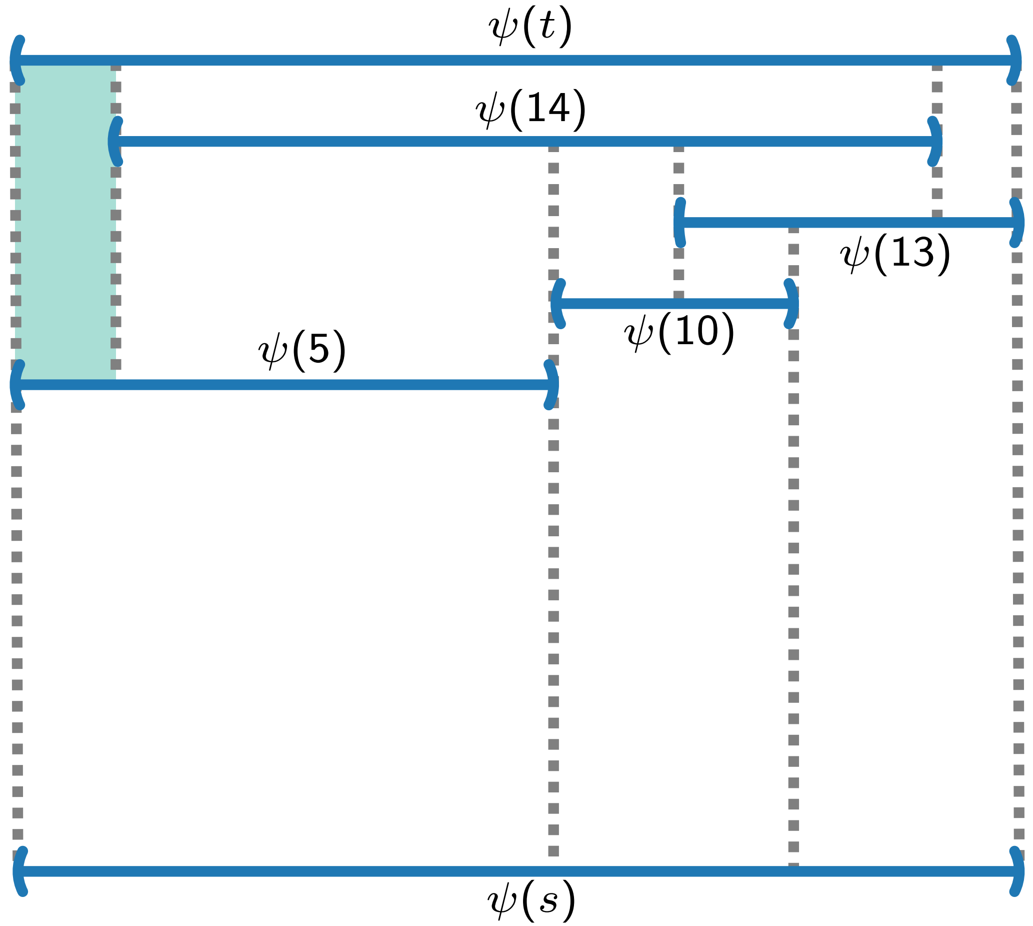


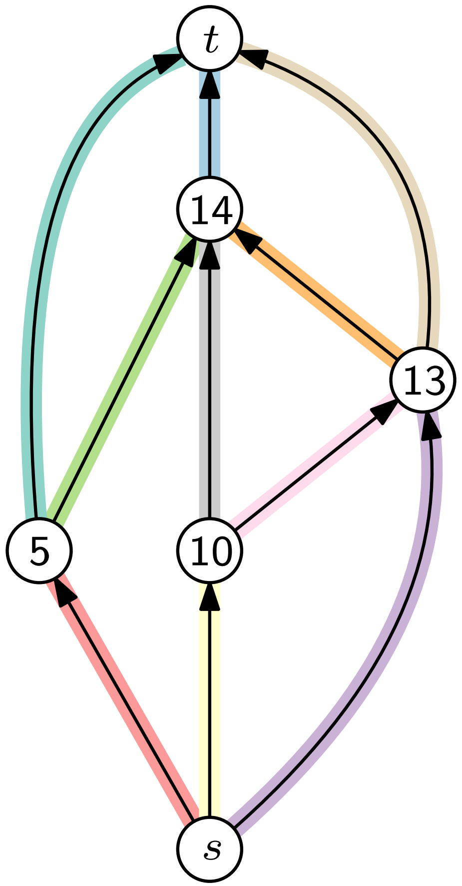
R-Nodes



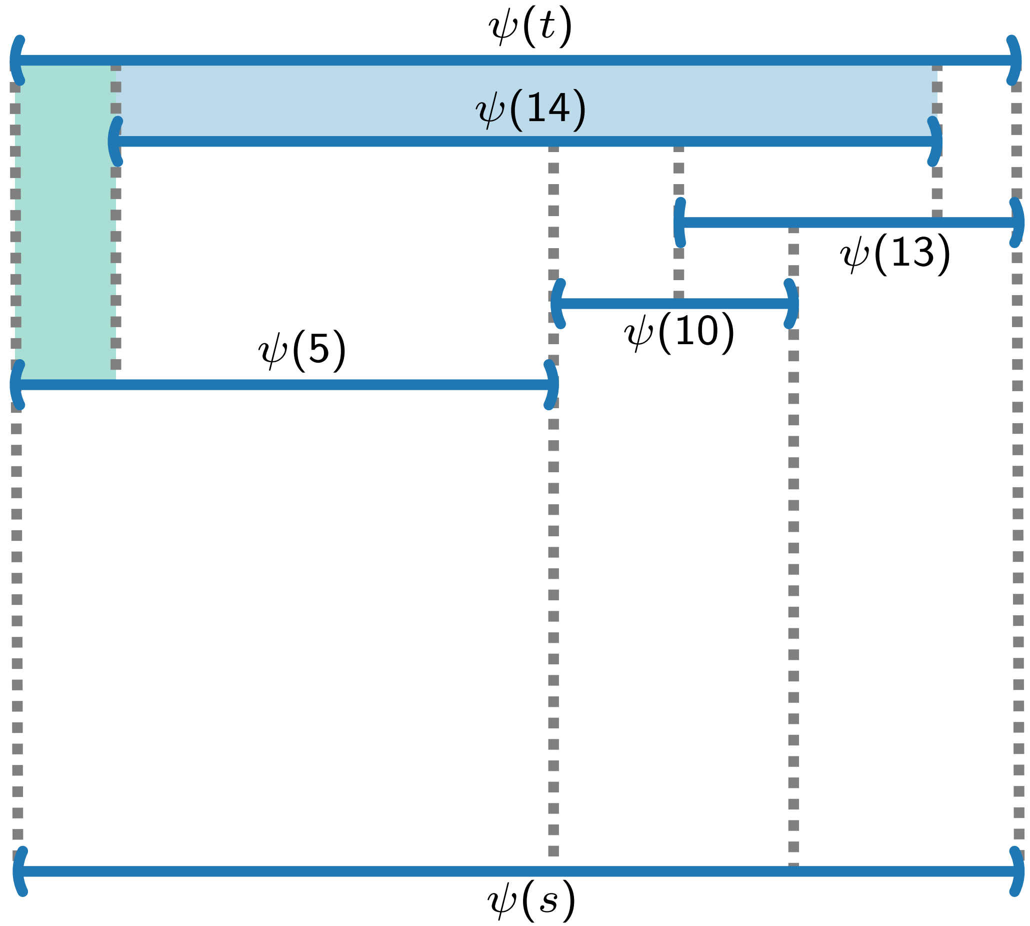


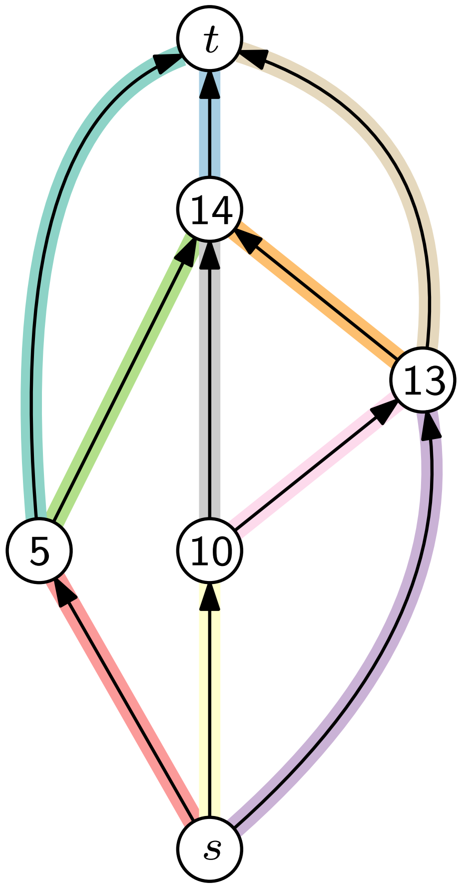
R-Nodes



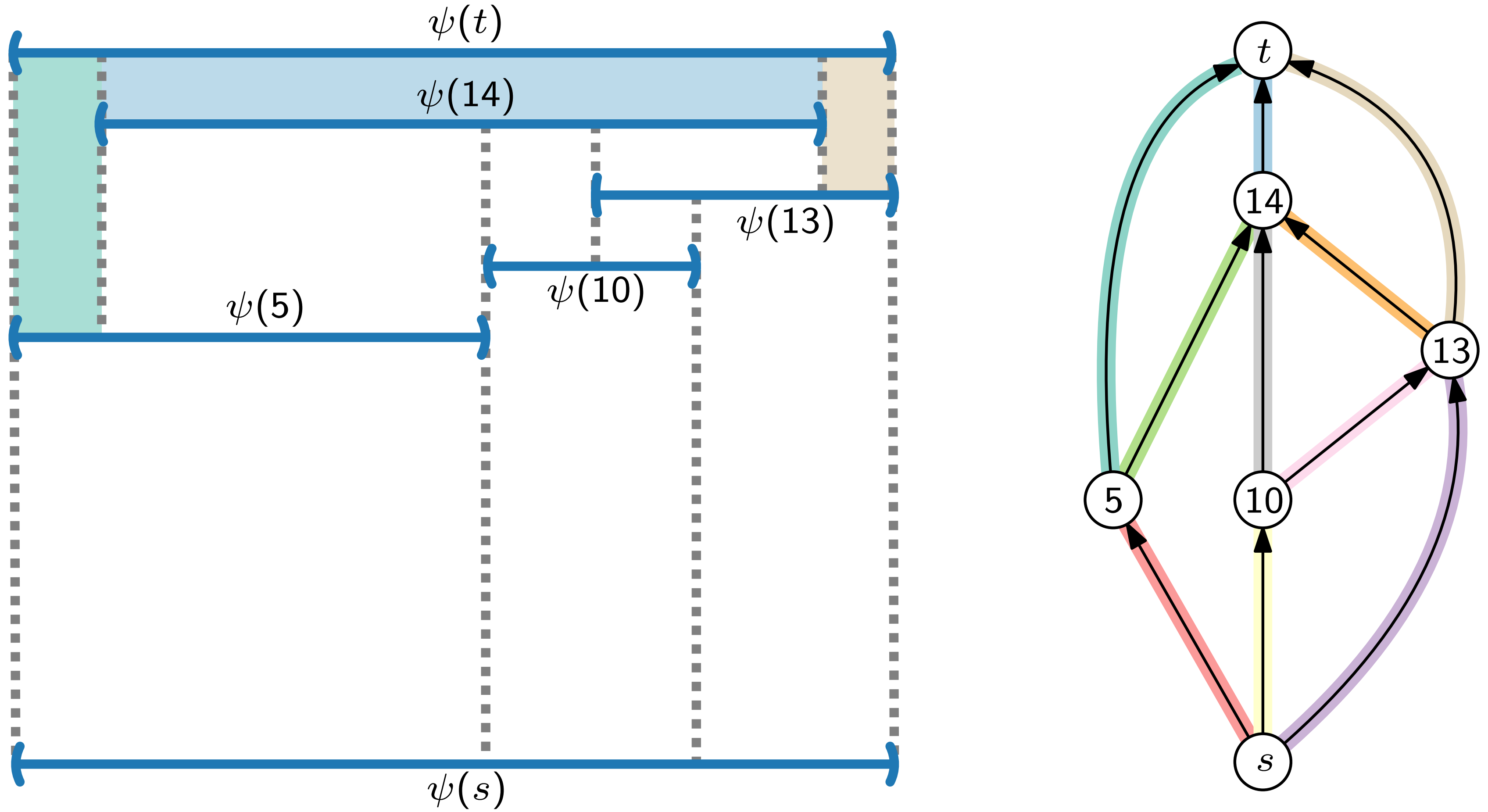


R-Nodes

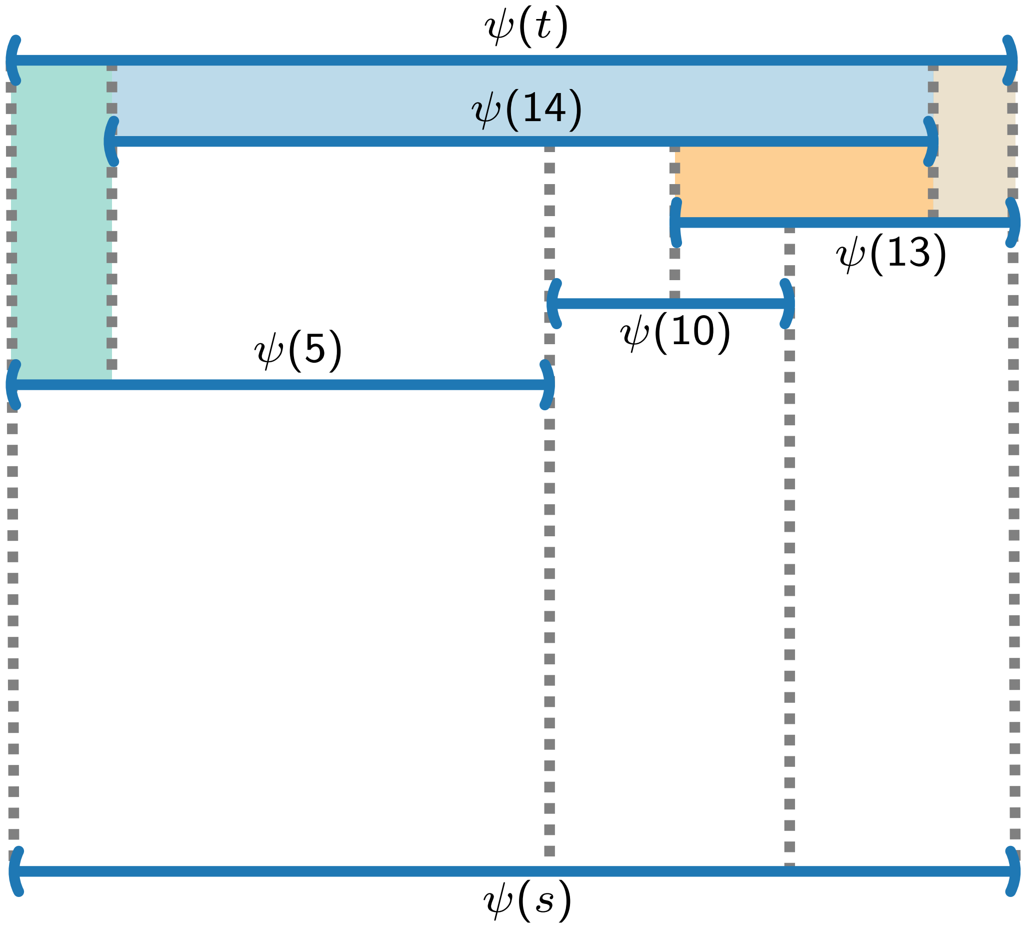


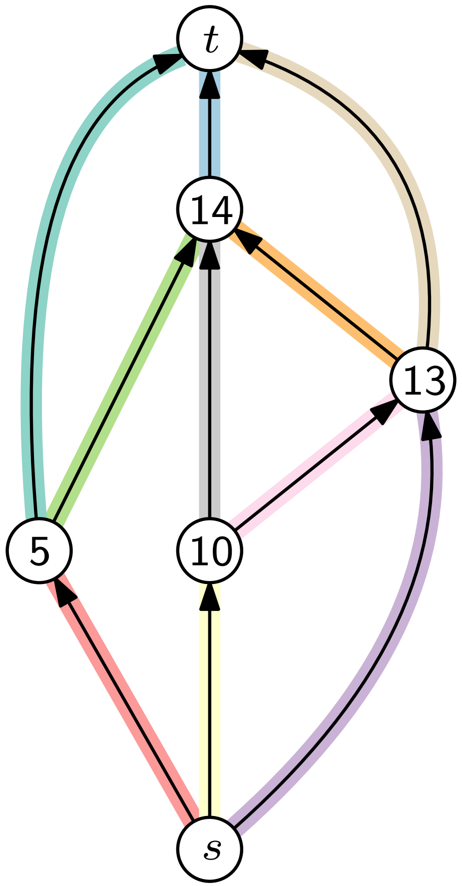


R-Nodes

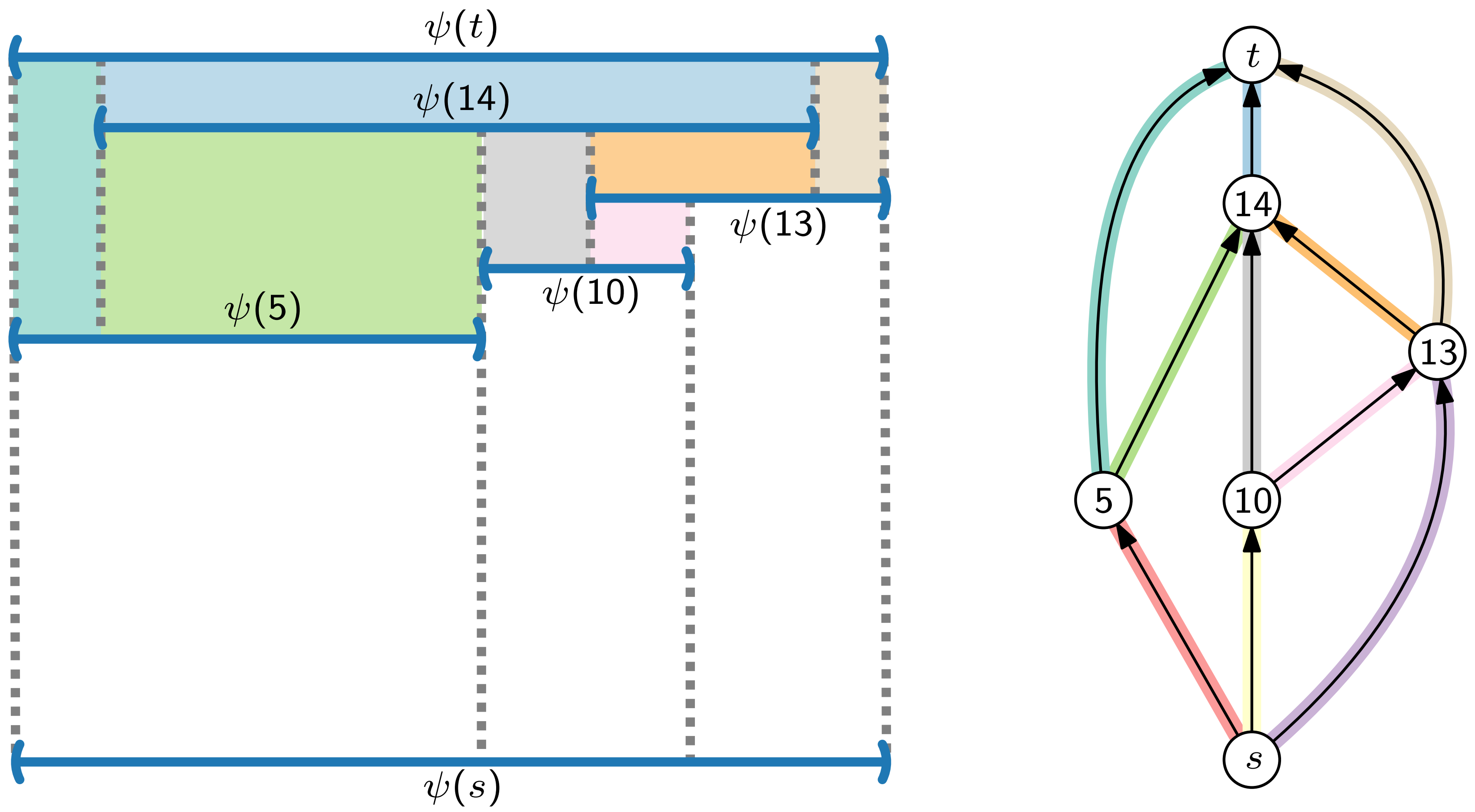


R-Nodes

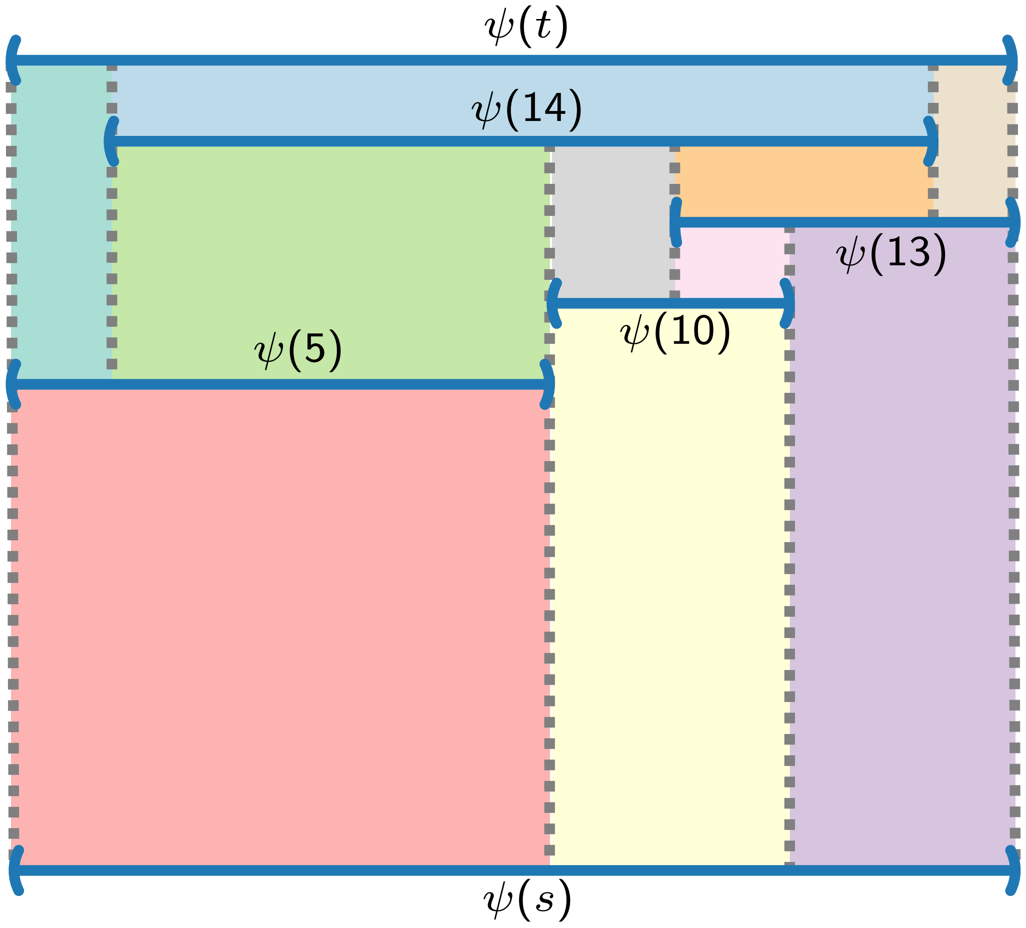


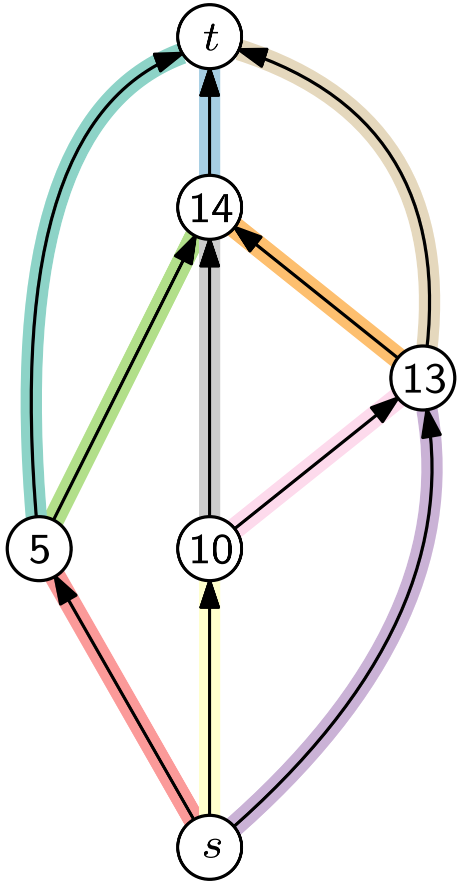


R-Nodes

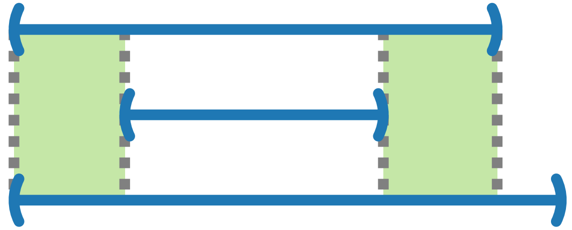


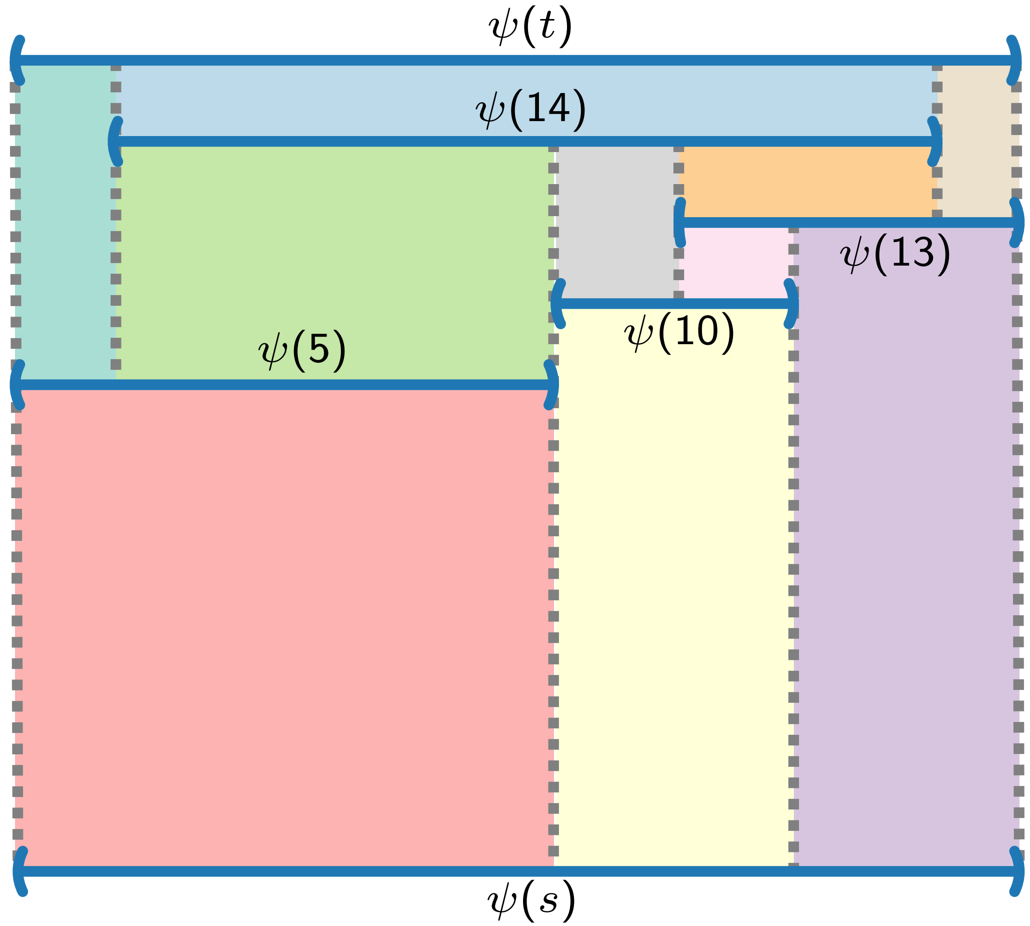
R-Nodes

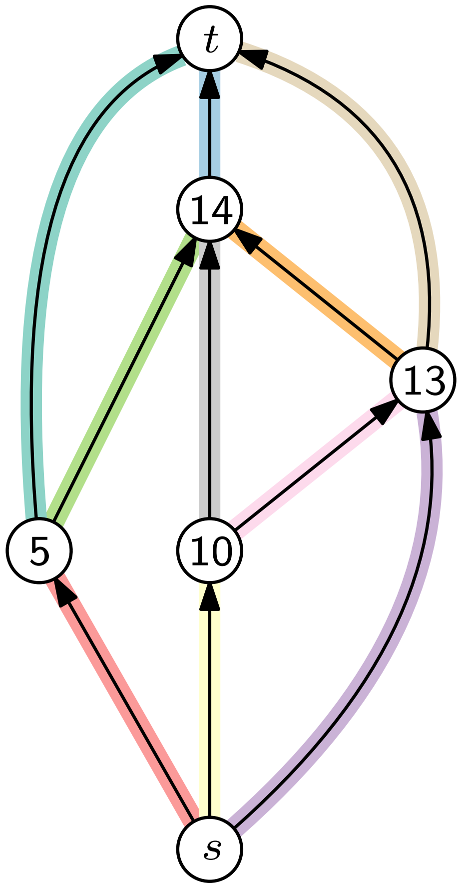




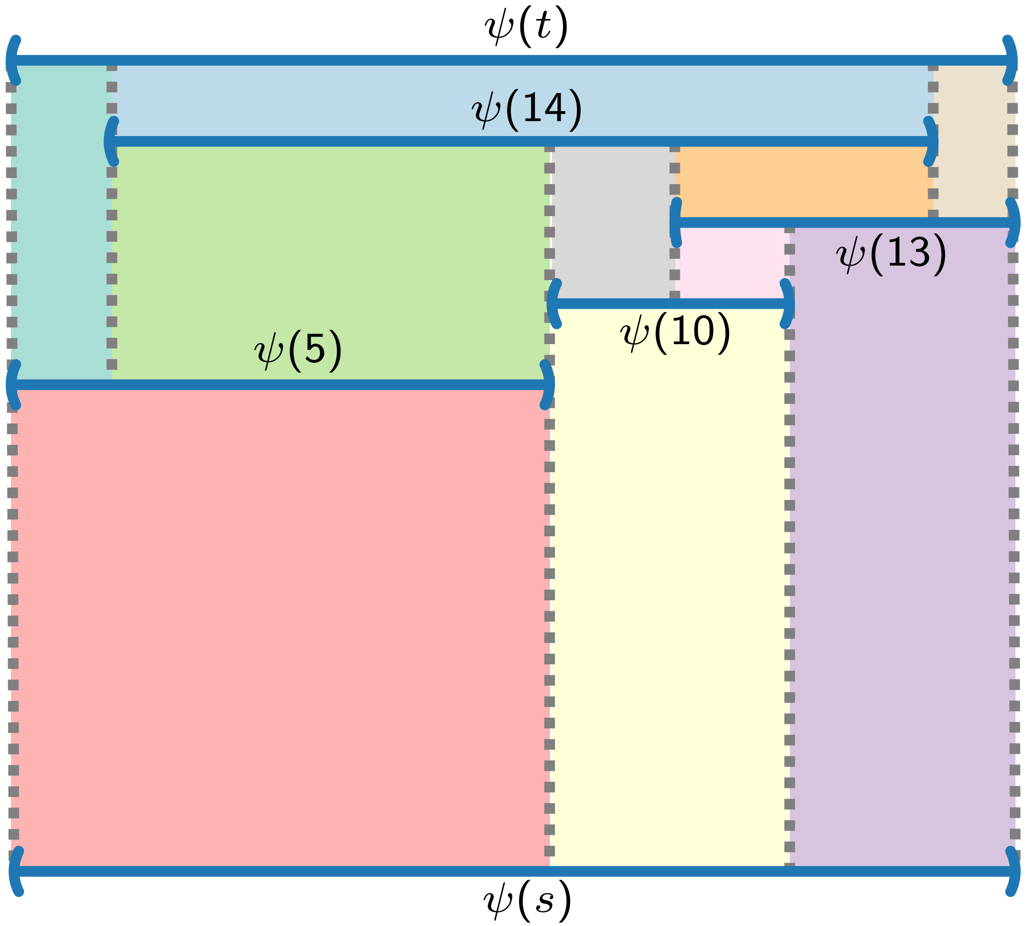
R-Nodes

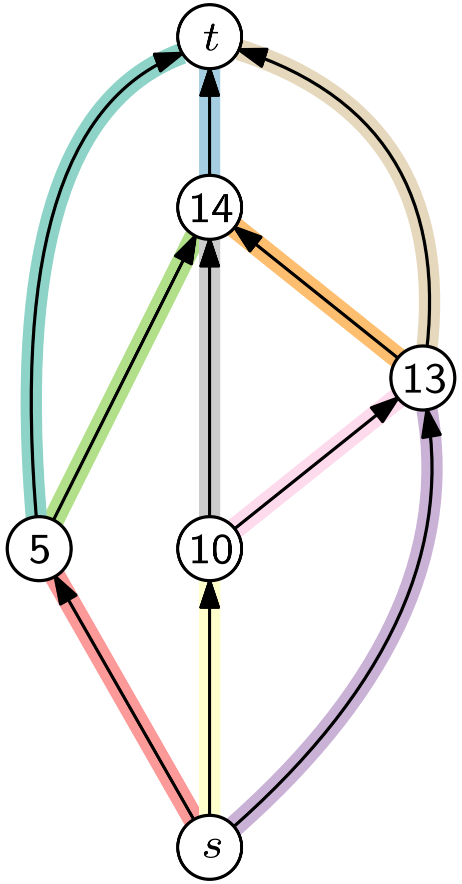


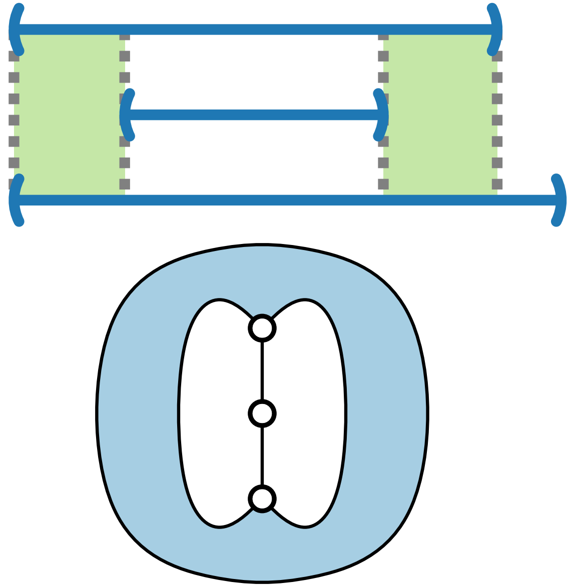




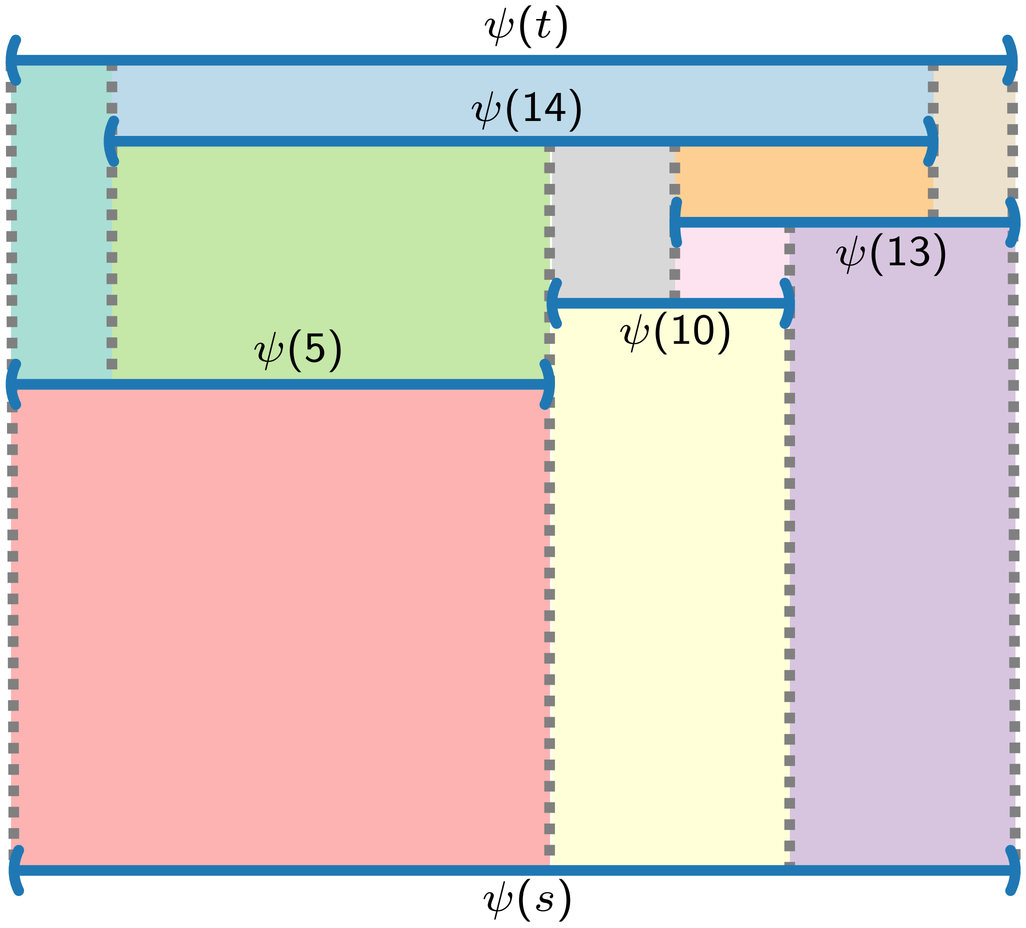
R-Nodes



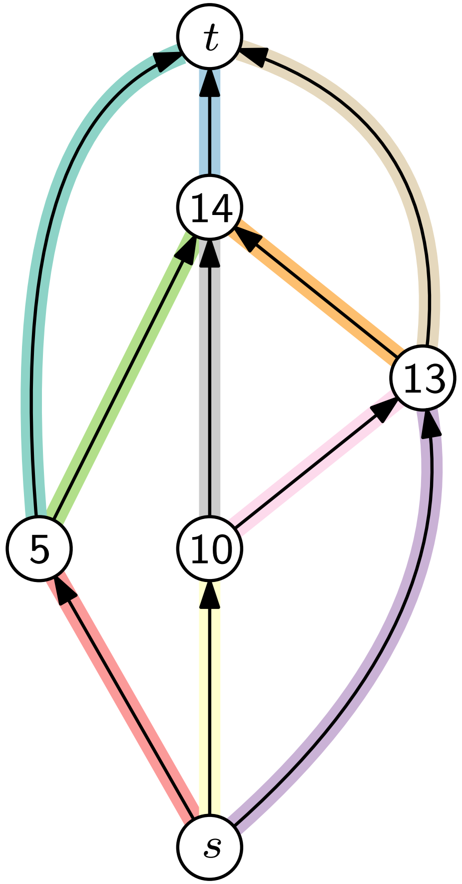




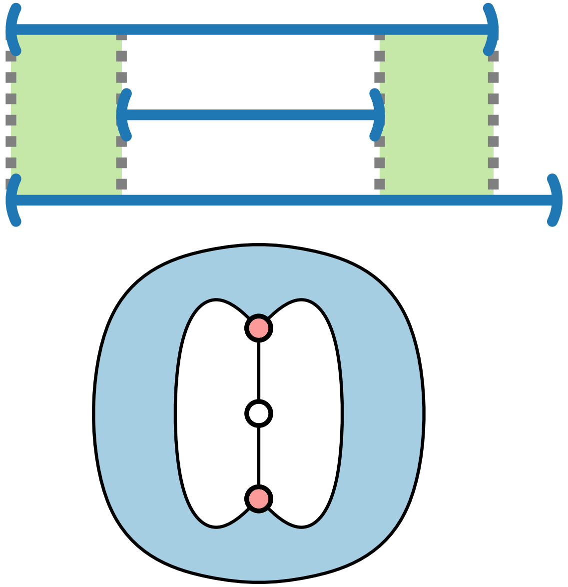
R-Nodes



$\psi(t)$
$\psi(14)$
$\psi(13)$
$\psi(10)$
$\psi(5)$
$\psi(s)$

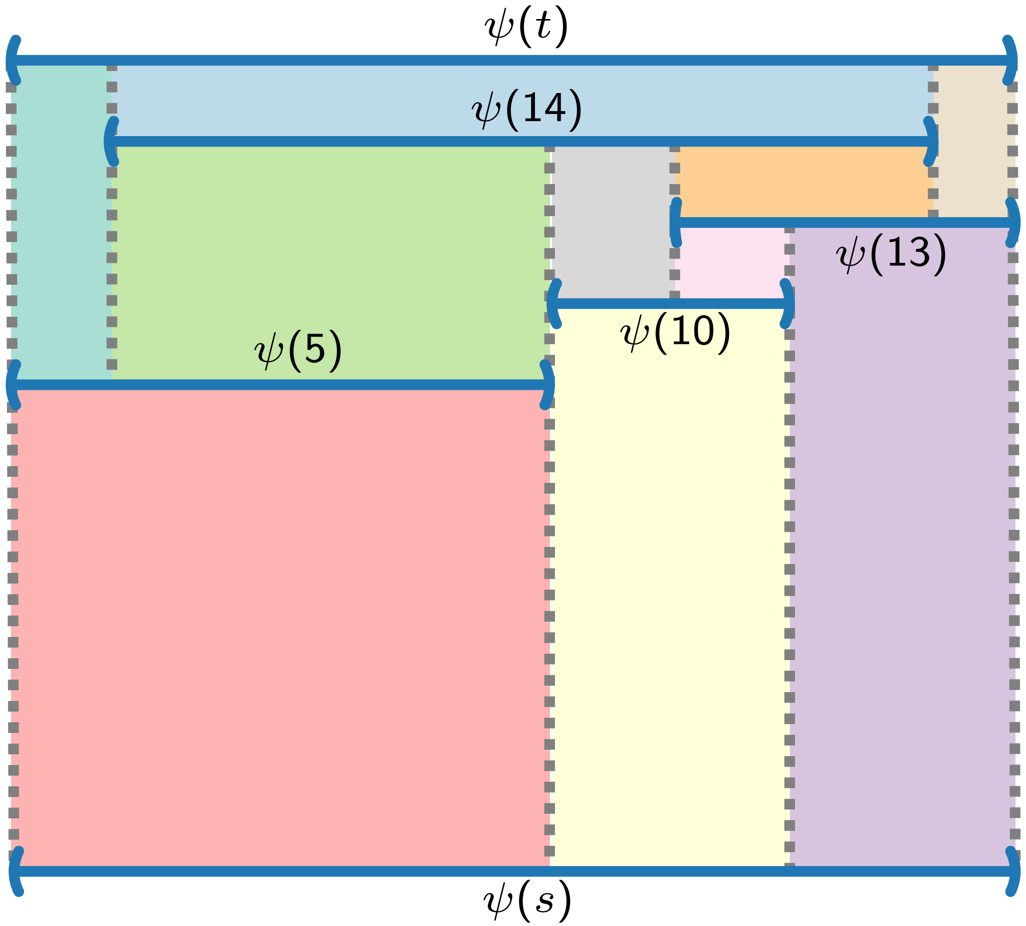


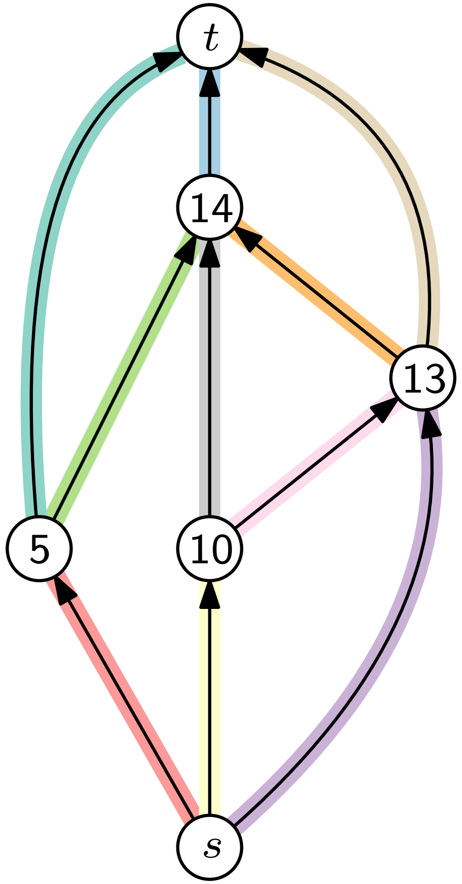
t
14
13
5
10
s

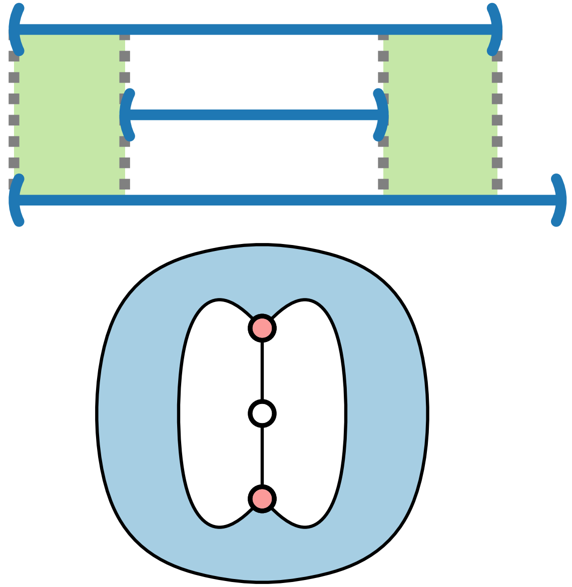


Separation pair!

R-Nodes



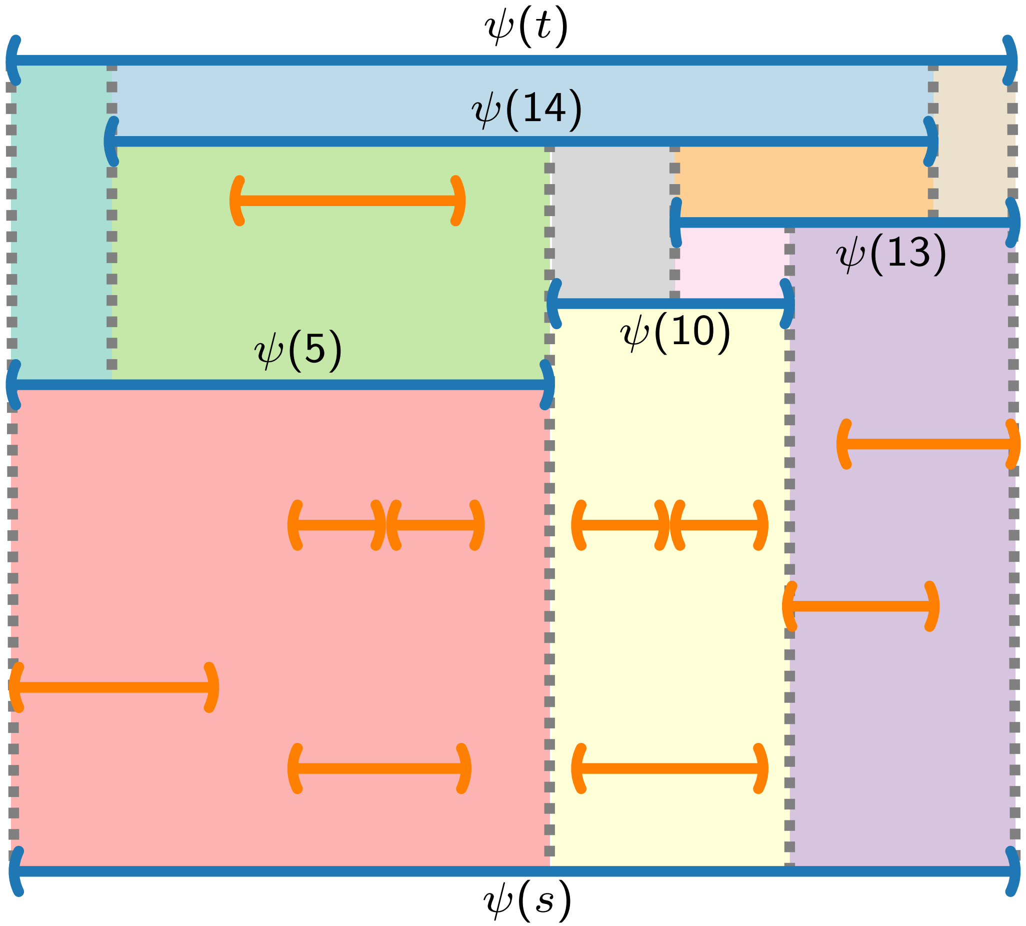


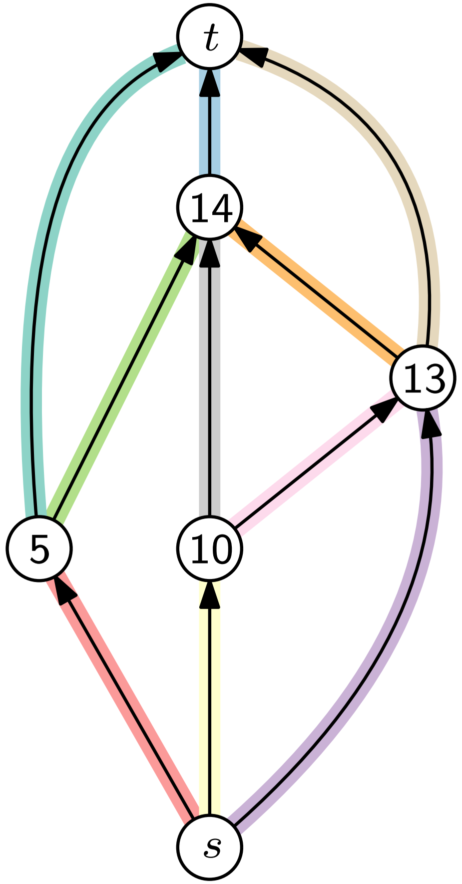


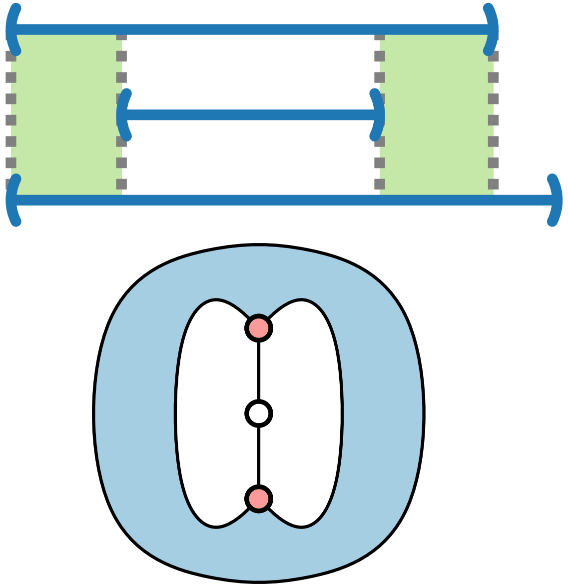
Separation pair!

($\nexists$ in **R**-component.)

R-Nodes





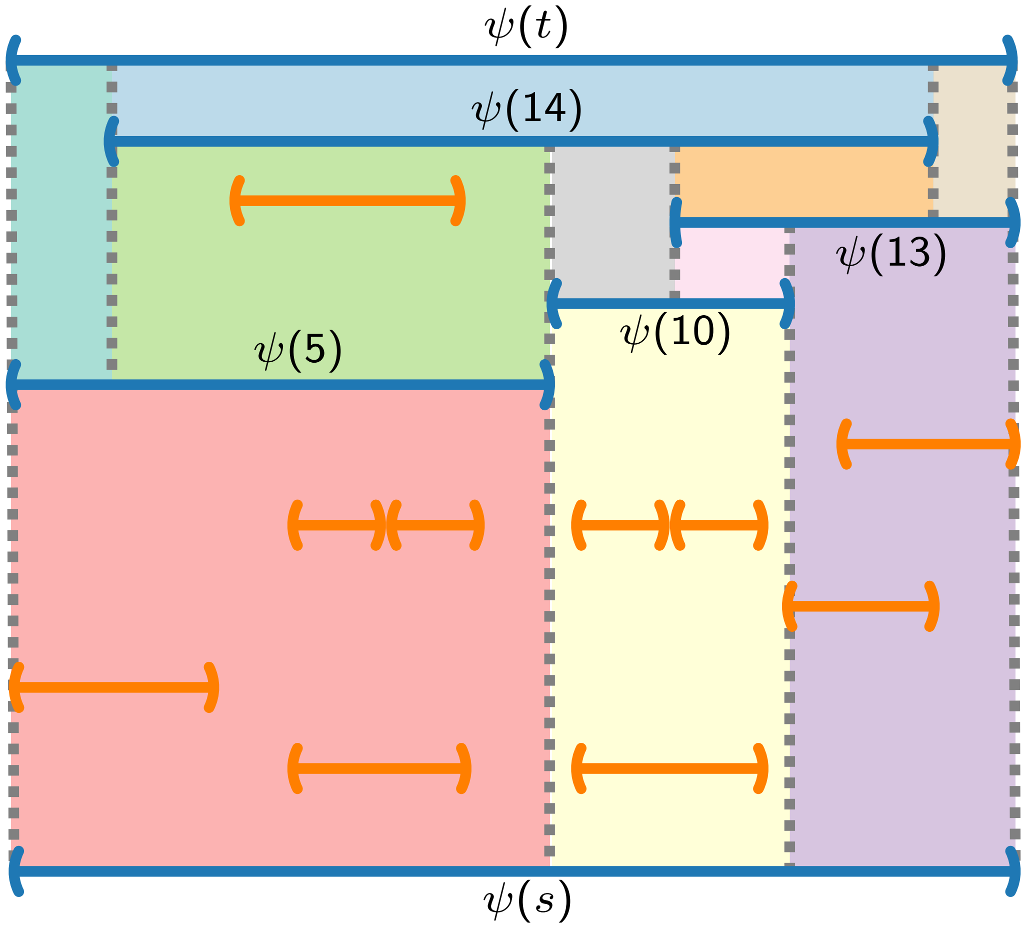


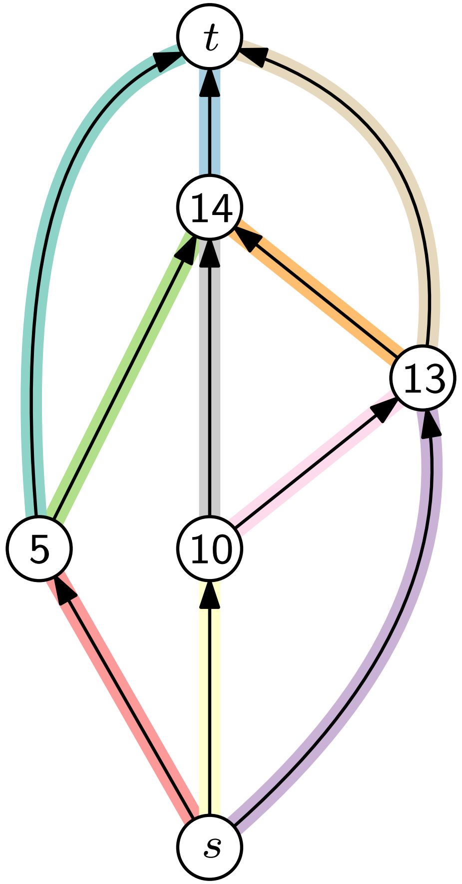
Separation pair!

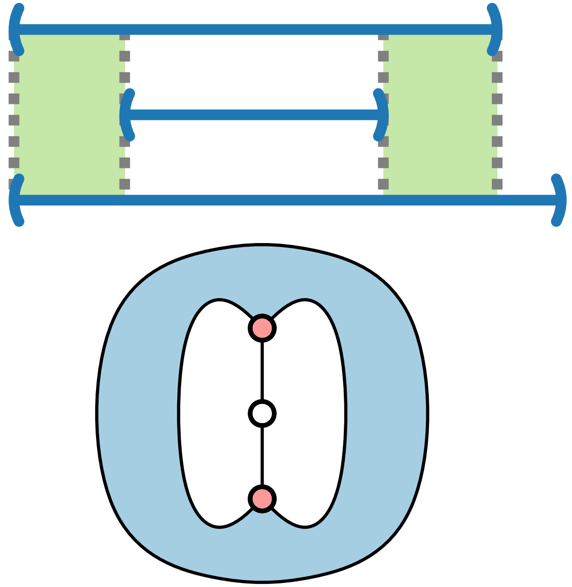
($\nexists$ in **R**-component.)

R-Nodes

- For each child (edge) e:





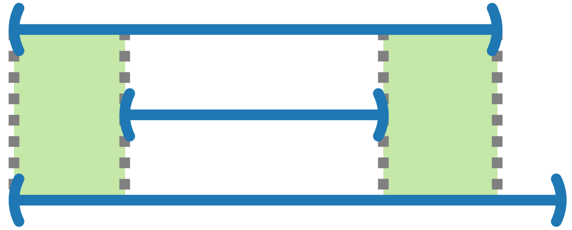


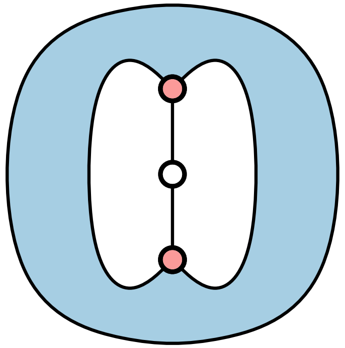
Separation pair!

($\nexists$ in **R**-component.)

R-Nodes

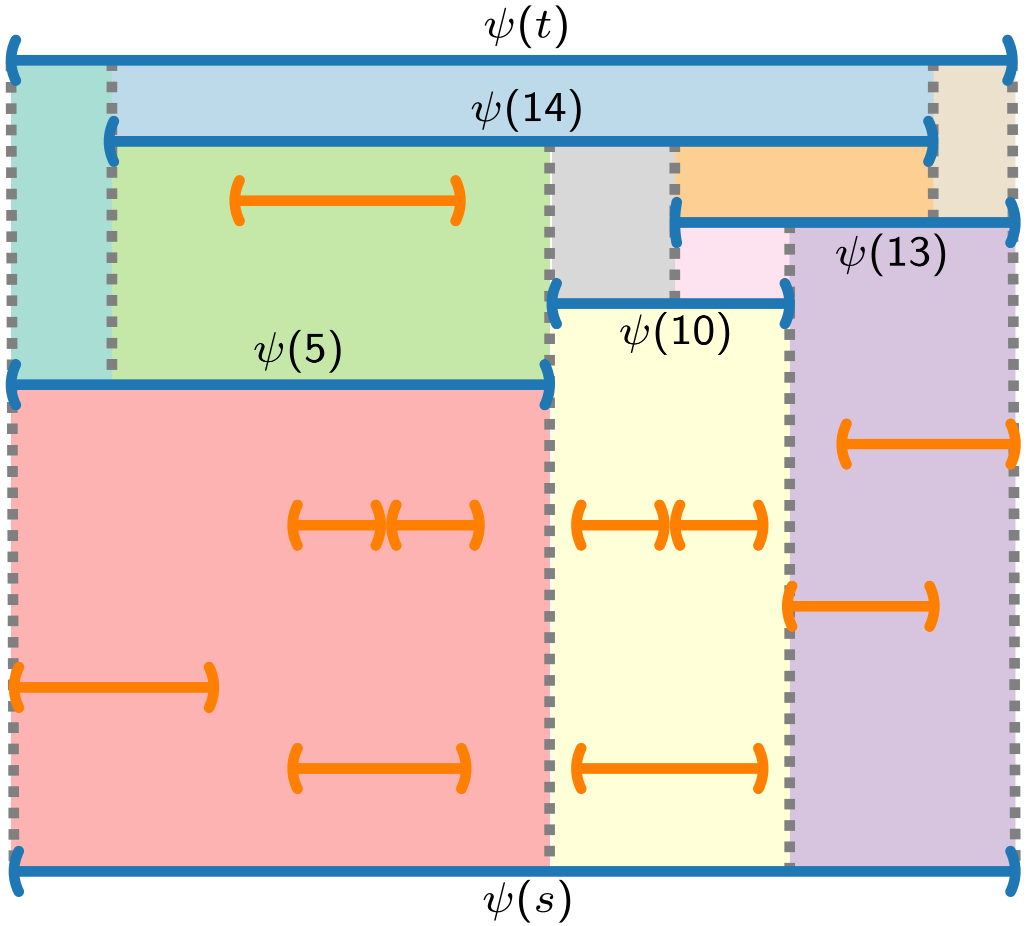
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.

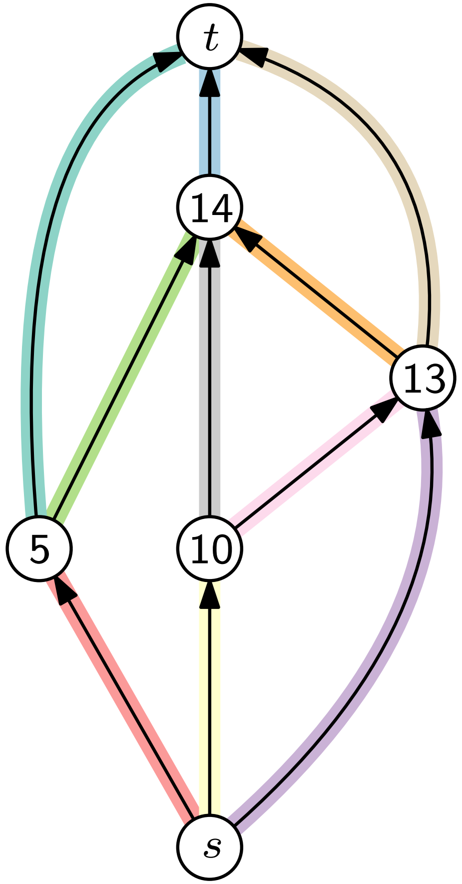




Separation pair!

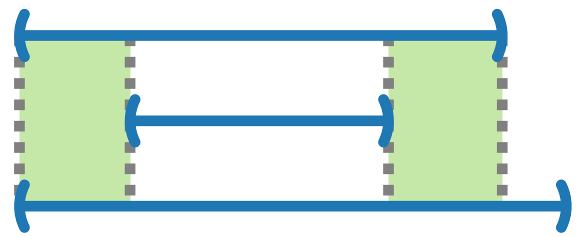
($\nexists$ in **R**-component.)

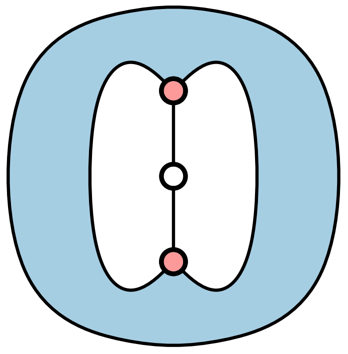


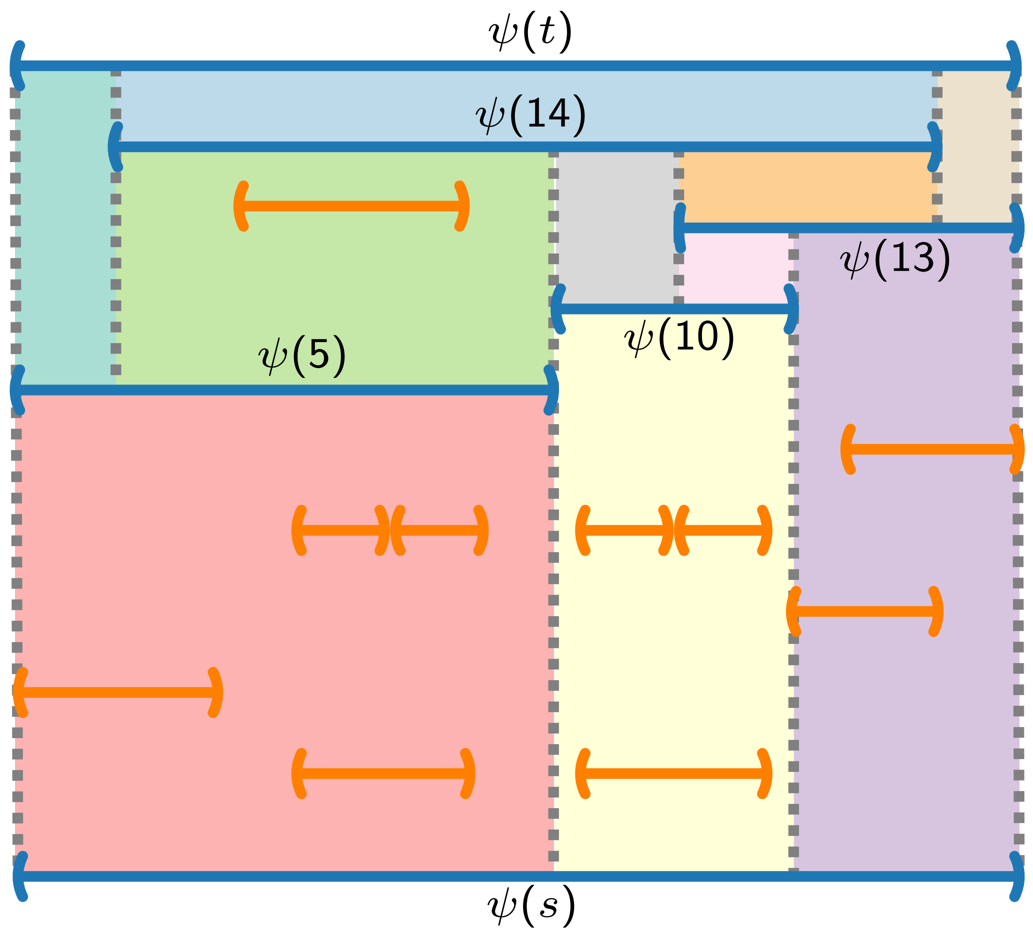


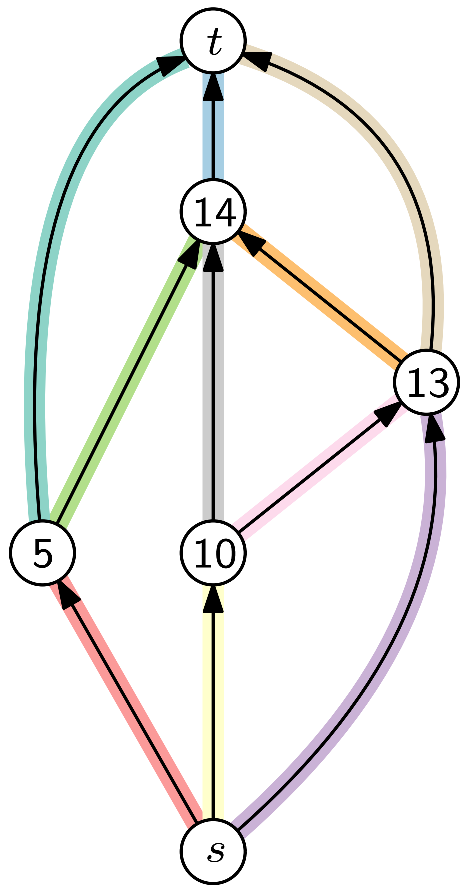
R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.



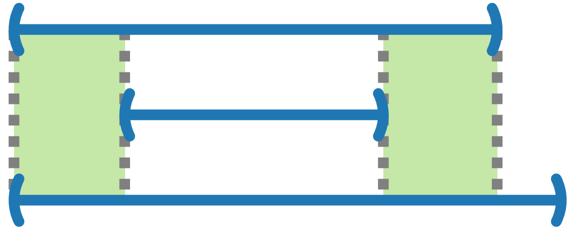


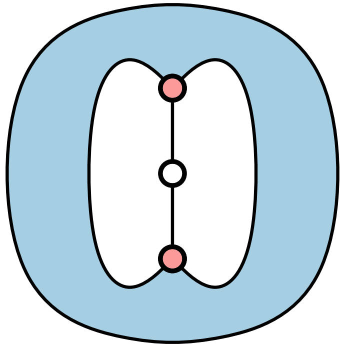




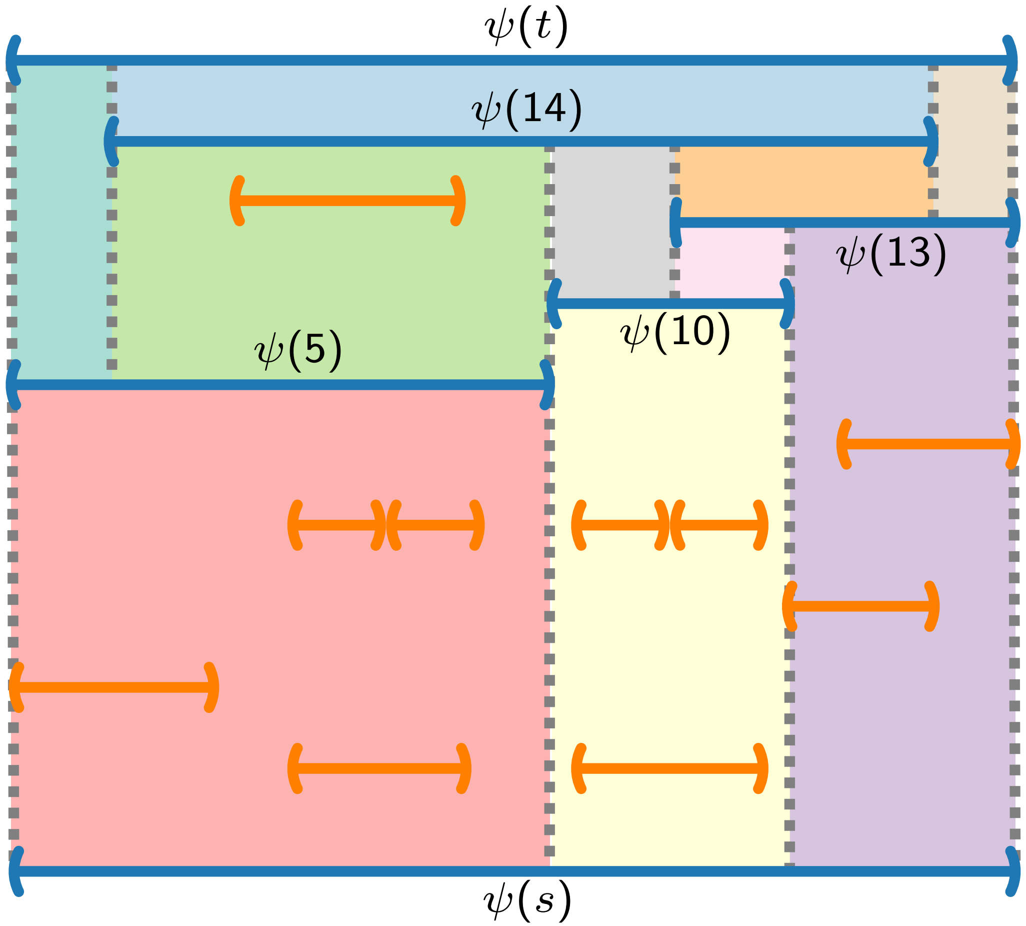
R-Nodes with 2-SAT Formulation

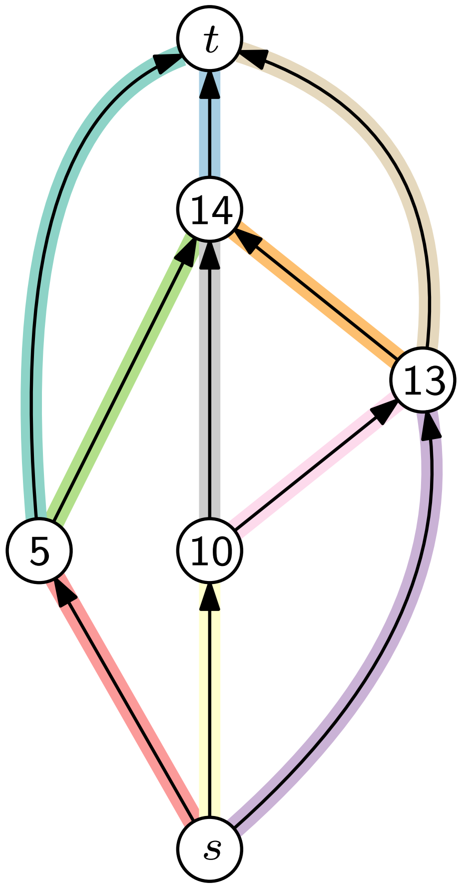
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).





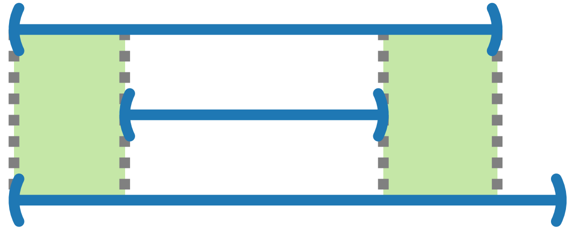
Separation pair!
($\nexists$ in **R**-component.)

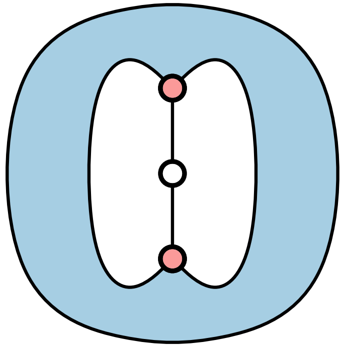


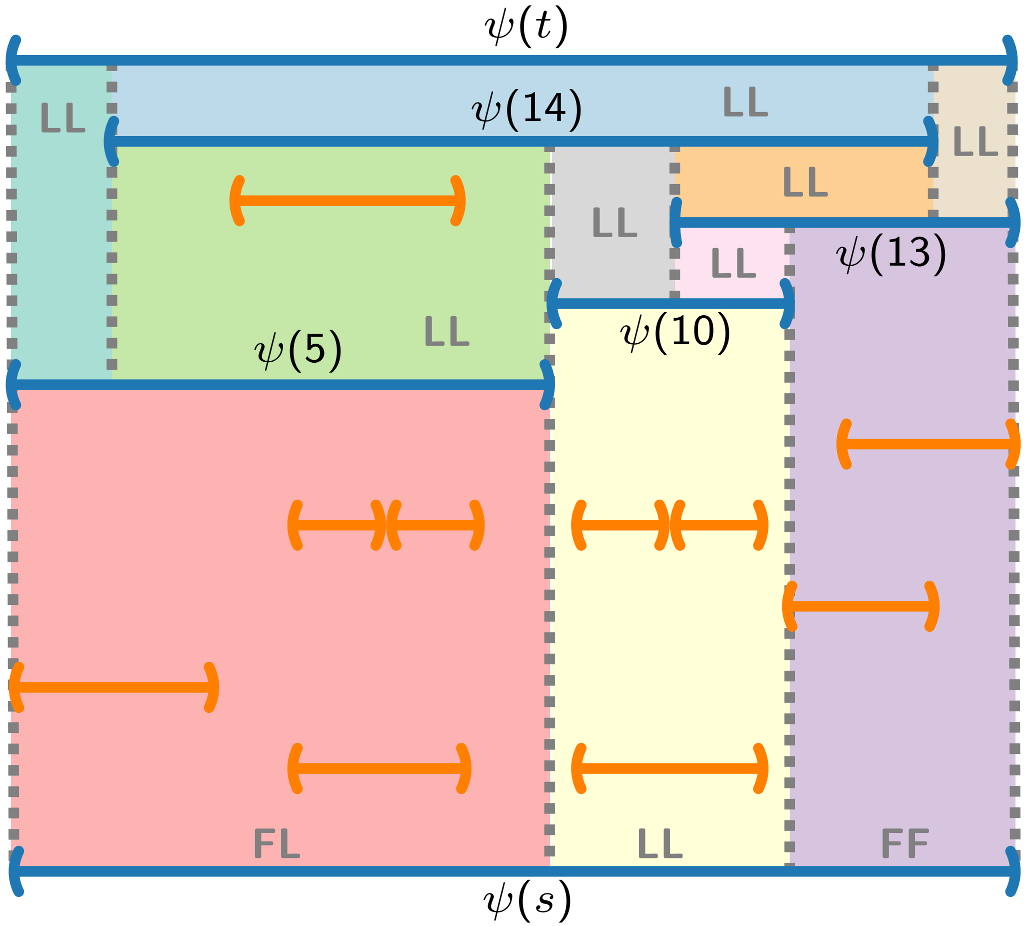


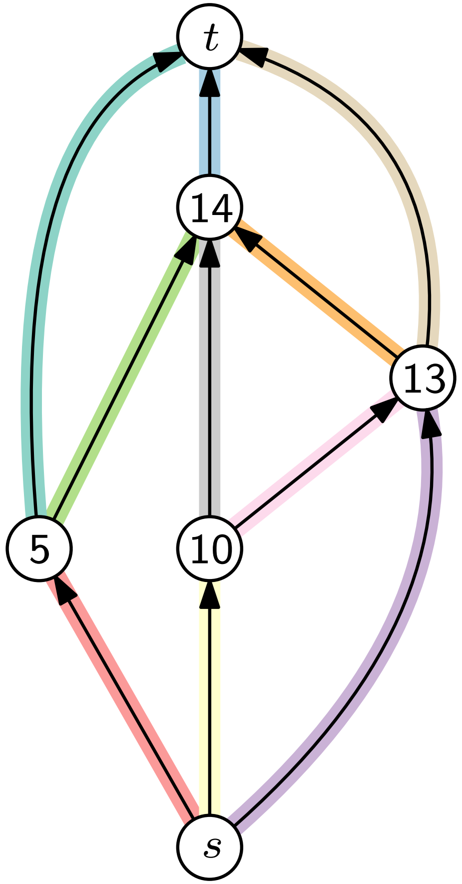
R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of $\{$**FF, FL, LF, LL**$\}$ that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).



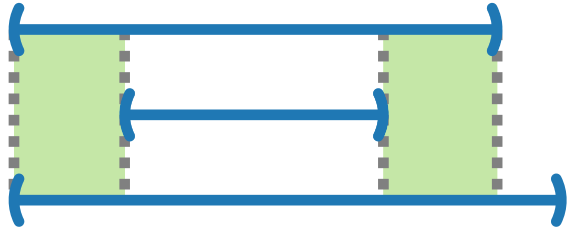


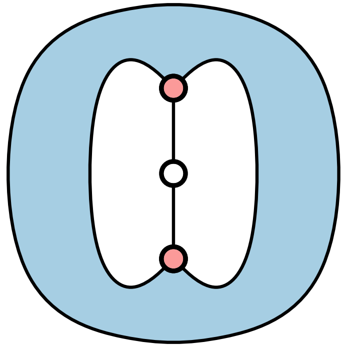


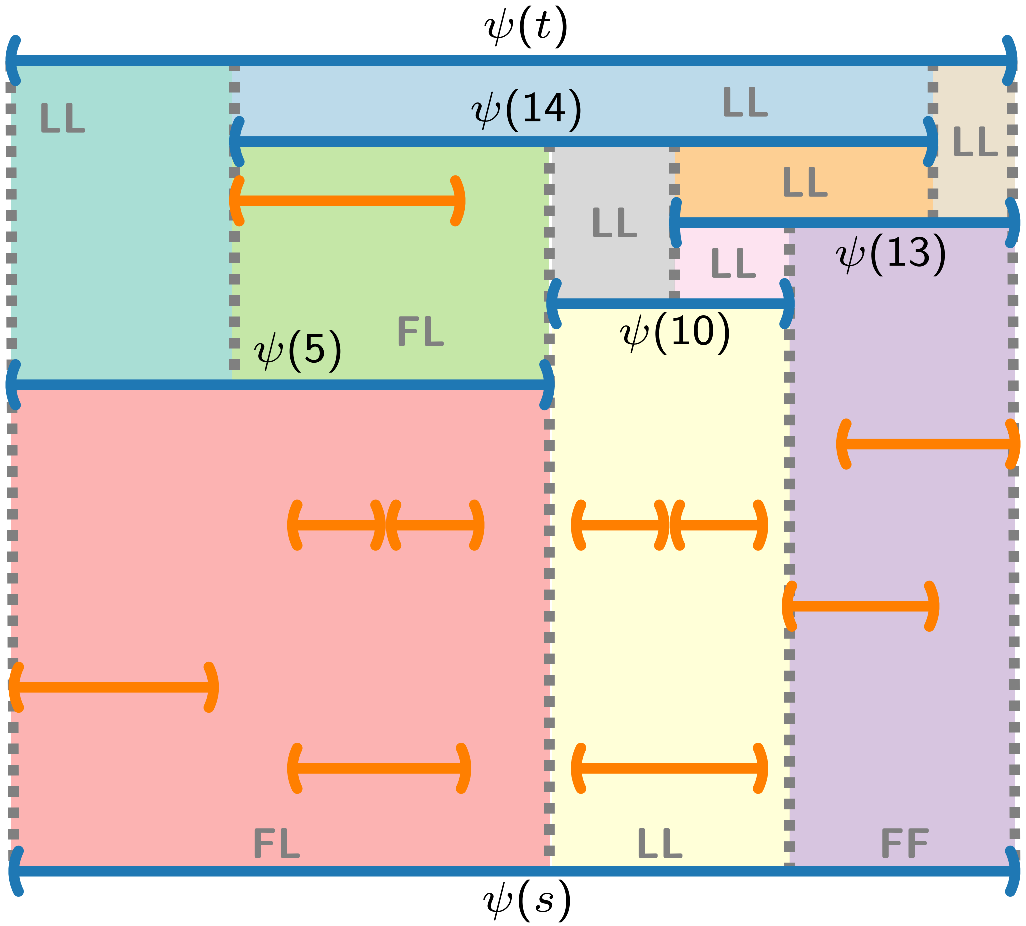


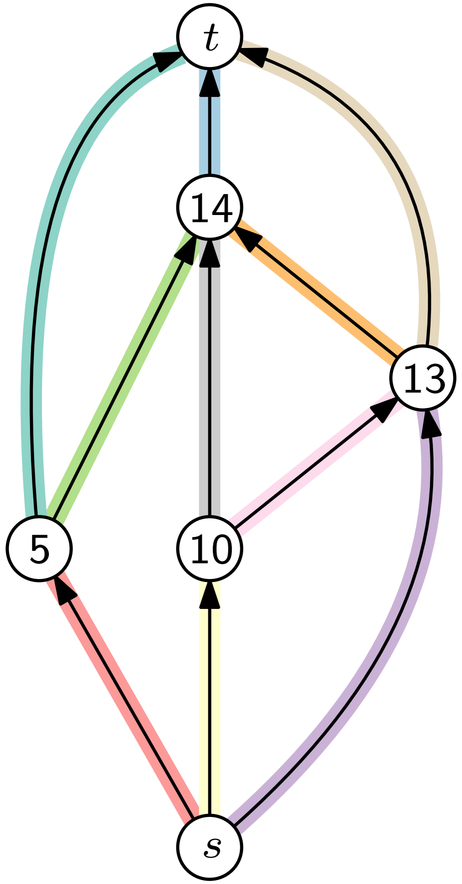
R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).



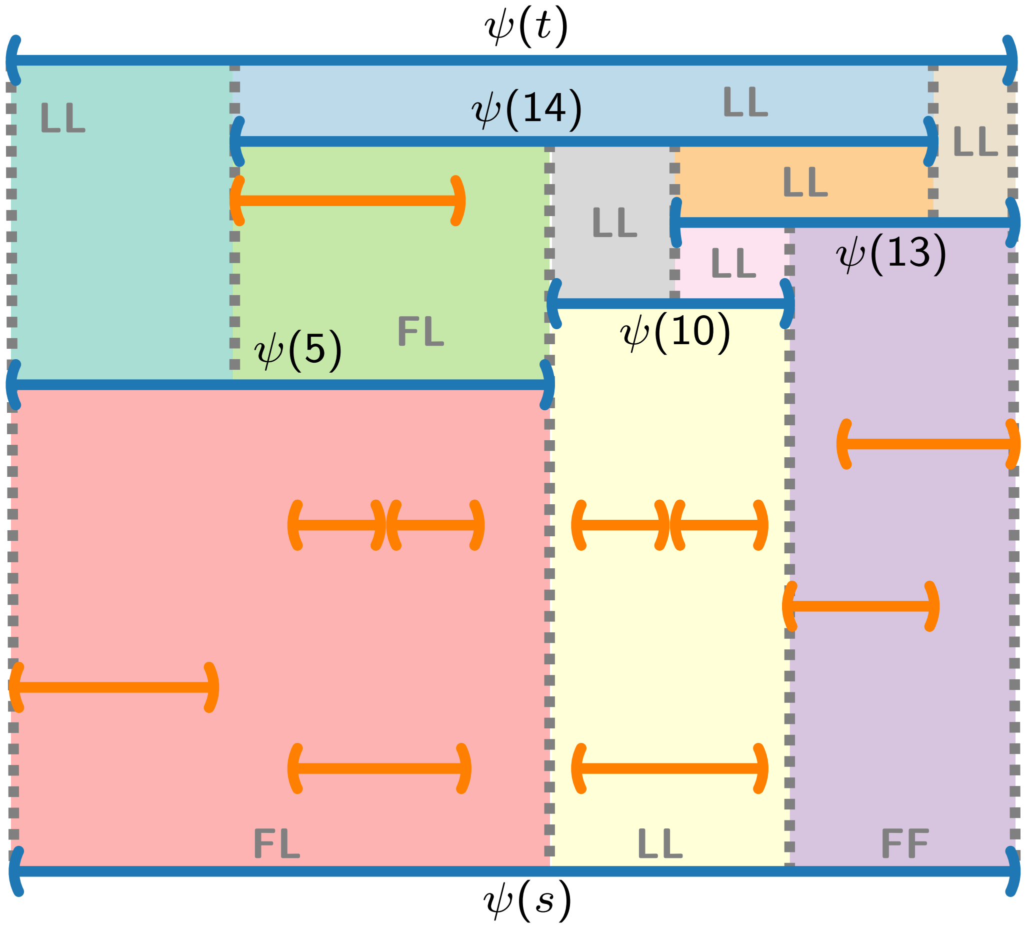


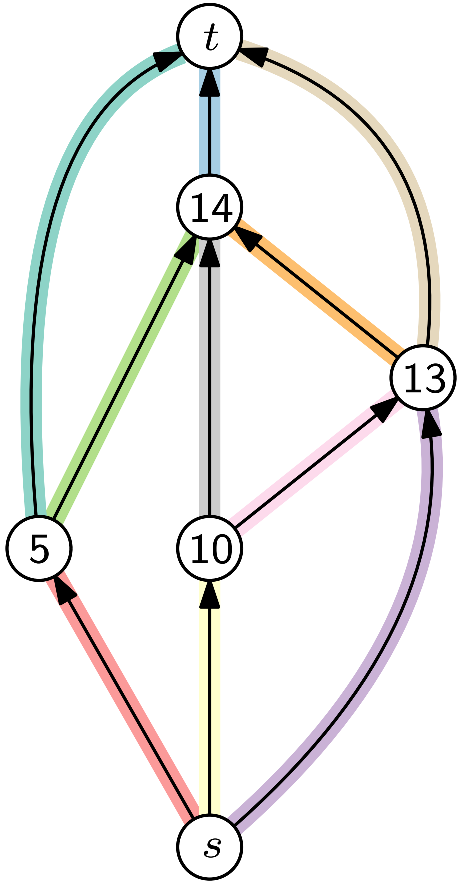


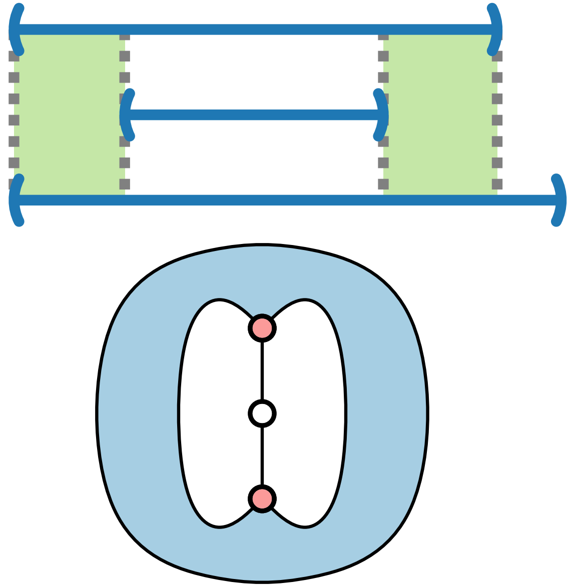


R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*





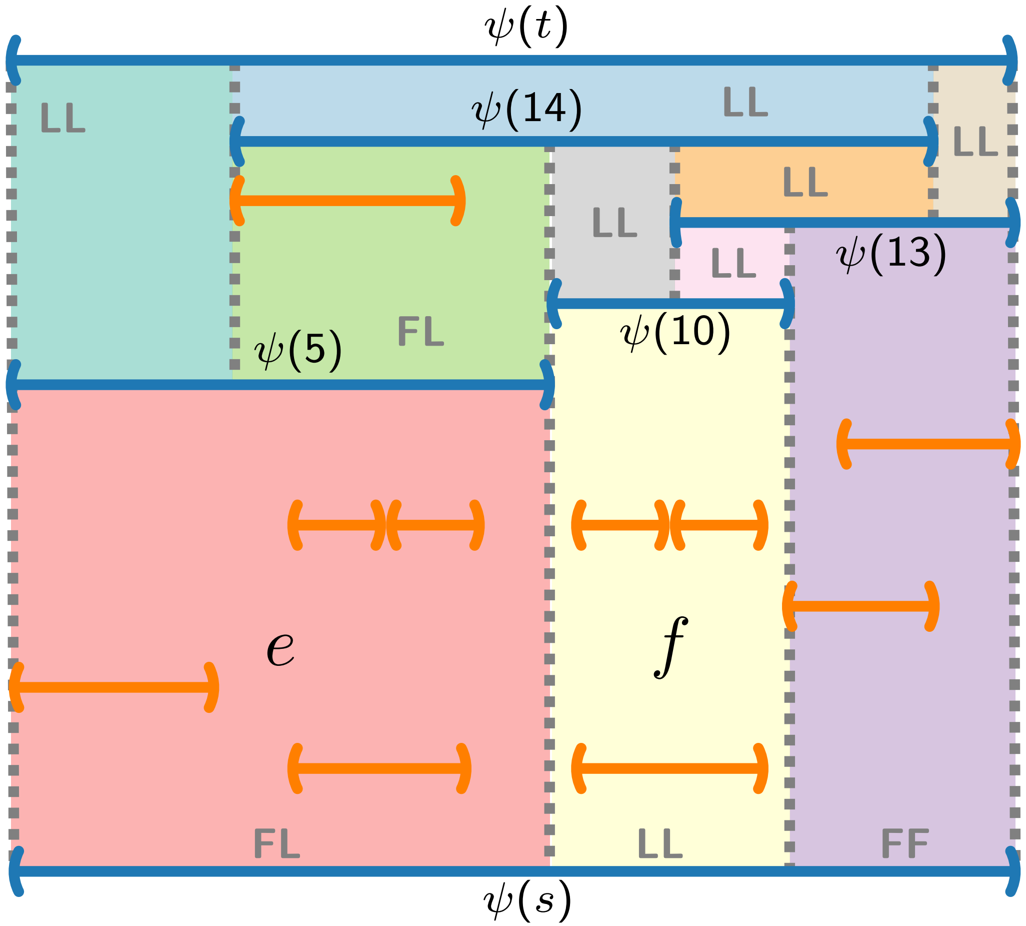


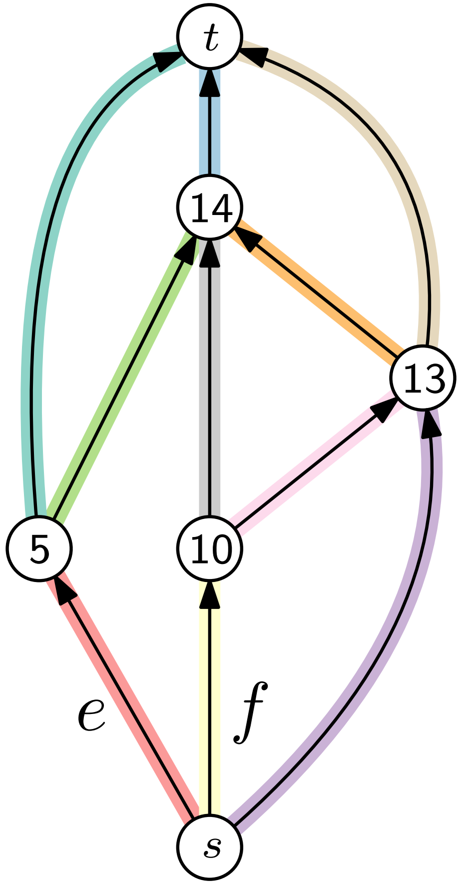
Separation pair!

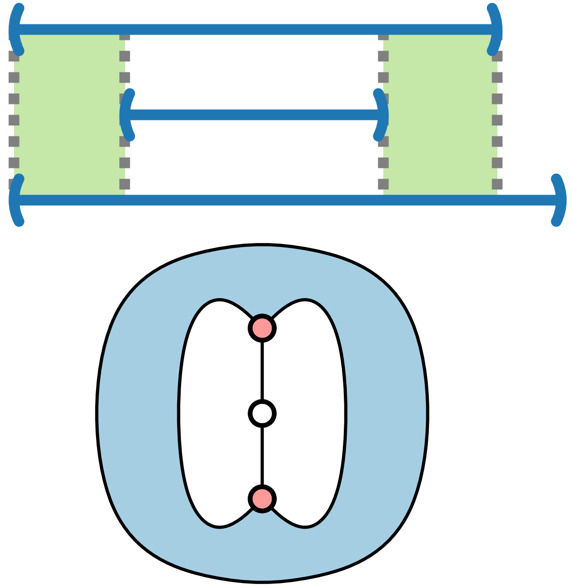
($\nexists$ in **R**-component.)

R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*





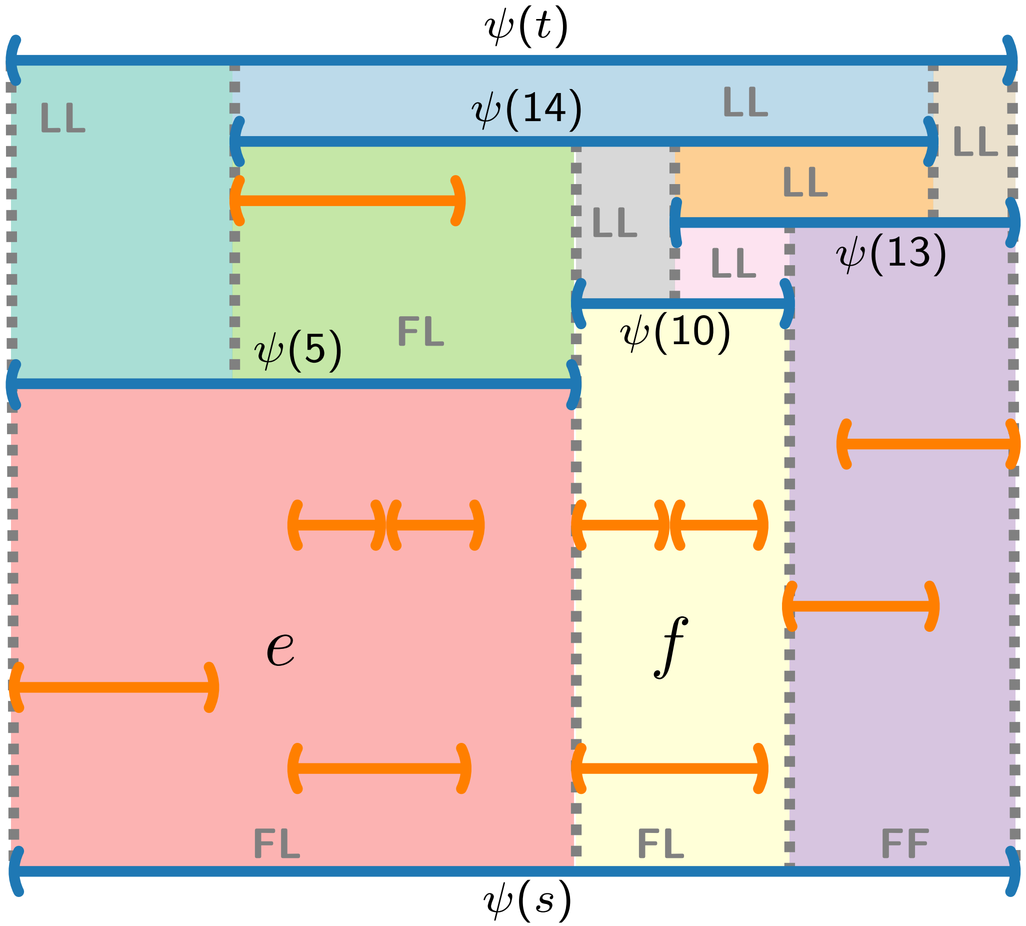


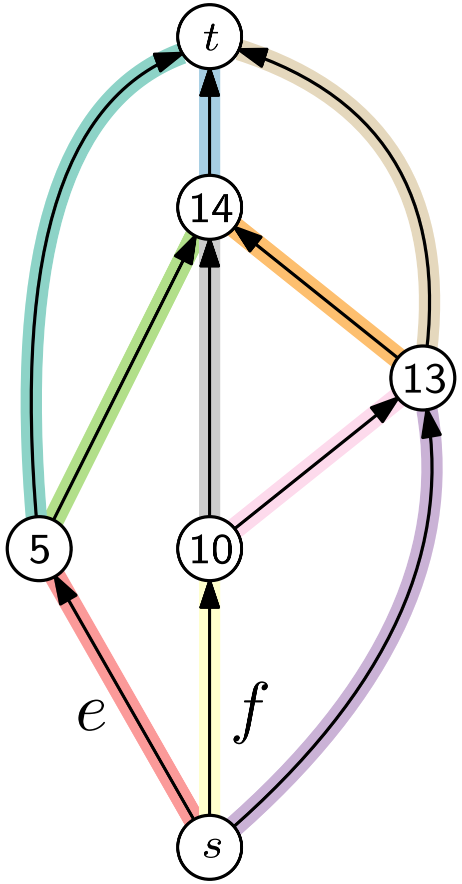
Separation pair!

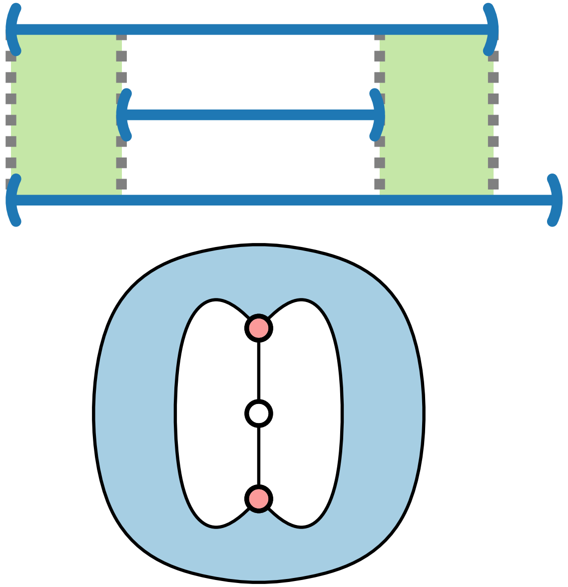
($\nexists$ in **R**-component.)

R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF**, **FL**, **LF**, **LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*





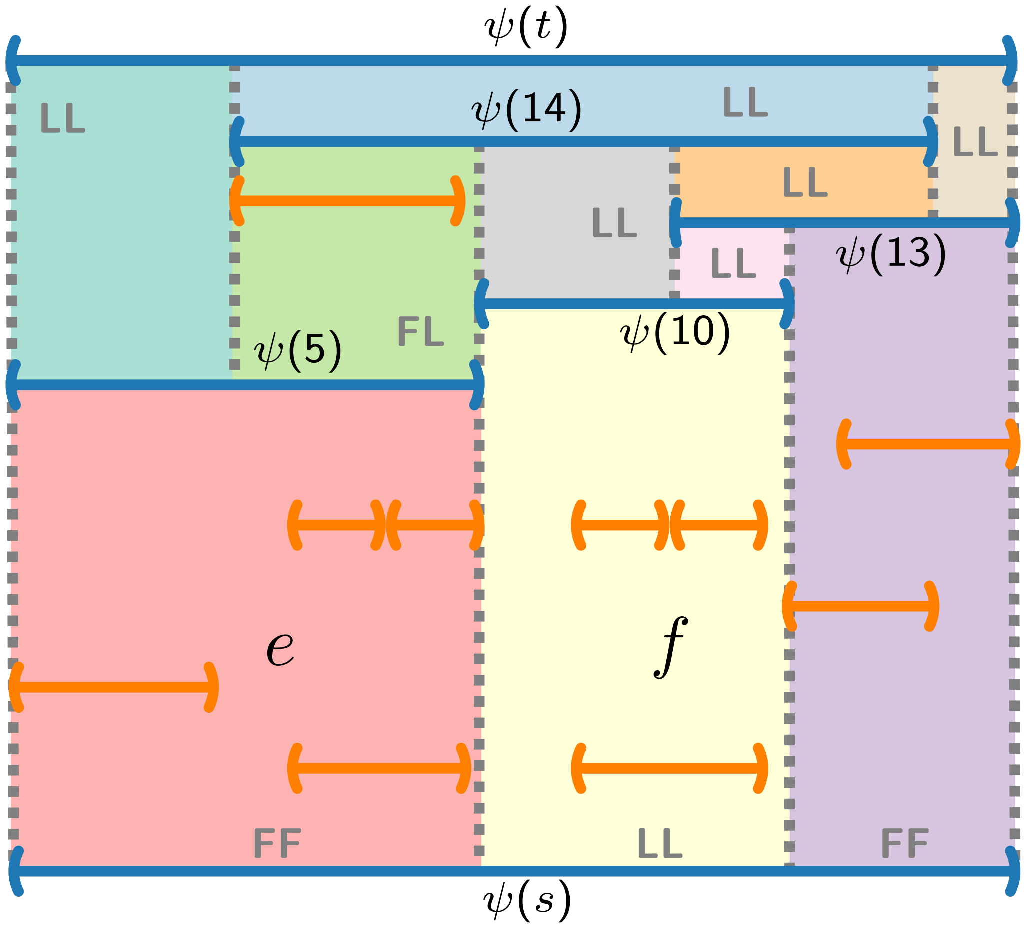


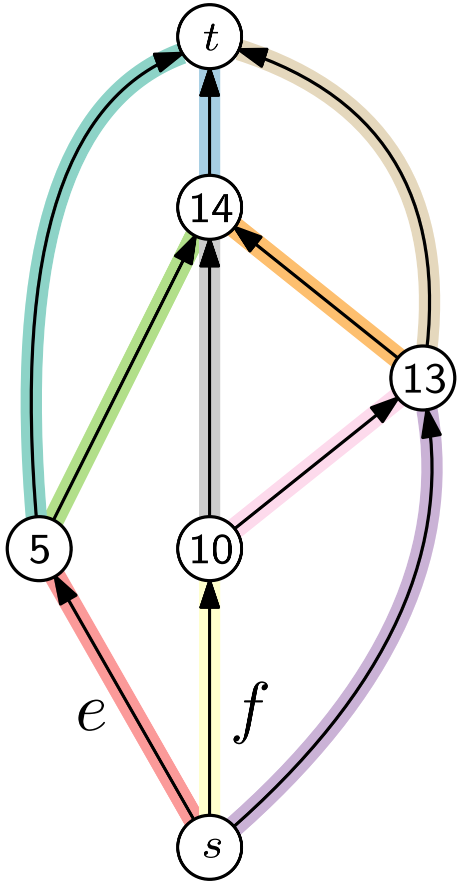
Separation pair!

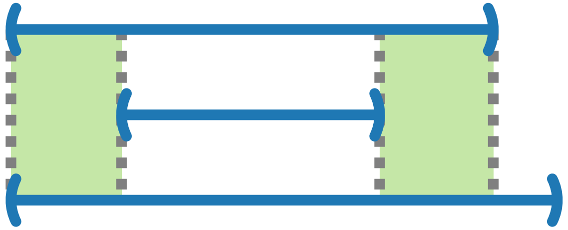
($\nexists$ in **R**-component.)

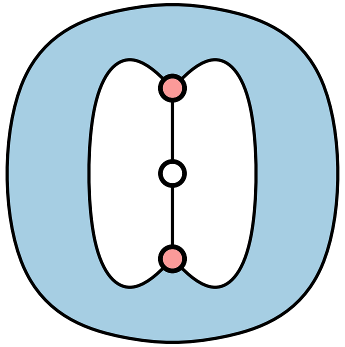
R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*







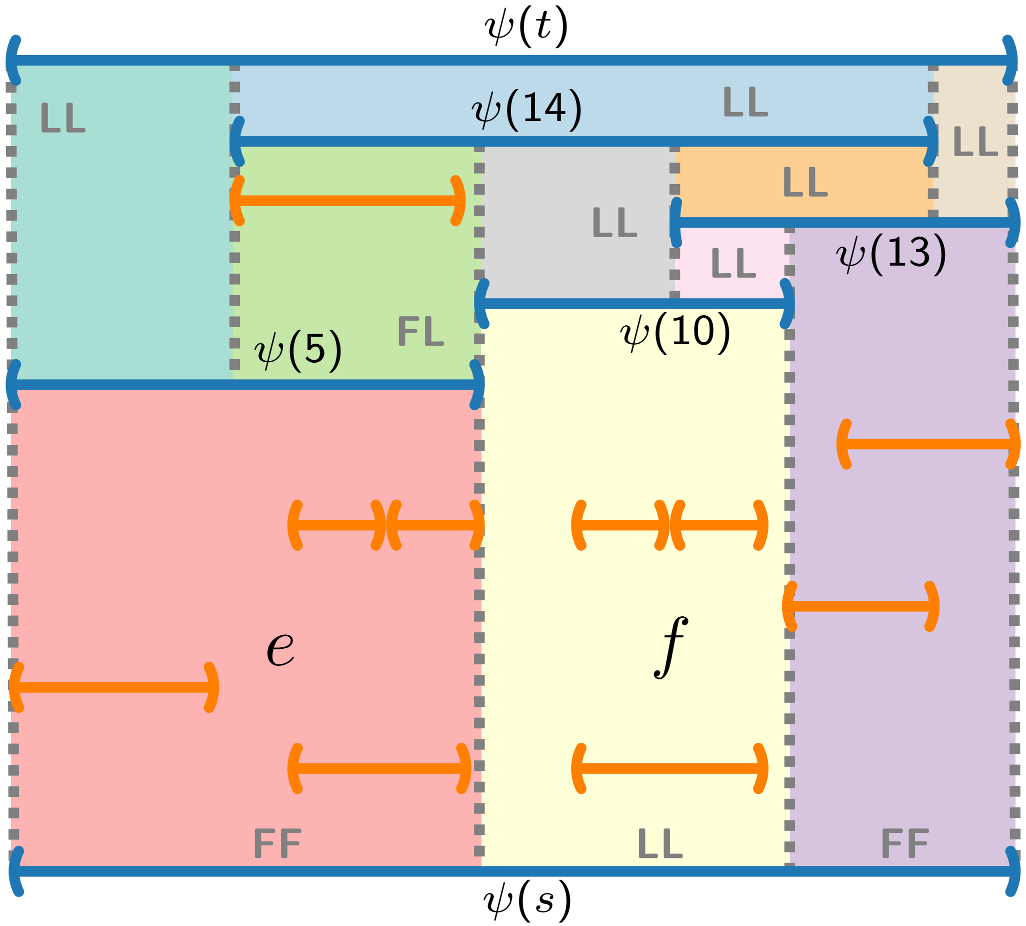


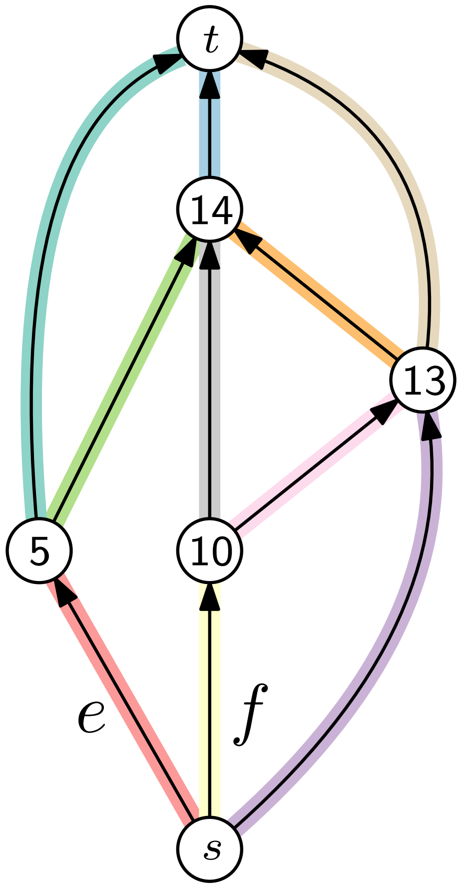
Separation pair!

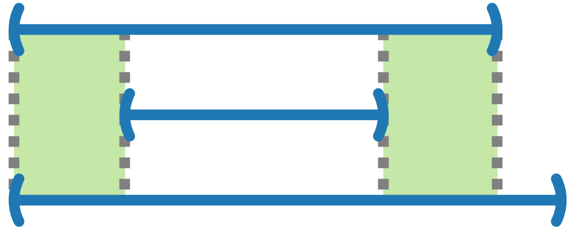
($\nexists$ in **R**-component.)

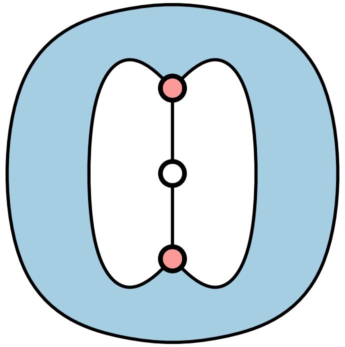
R-Nodes with 2-SAT Formulation

- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*: e.g., $\neg(\neg r_e \wedge \neg l_f)$









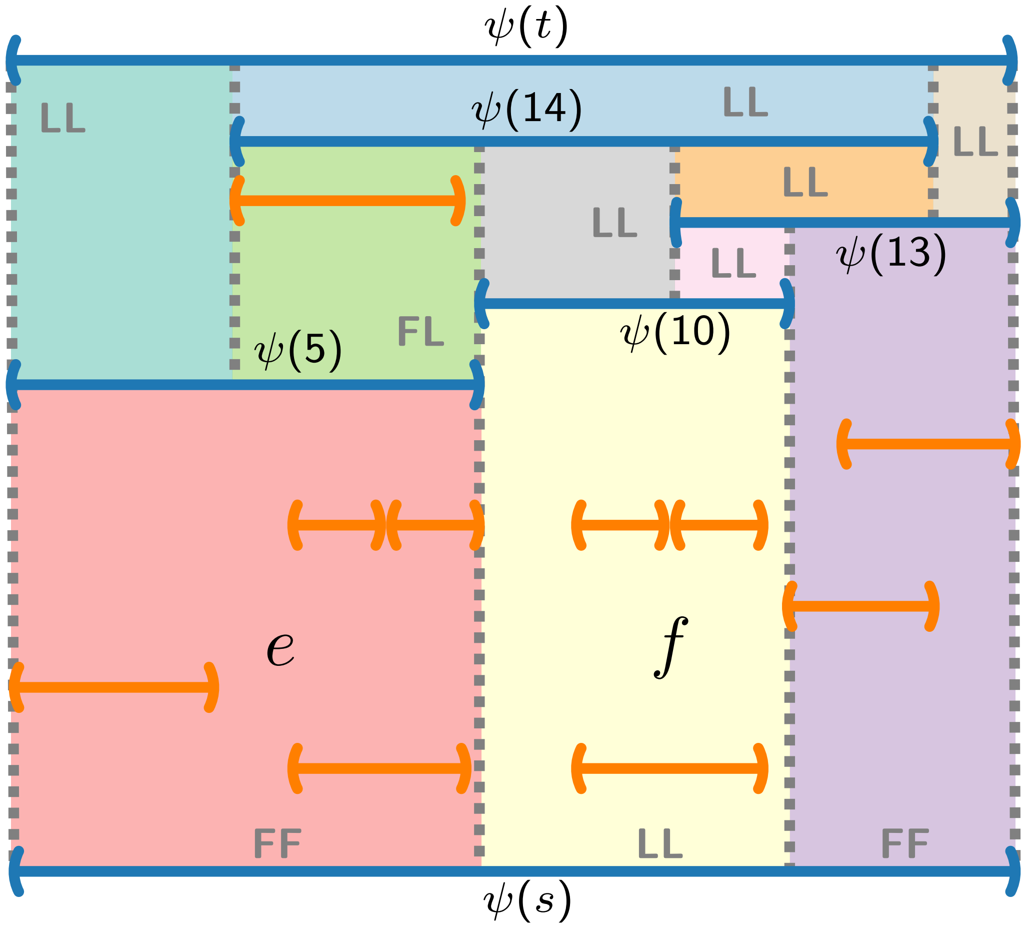
Separation pair!

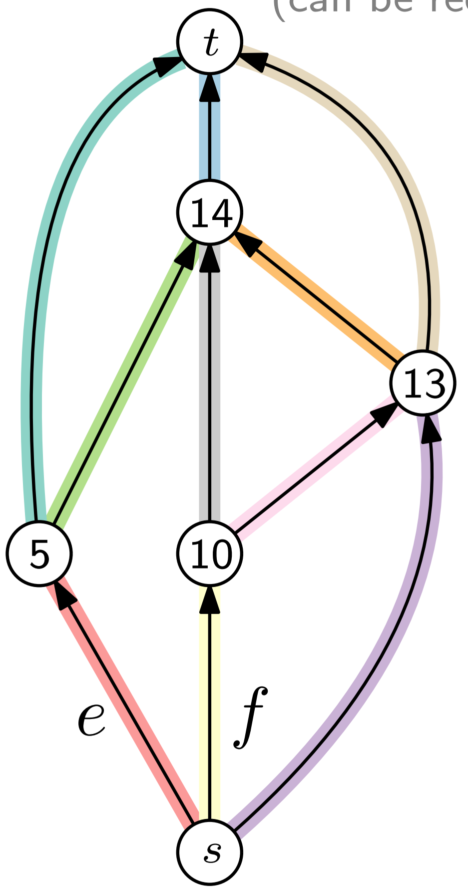
($\nexists$ in **R**-component.)

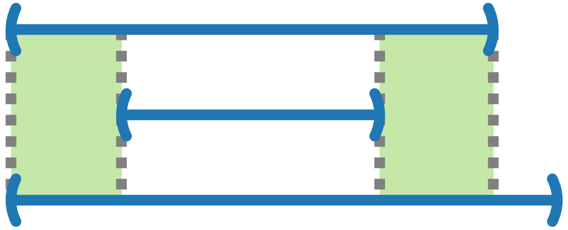
R-Nodes with 2-SAT Formulation

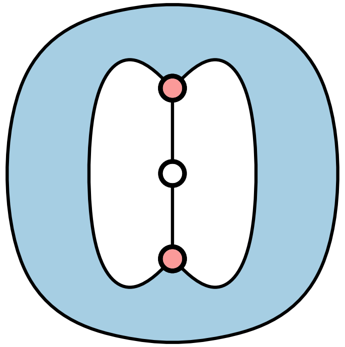
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*: e.g., $\neg(\neg r_e \wedge \neg l_f)$ $\rightarrow O(n^2)$ many.

(can be reduced to $O(n \log^2 n)$)









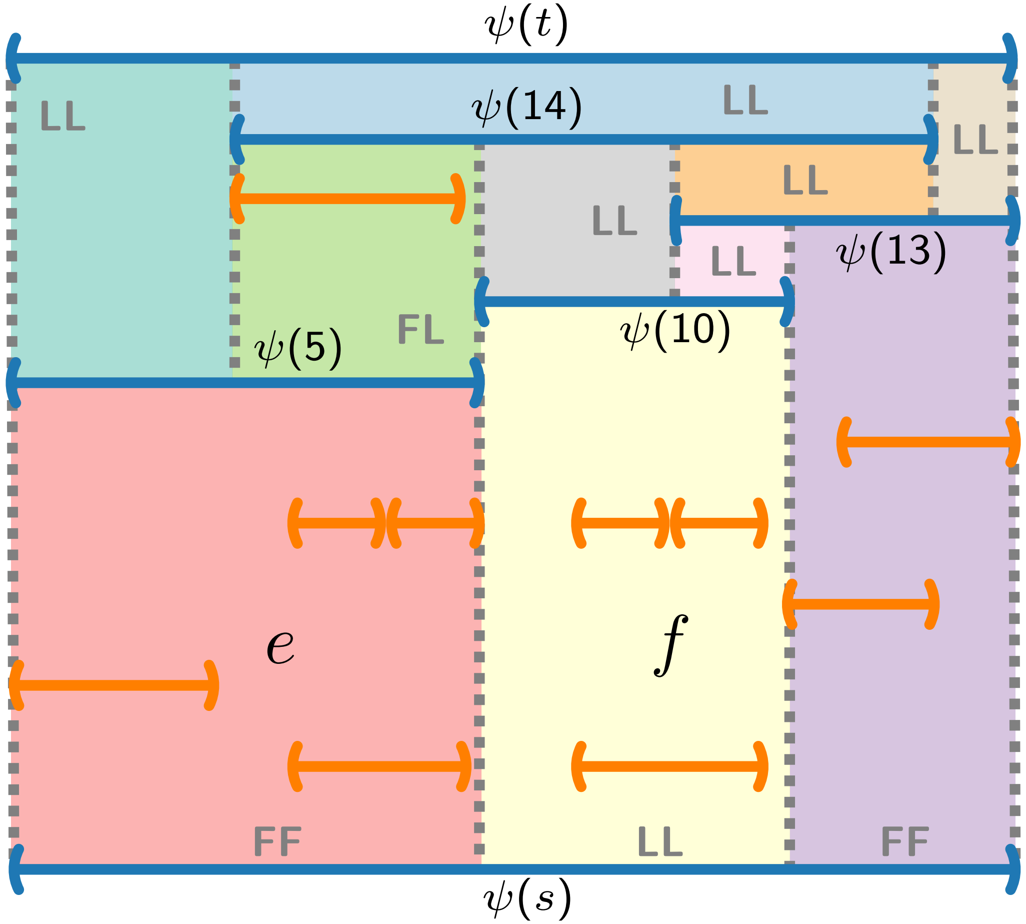
Separation pair!

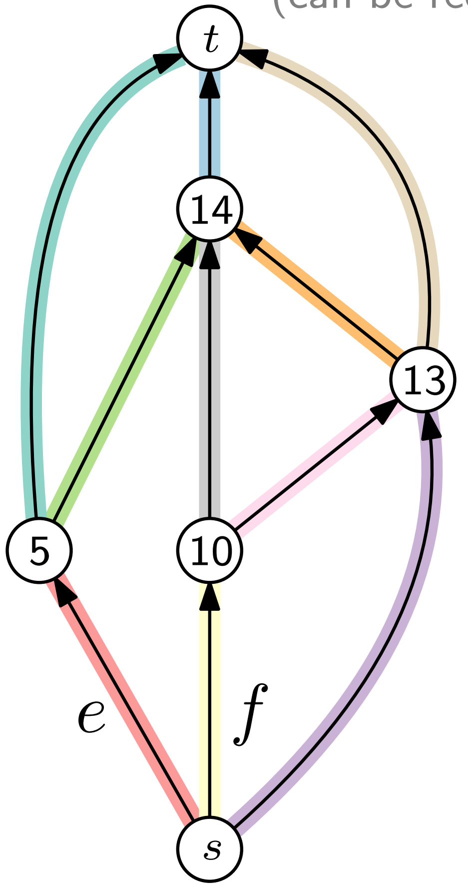
($\nexists$ in **R**-component.)

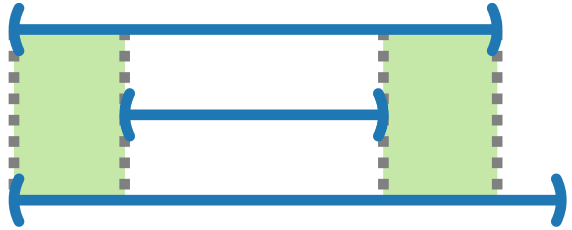
R-Nodes with 2-SAT Formulation

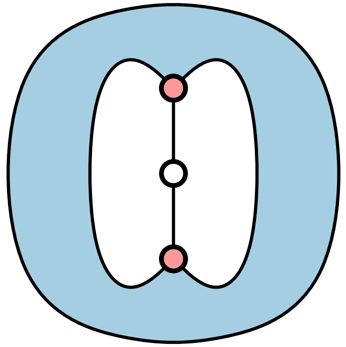
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*: e.g., $\neg(\neg r_e \wedge \neg l_f)$ $\rightarrow O(n^2)$ many.

(can be reduced to $O(n \log^2 n)$)









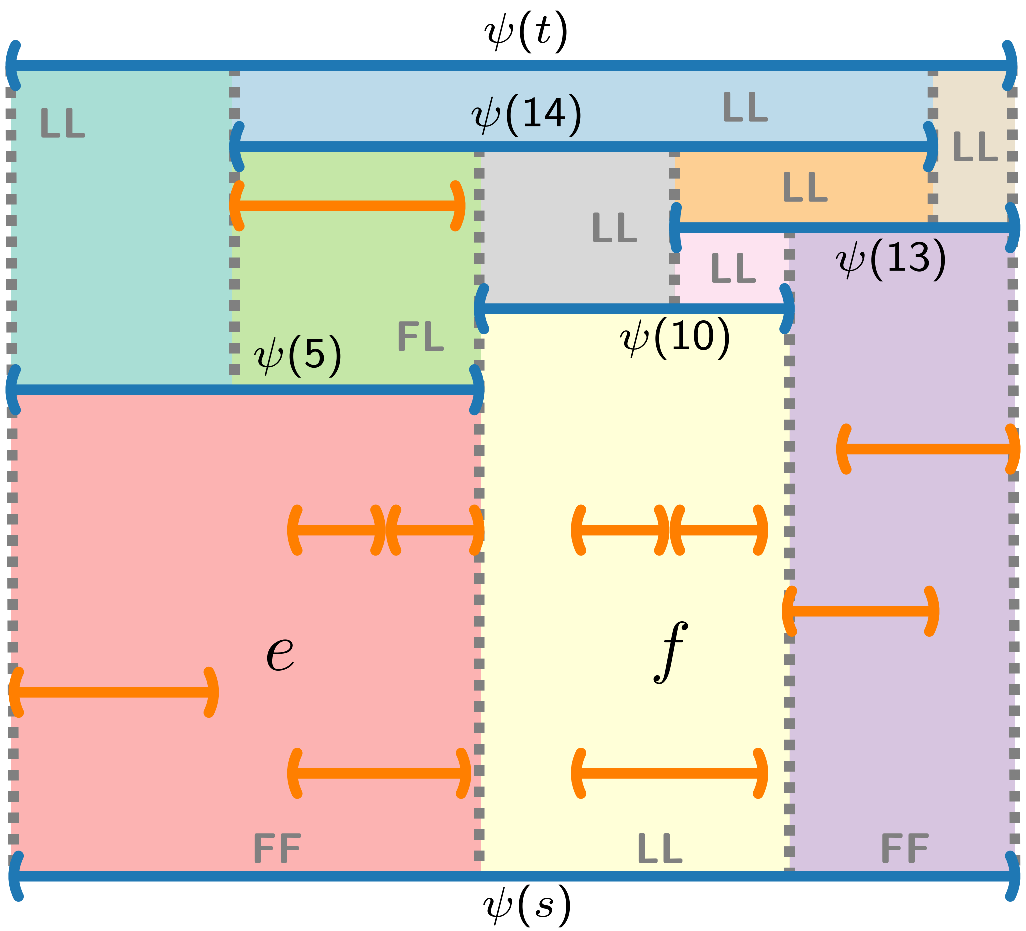
Separation pair!
($\nexists$ in **R**-component.)

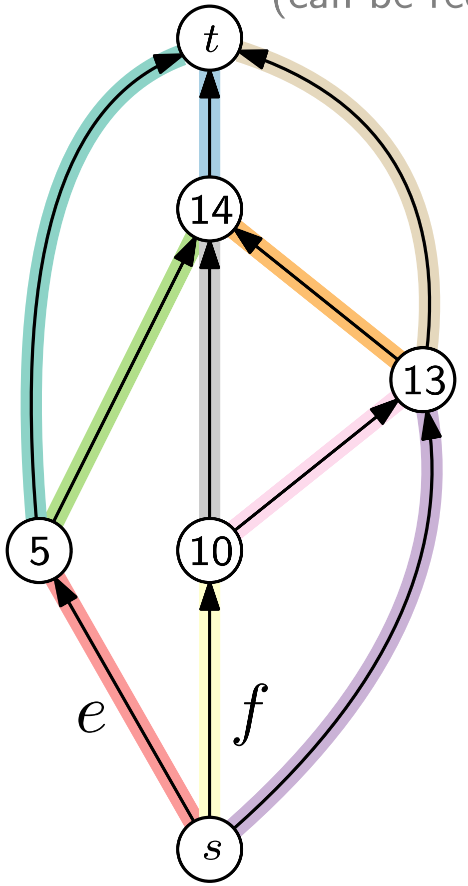
- Finding a satisfying assignment of a 2-SAT formula can be done in linear time!

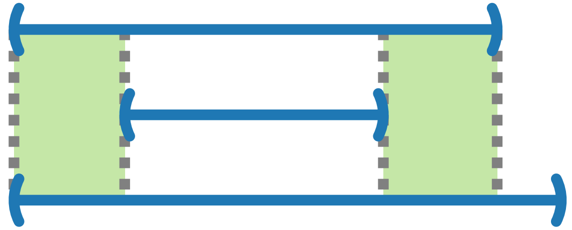
R-Nodes with 2-SAT Formulation

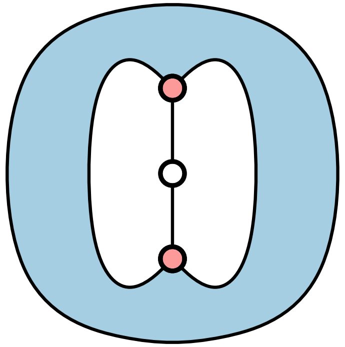
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*: e.g., $\neg(\neg r_e \wedge \neg l_f)$ $\rightarrow O(n^2)$ many.

(can be reduced to $O(n \log^2 n)$)









Separation pair!

($\nexists$ in **R**-component.)

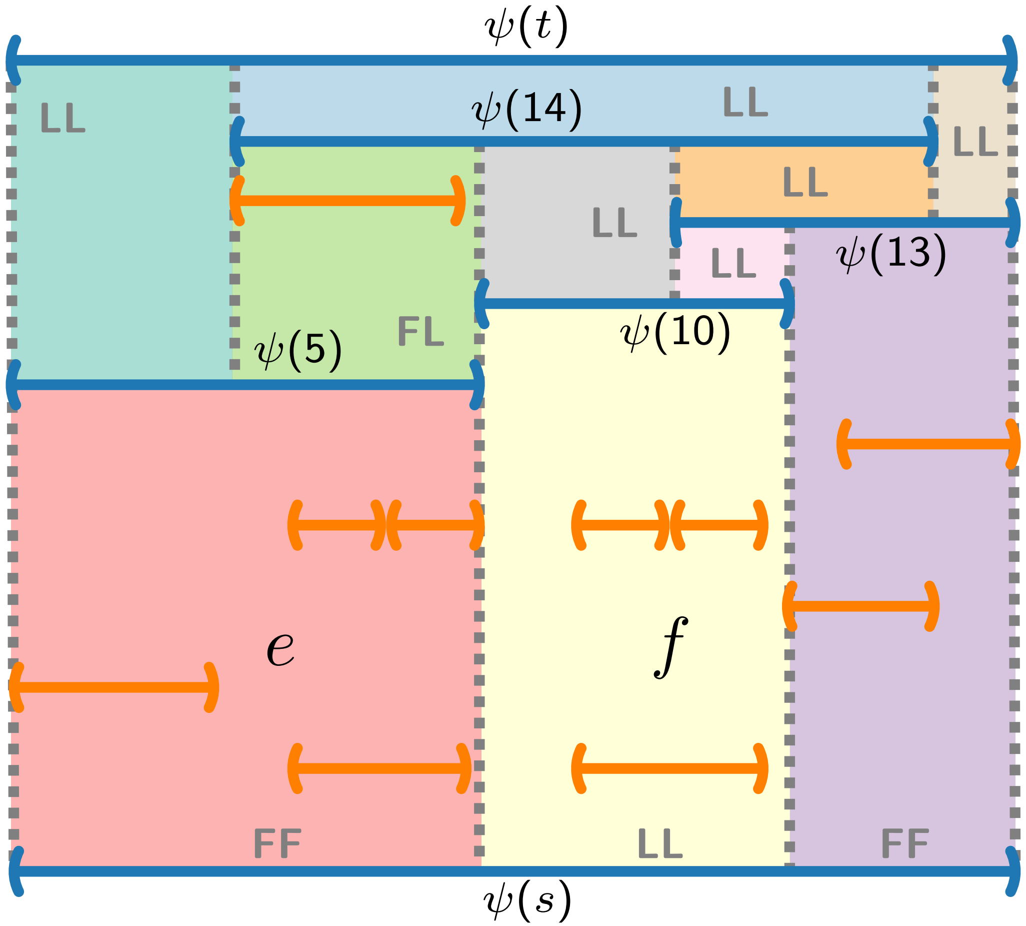
- Finding a satisfying assignment of a 2-SAT formula can be done in linear time!

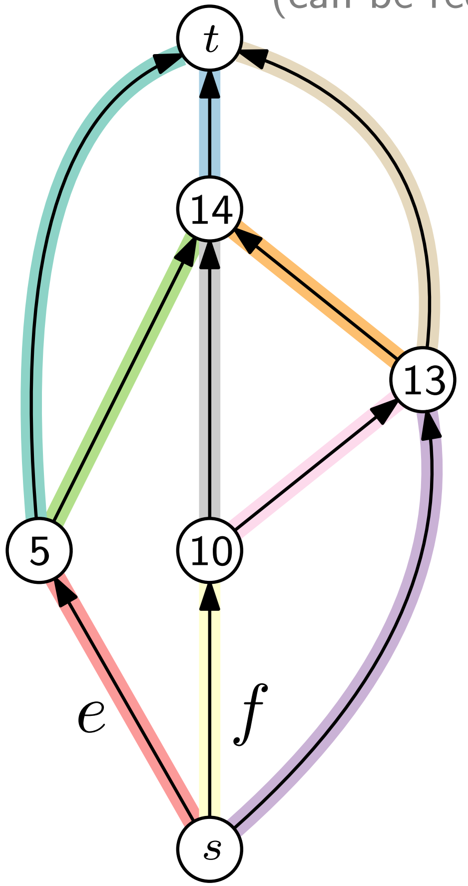
$\Rightarrow O(n^2)$ time in total

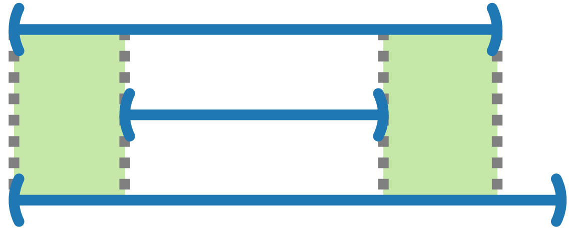
R-Nodes with 2-SAT Formulation

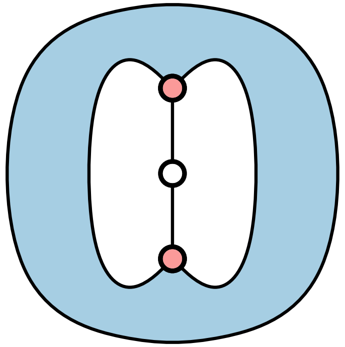
- For each child (edge) e:
  - Find all types of {**FF, FL, LF, LL**} that admit a drawing.
  - Use two variables (l_e and r_e) to encode the type of its tile (**F** $= 0$).
  - Add *consistency clauses*: e.g., $\neg(\neg r_e \wedge \neg l_f)$ $\rightarrow O(n^2)$ many.

(can be reduced to $O(n \log^2 n)$)









Separation pair!

($\nexists$ in **R**-component.)

- Finding a satisfying assignment of a 2-SAT formula can be done in linear time!

$\Rightarrow O(n^2)$ time in total

or $O(n \log^2 n)$

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time *for st-graphs*.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from Planar Monotone 3-SAT

Theorem 3.
ε-bar visibility representation extension is NP-complete even for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from 3-Partition

Results and Outline

[Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]

Theorem 1.
Rectangular ε-bar visibility representation extension can be solved in $\mathcal{O}(n \log^2 n)$ time *for st-graphs*.

- Dynamic program via SPQR-trees
- Easier version: $\mathcal{O}(n^2)$

Theorem 2.
ε-bar visibility representation extension is NP-complete.

- Reduction from Planar Monotone 3-SAT

Theorem 3.
ε-bar visibility representation extension is NP-complete even for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from 3-Partition

NP-Hardness of RepExt in the General Case

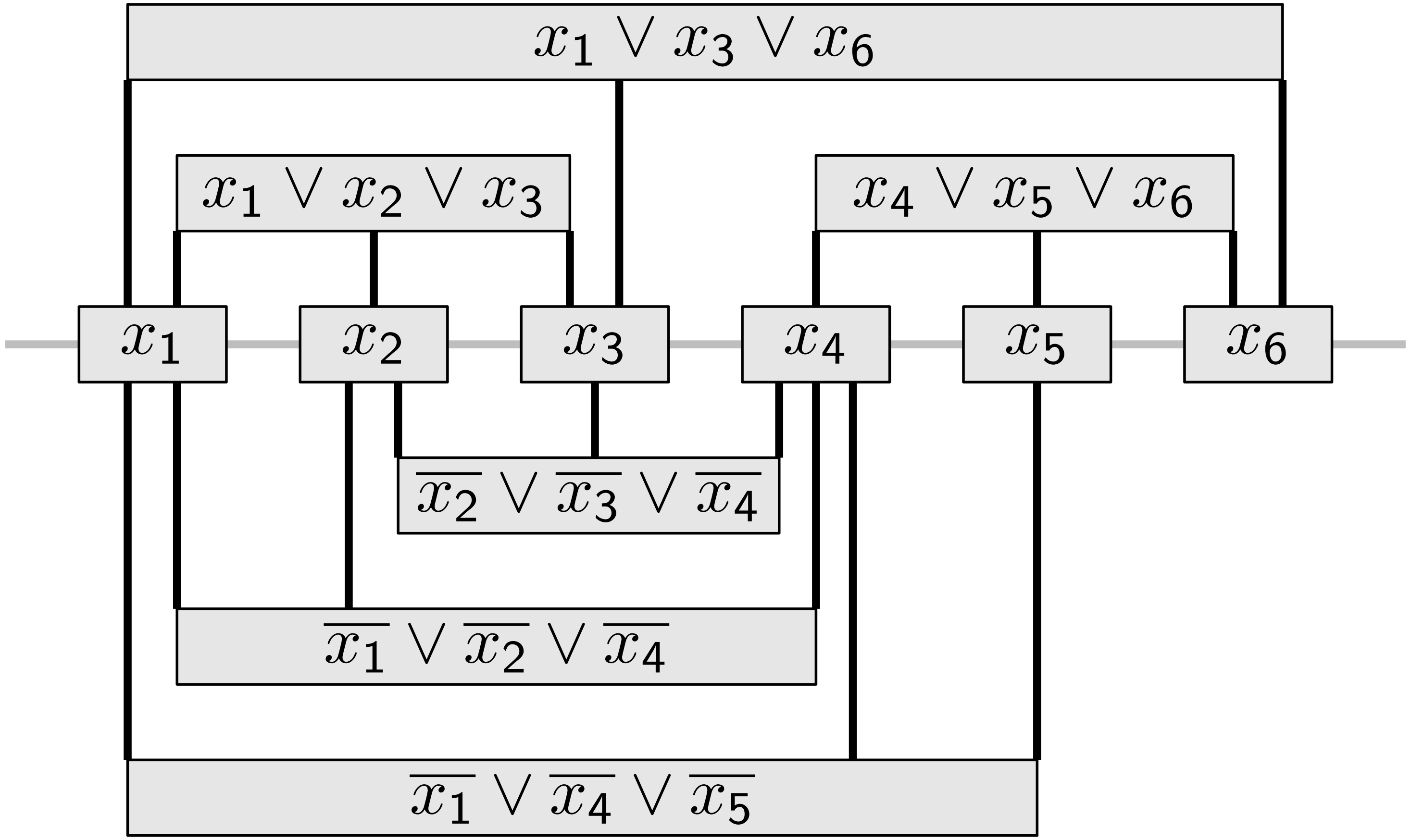
Theorem 2.
ε-Bar visibility representation extension is NP-complete.

- Membership in NP?
- NP-hard: Reduction from Planar Monotone 3-SAT

NP-Hardness of RepExt in the General Case

Theorem 2.
ε-Bar visibility representation extension is NP-complete.

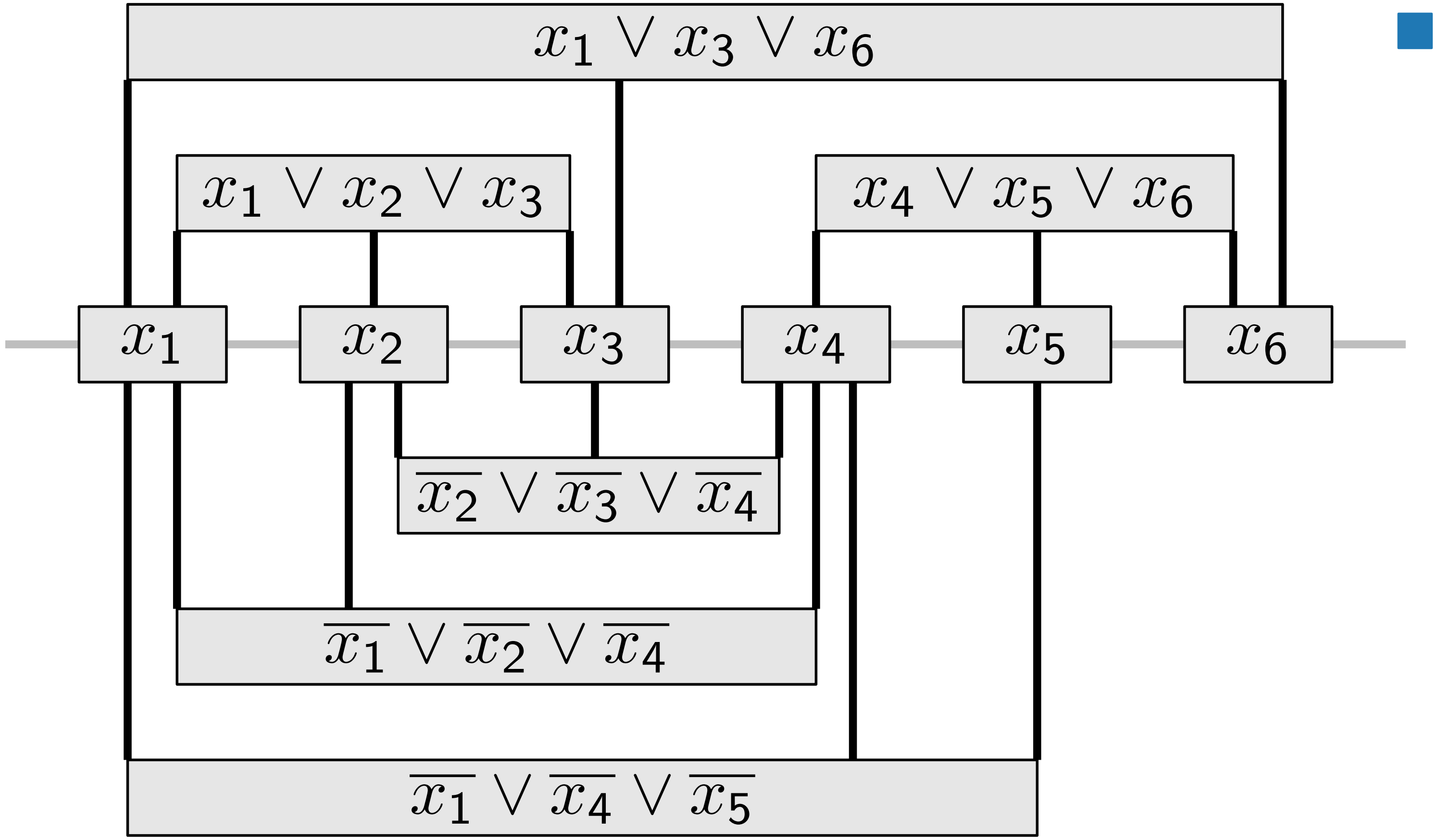
- Membership in NP?
- NP-hard: Reduction from Planar Monotone 3-SAT



NP-Hardness of RepExt in the General Case

Theorem 2.
ε-Bar visibility representation extension is NP-complete.

- Membership in NP?
- NP-hard: Reduction from Planar Monotone 3-SAT

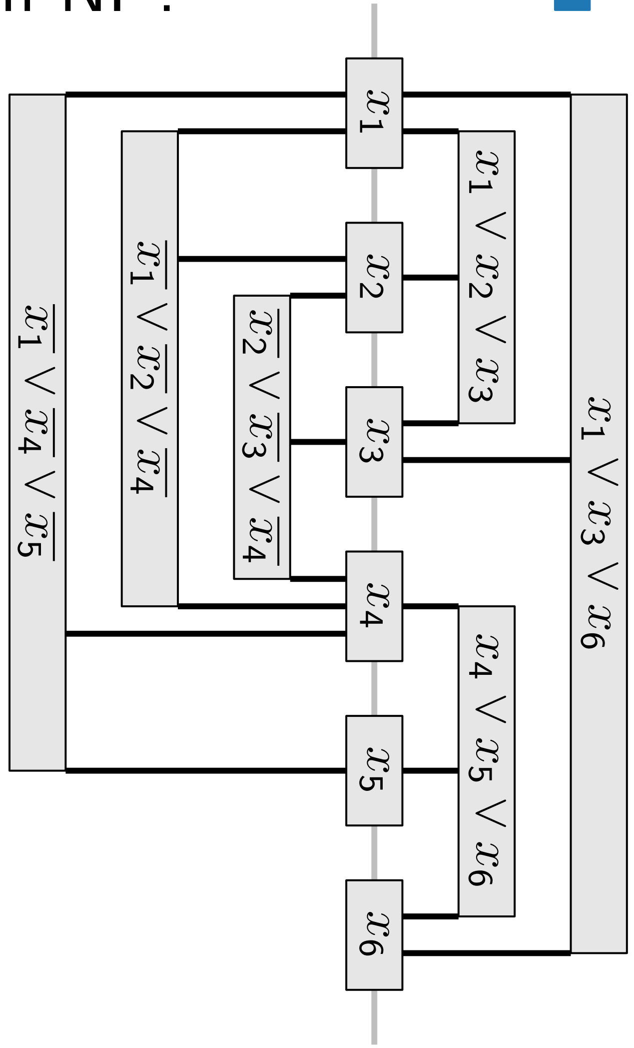


- NP-complete
  [de Berg & Khosravi ’10]

NP-Hardness of RepExt in the General Case

Theorem 2.
ε-Bar visibility representation extension is NP-complete.

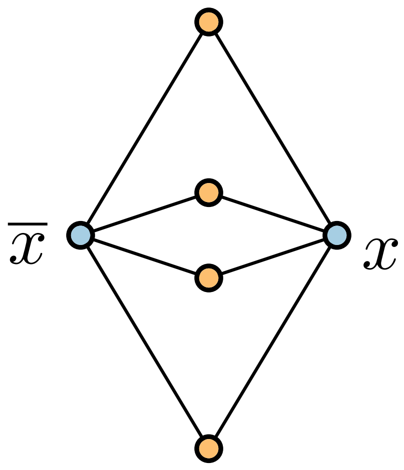
- Membership in NP?



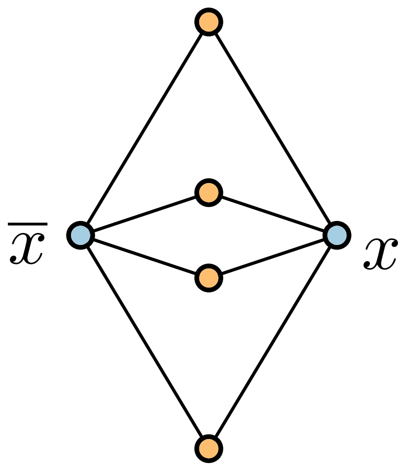
- NP-hard: Reduction from Planar Monotone 3-SAT

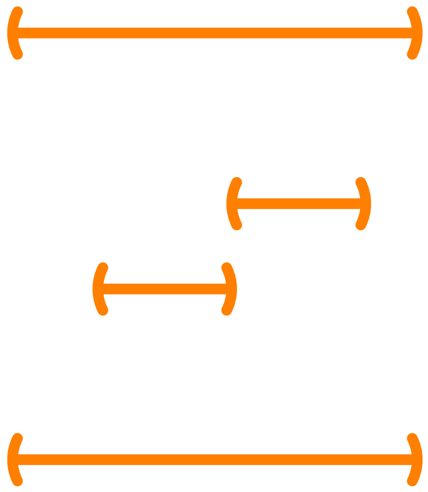
- NP-complete
  [de Berg & Khosravi '10]

Variable Gadget

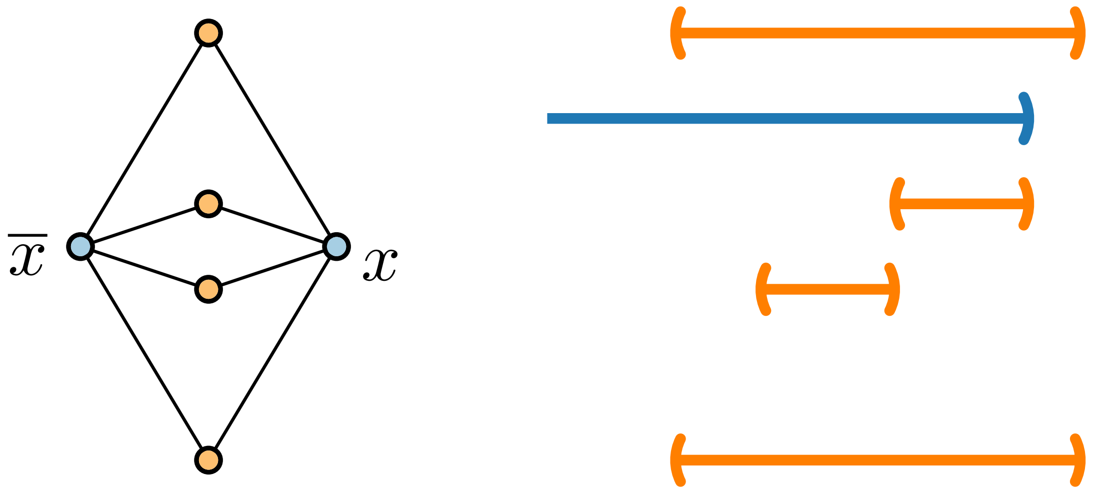


Variable Gadget

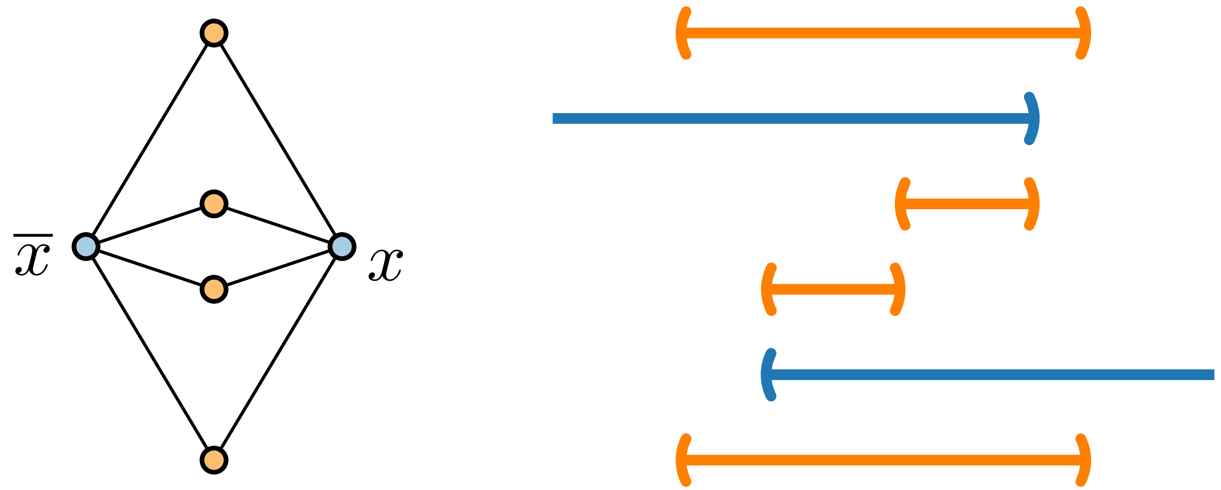




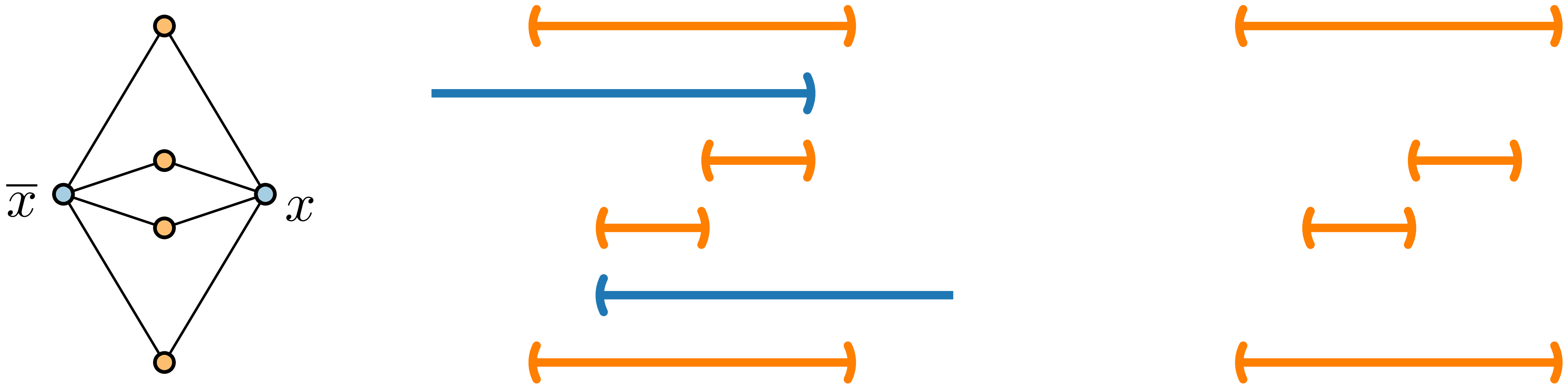
Variable Gadget



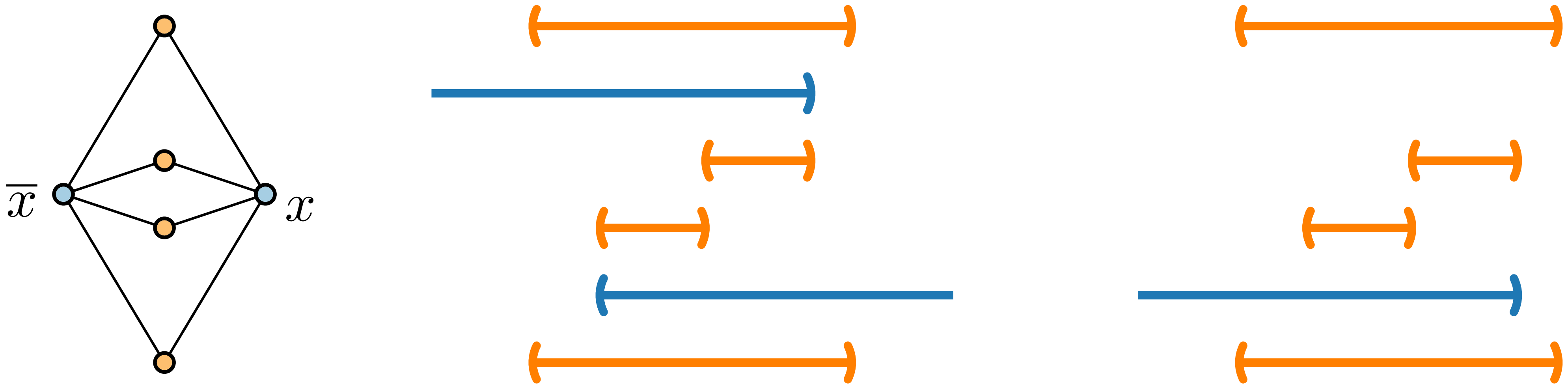
Variable Gadget



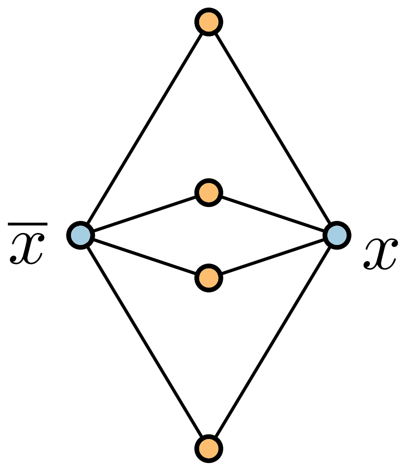
Variable Gadget

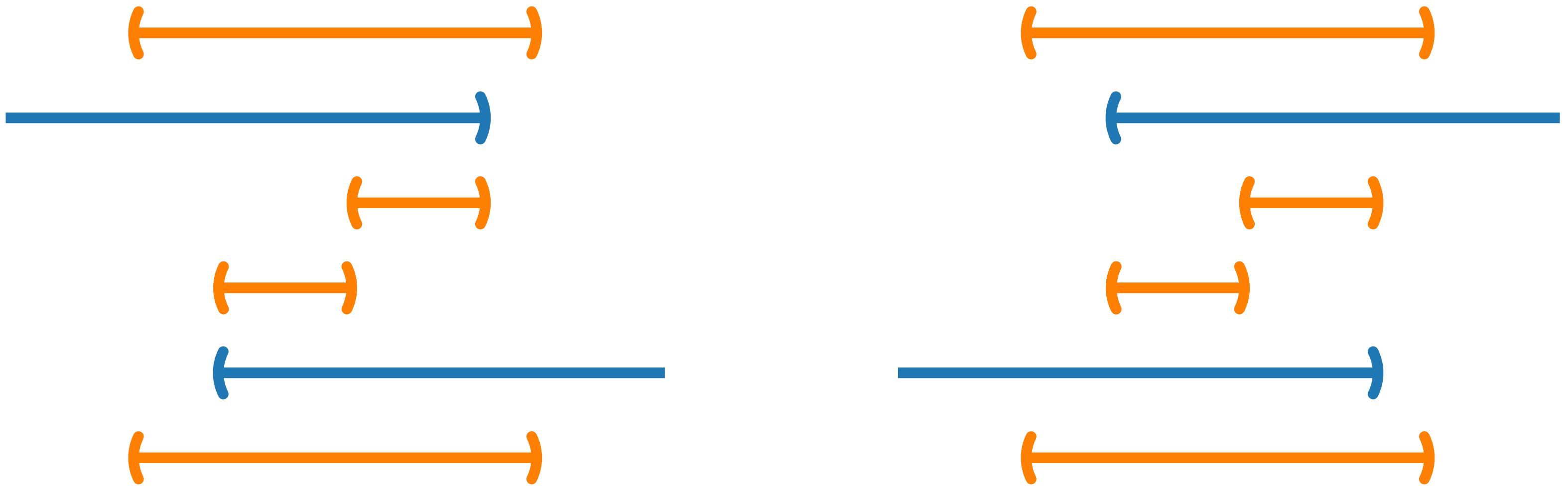


Variable Gadget

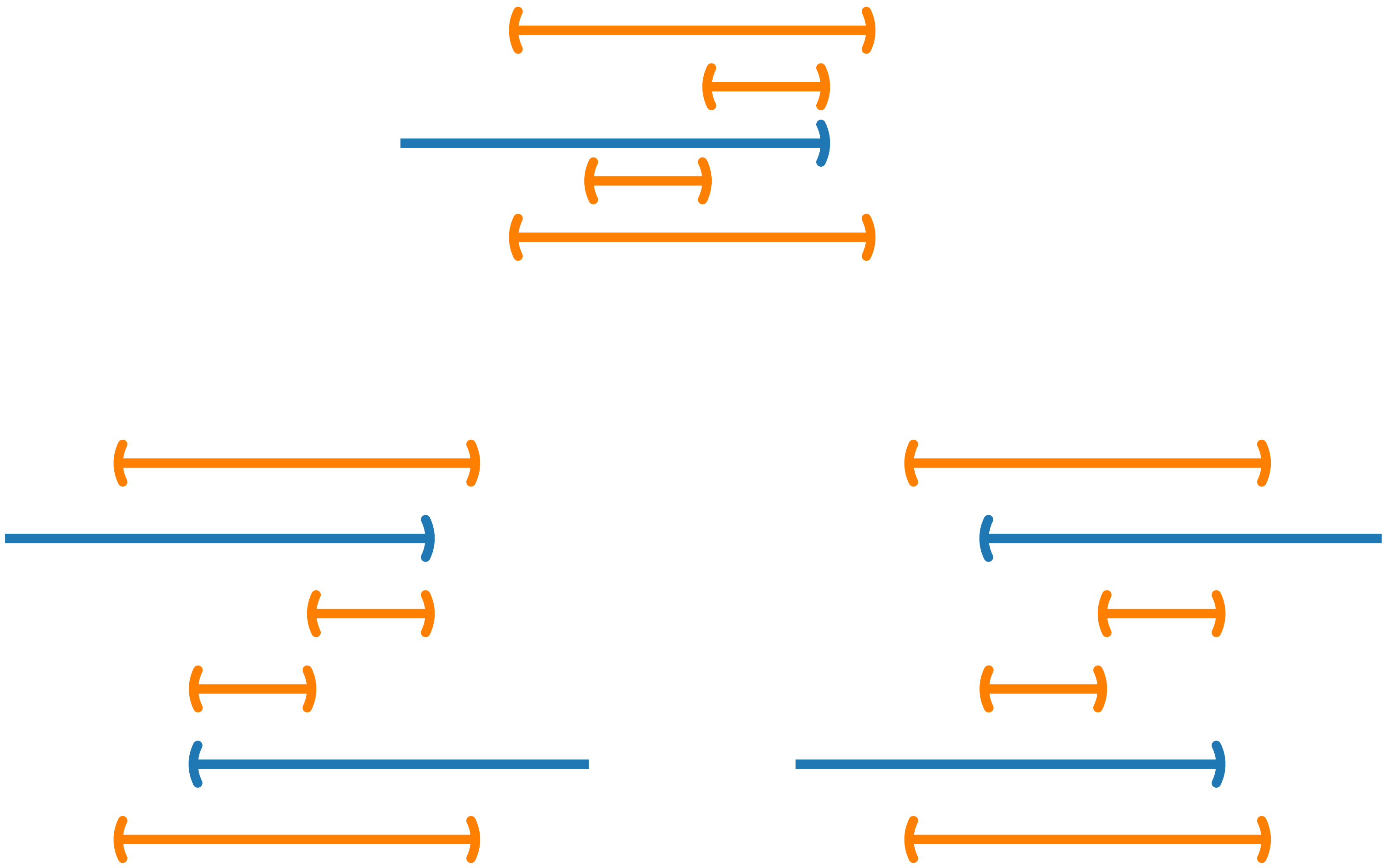


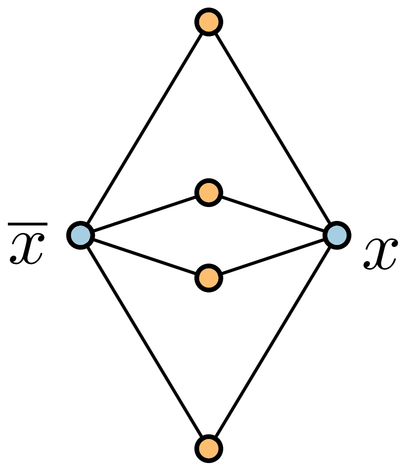
Variable Gadget



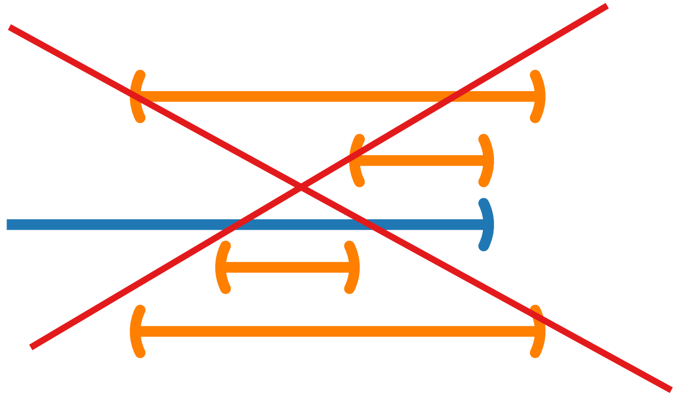


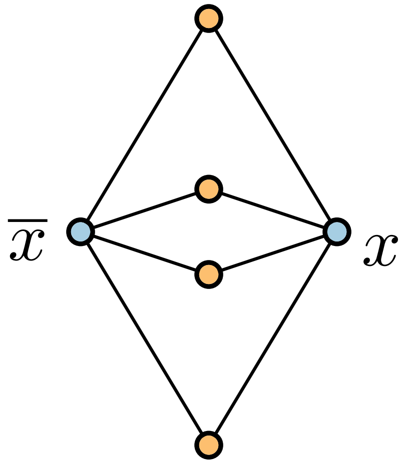
Variable Gadget

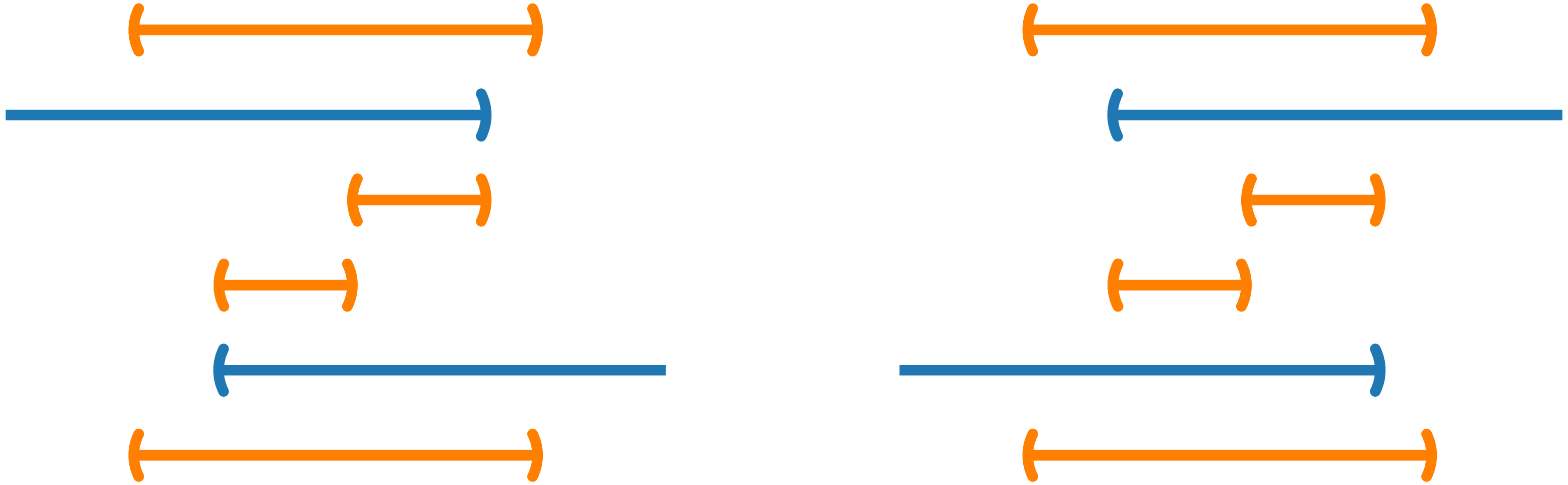




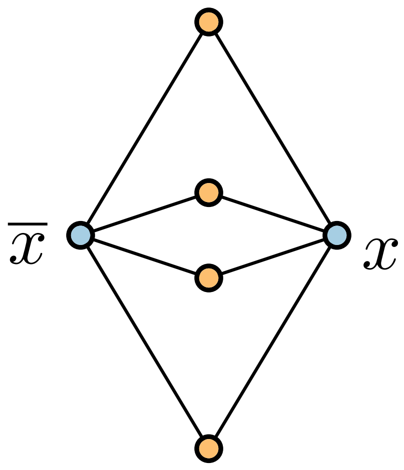
Variable Gadget

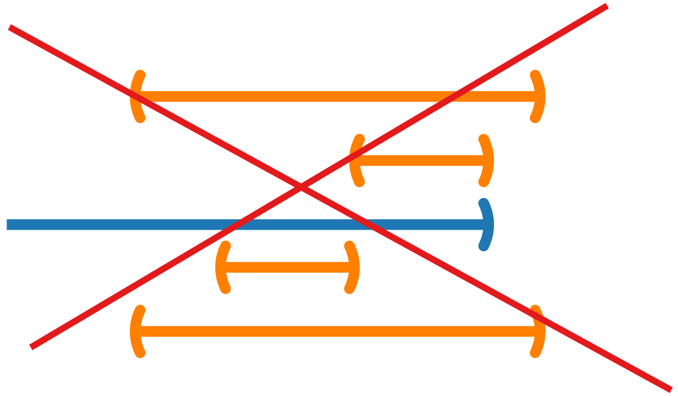


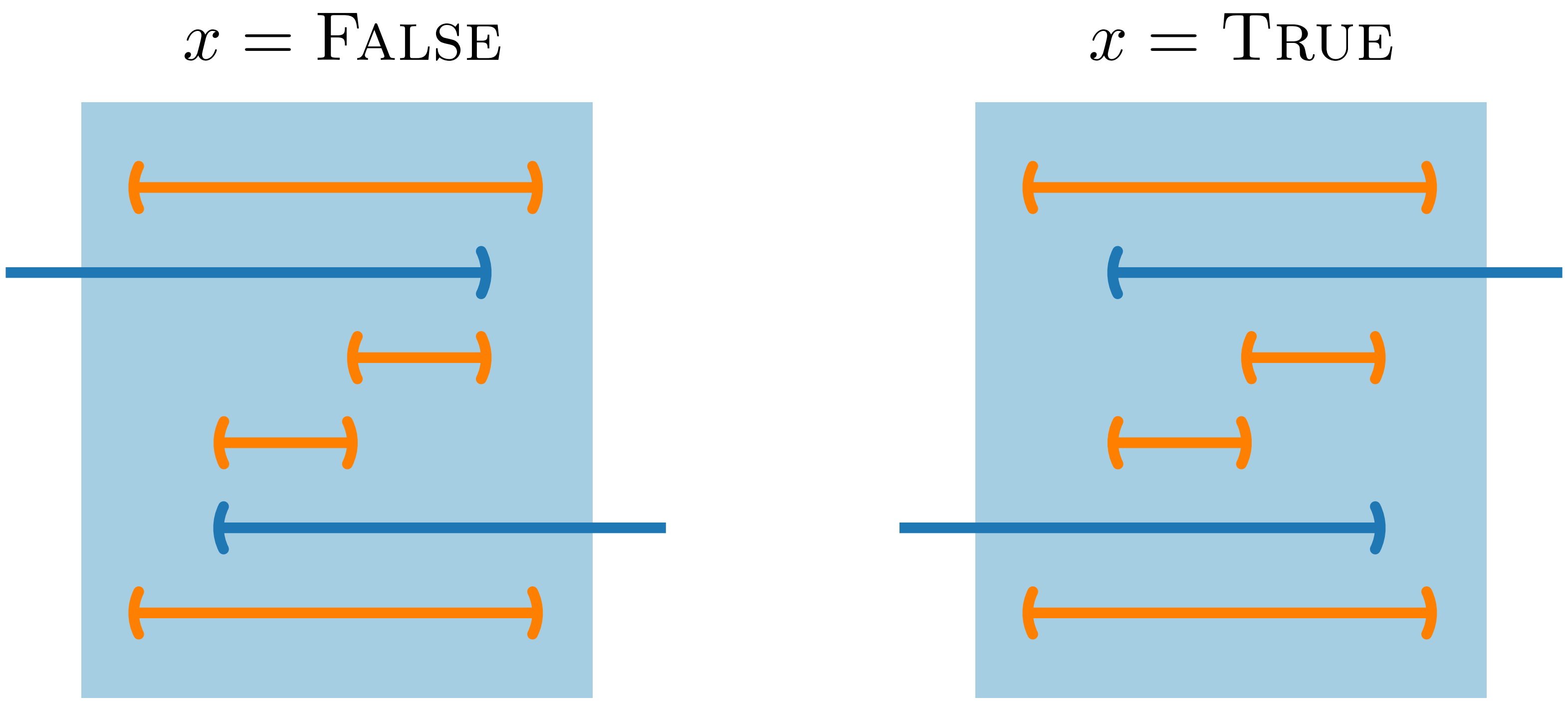




Variable Gadget







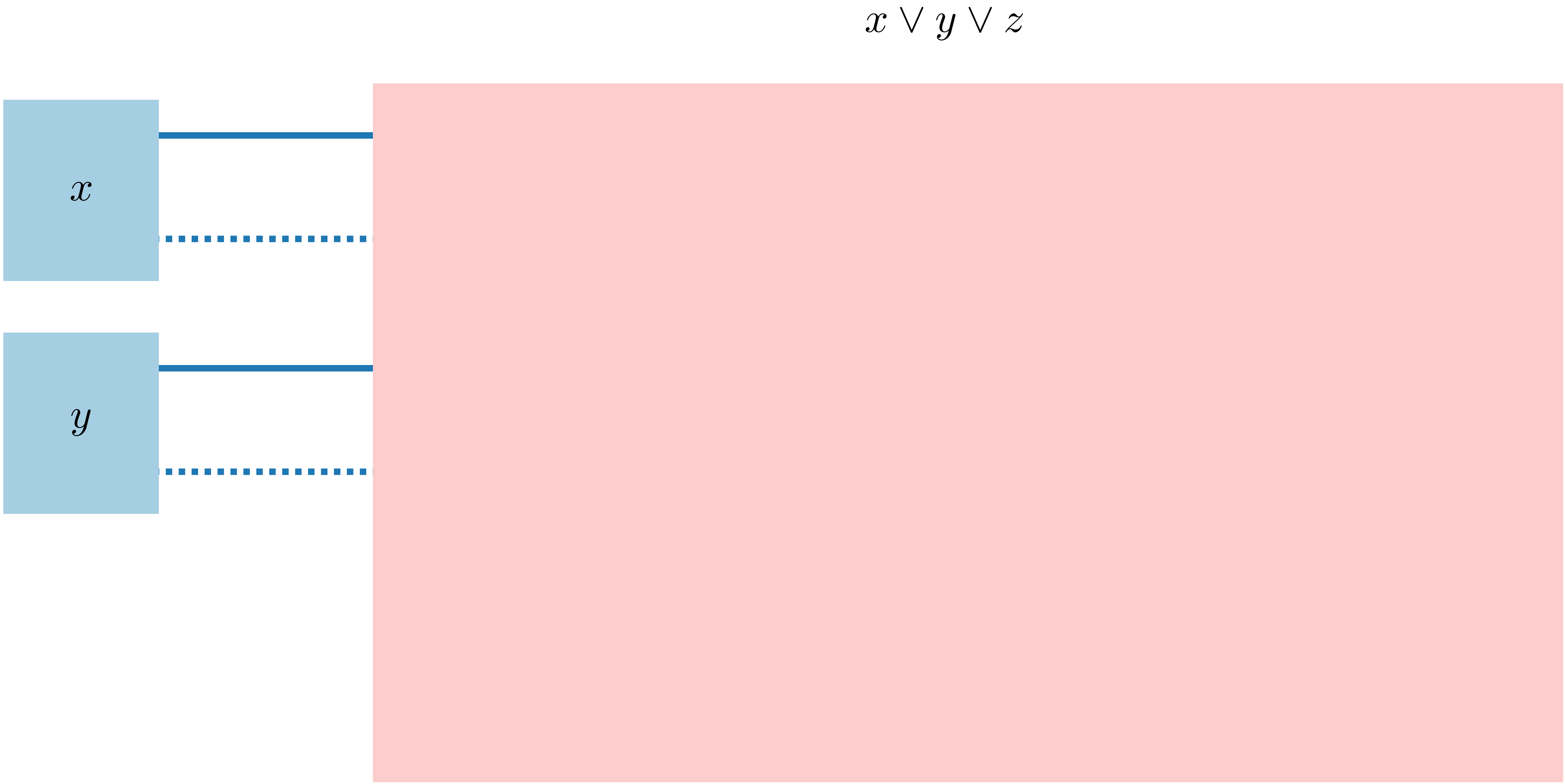
Clause Gadget

$$x \vee y \vee z$$

Clause Gadget

$x \vee y \vee z$

Clause Gadget

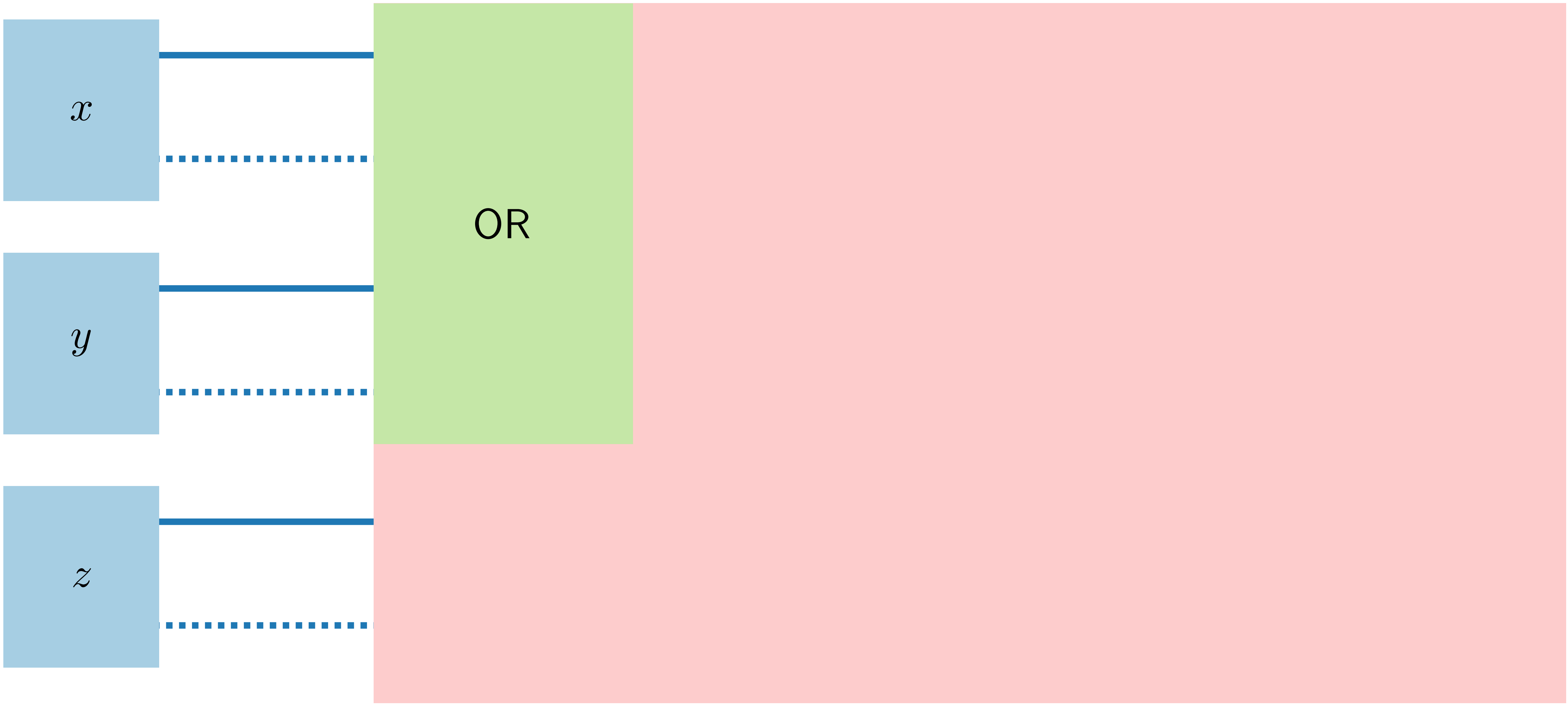


Clause Gadget

$x \vee y \vee z$

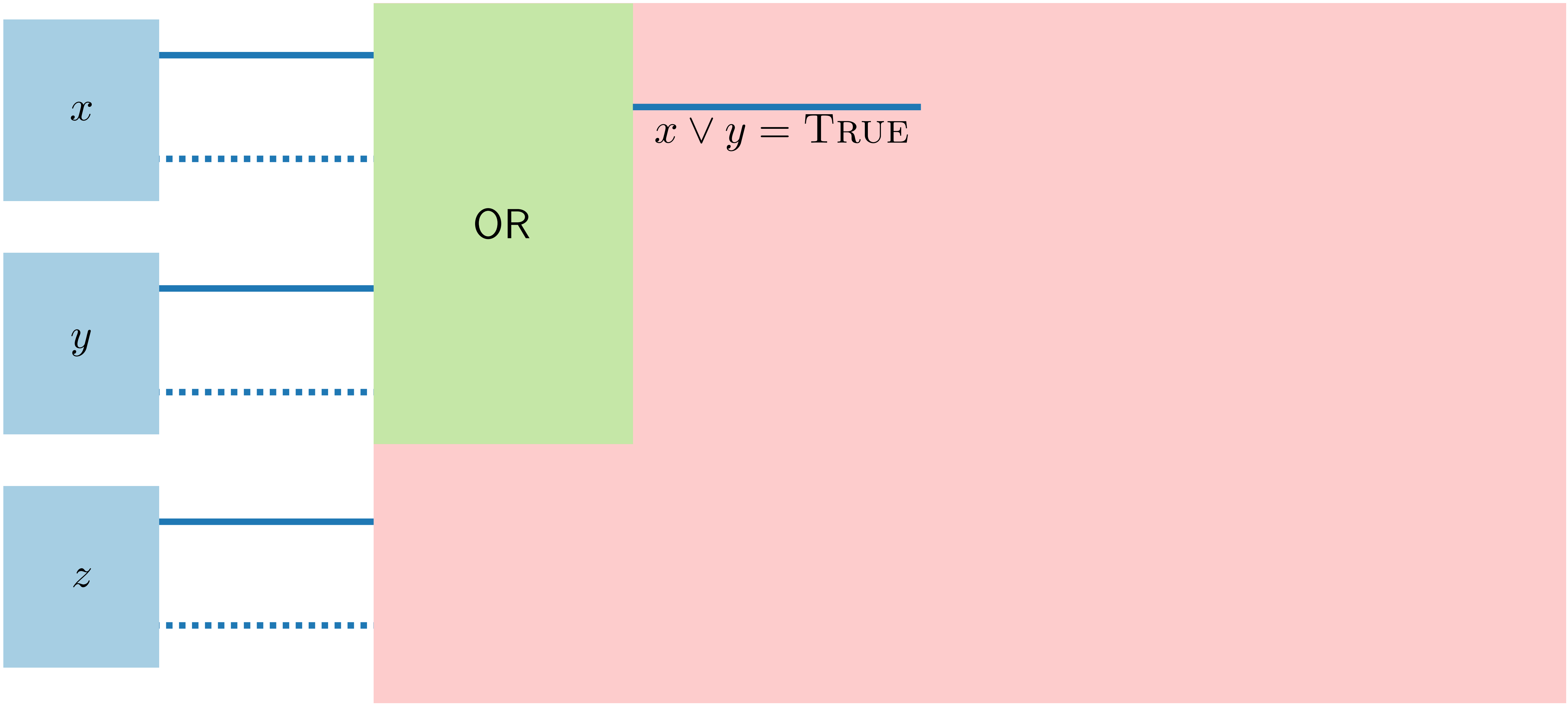
Clause Gadget

$x \vee y \vee z$



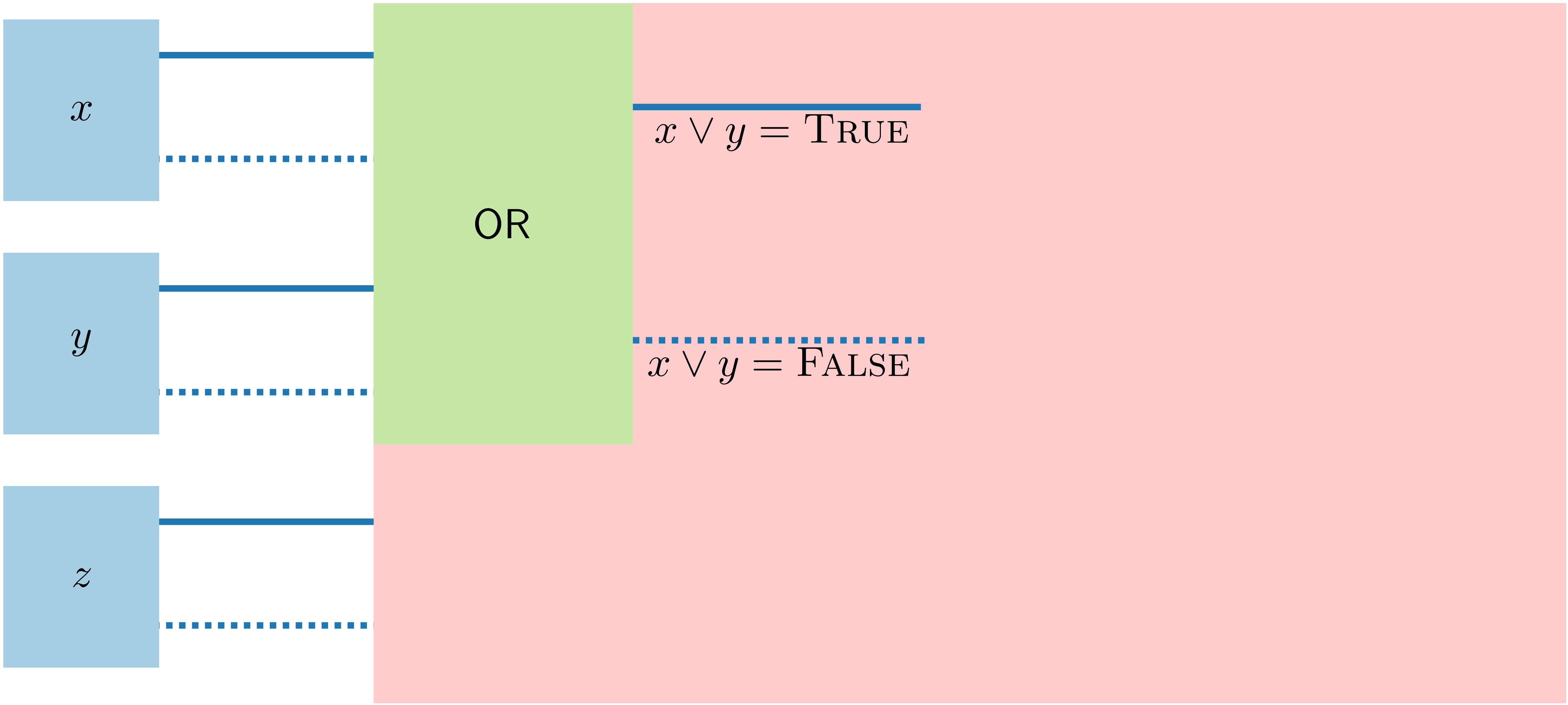
Clause Gadget

$x \vee y \vee z$



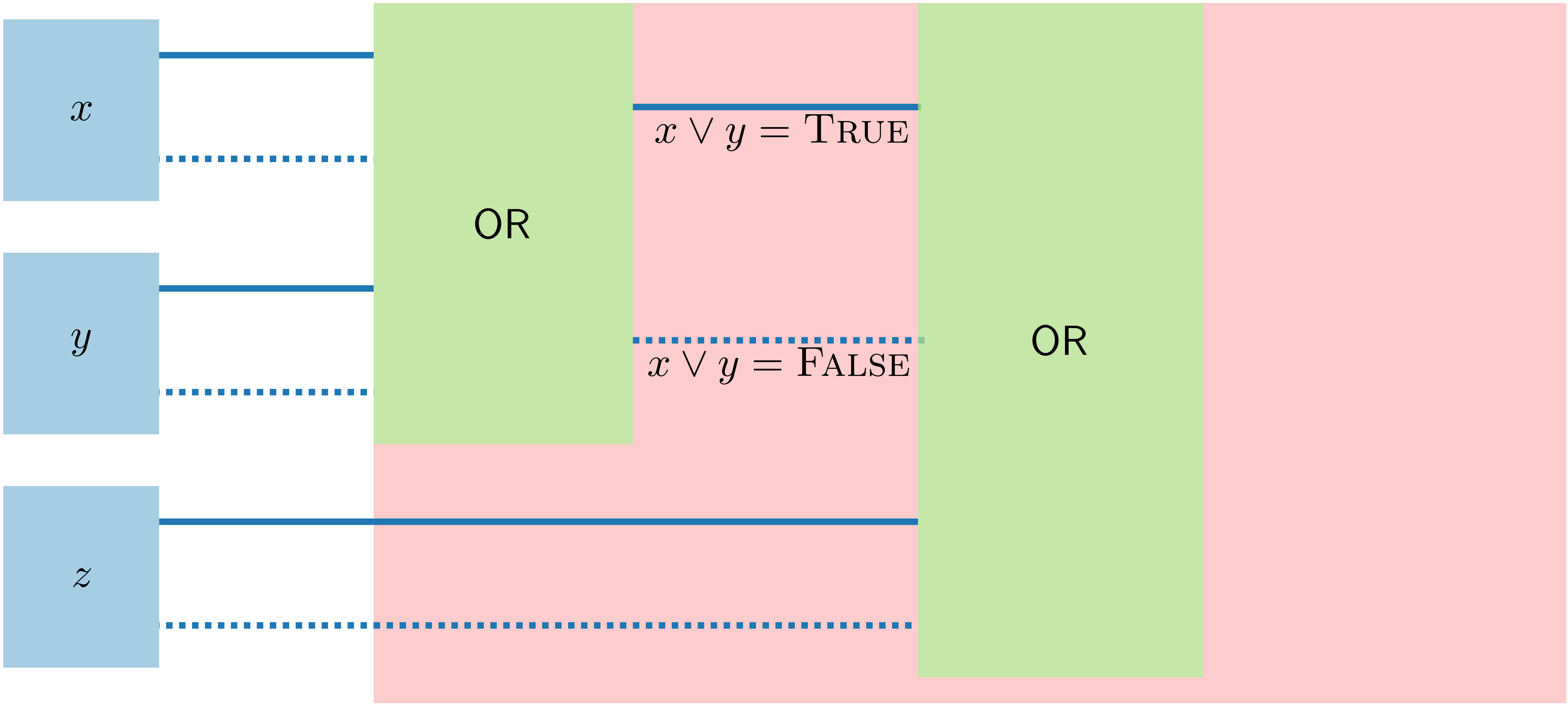
Clause Gadget

$x \lor y \lor z$



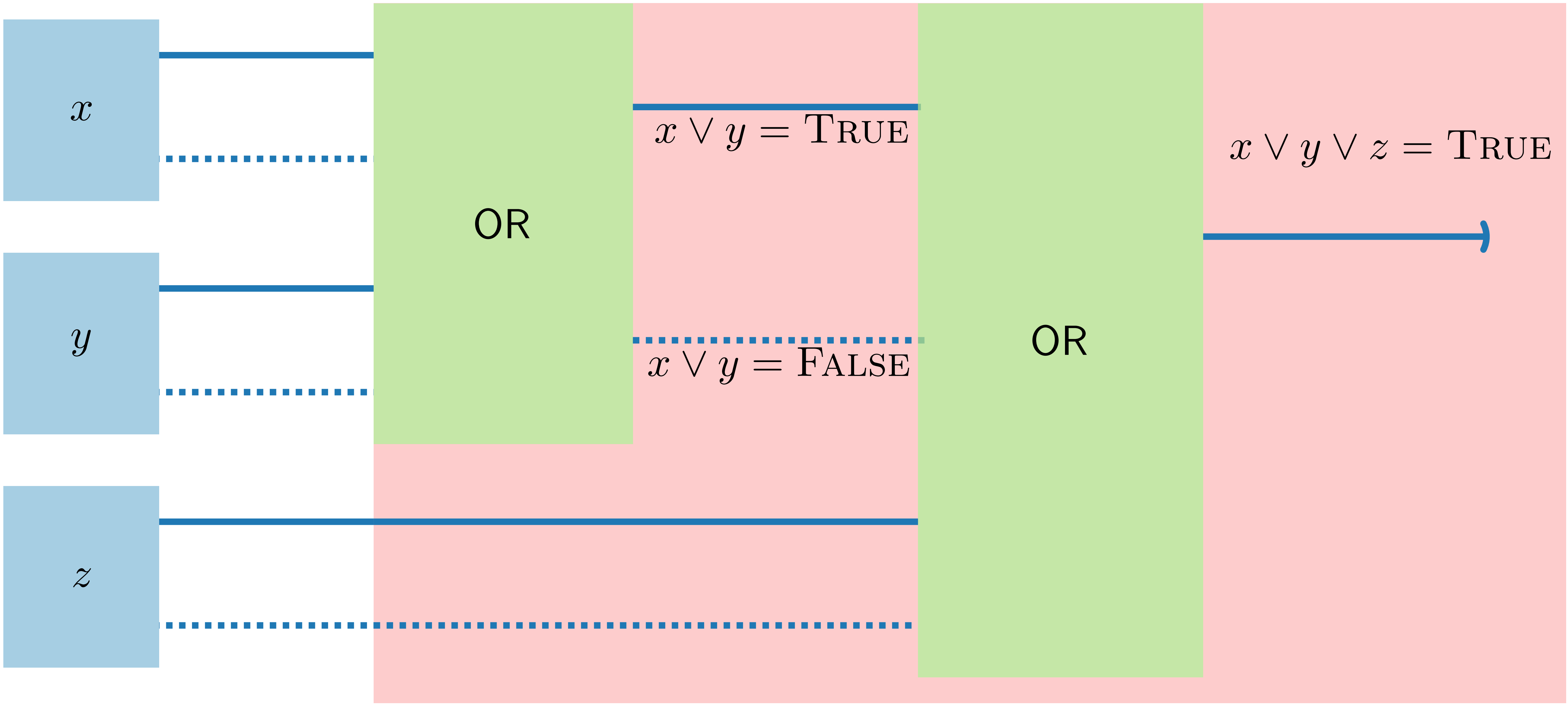
Clause Gadget

$x \lor y \lor z$



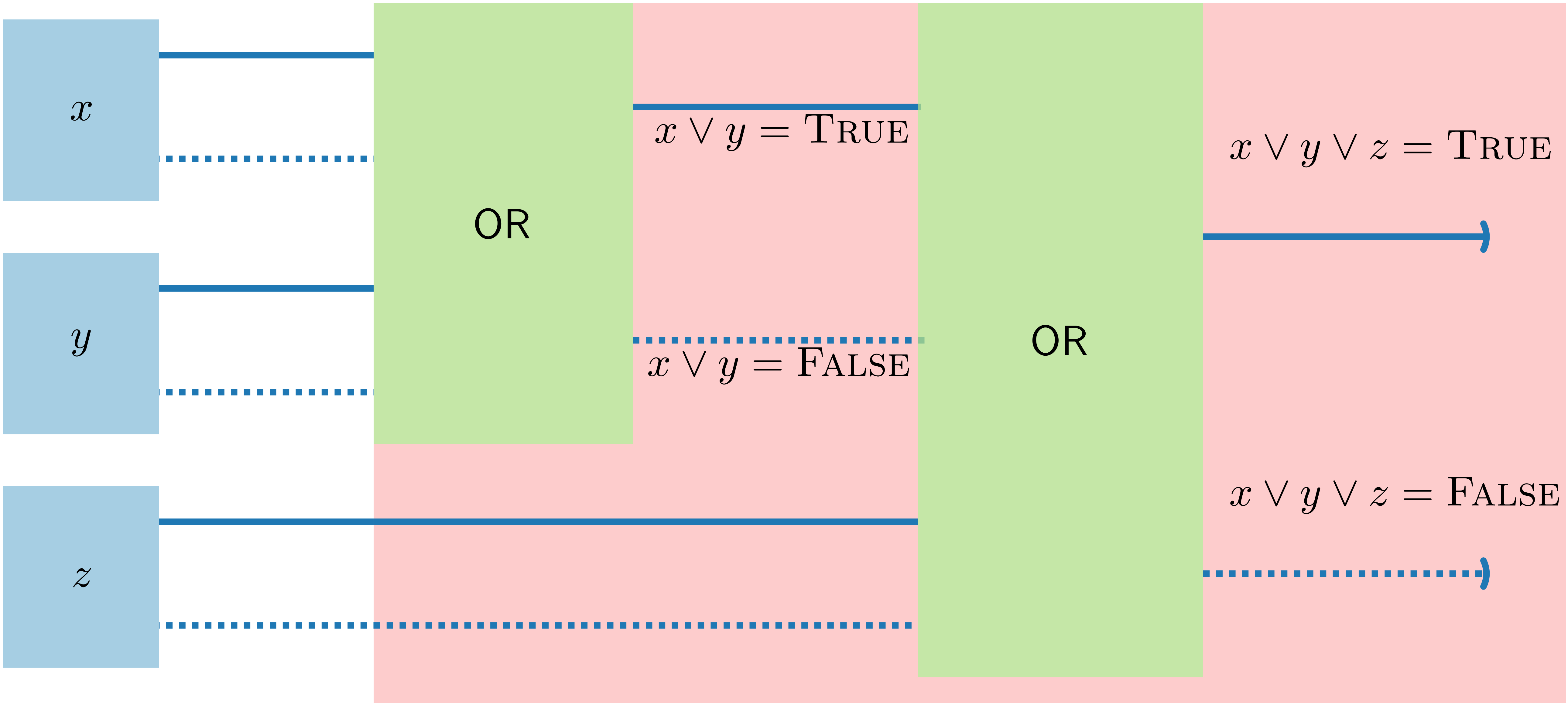
Clause Gadget

$$x \vee y \vee z$$



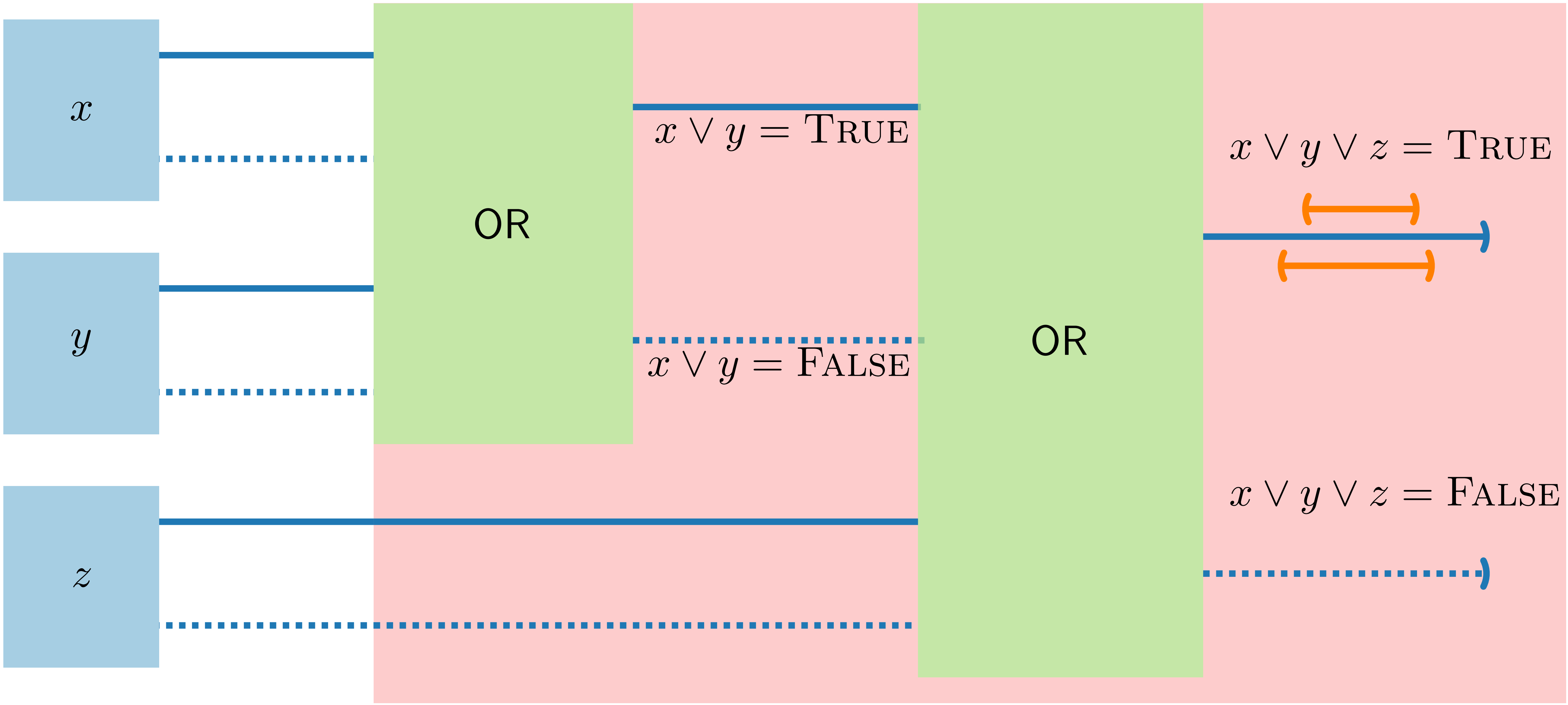
Clause Gadget

$x \vee y \vee z$

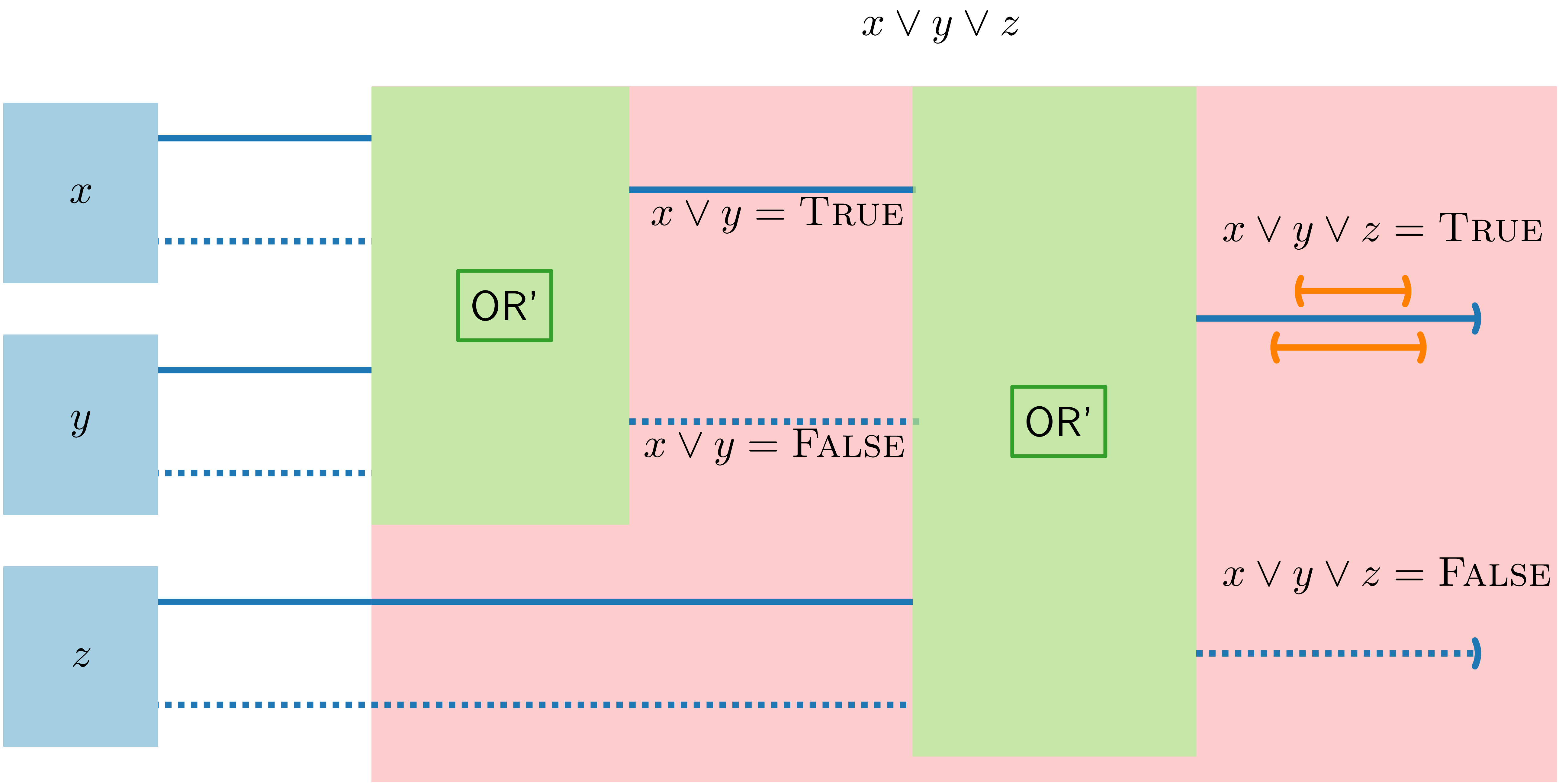


Clause Gadget

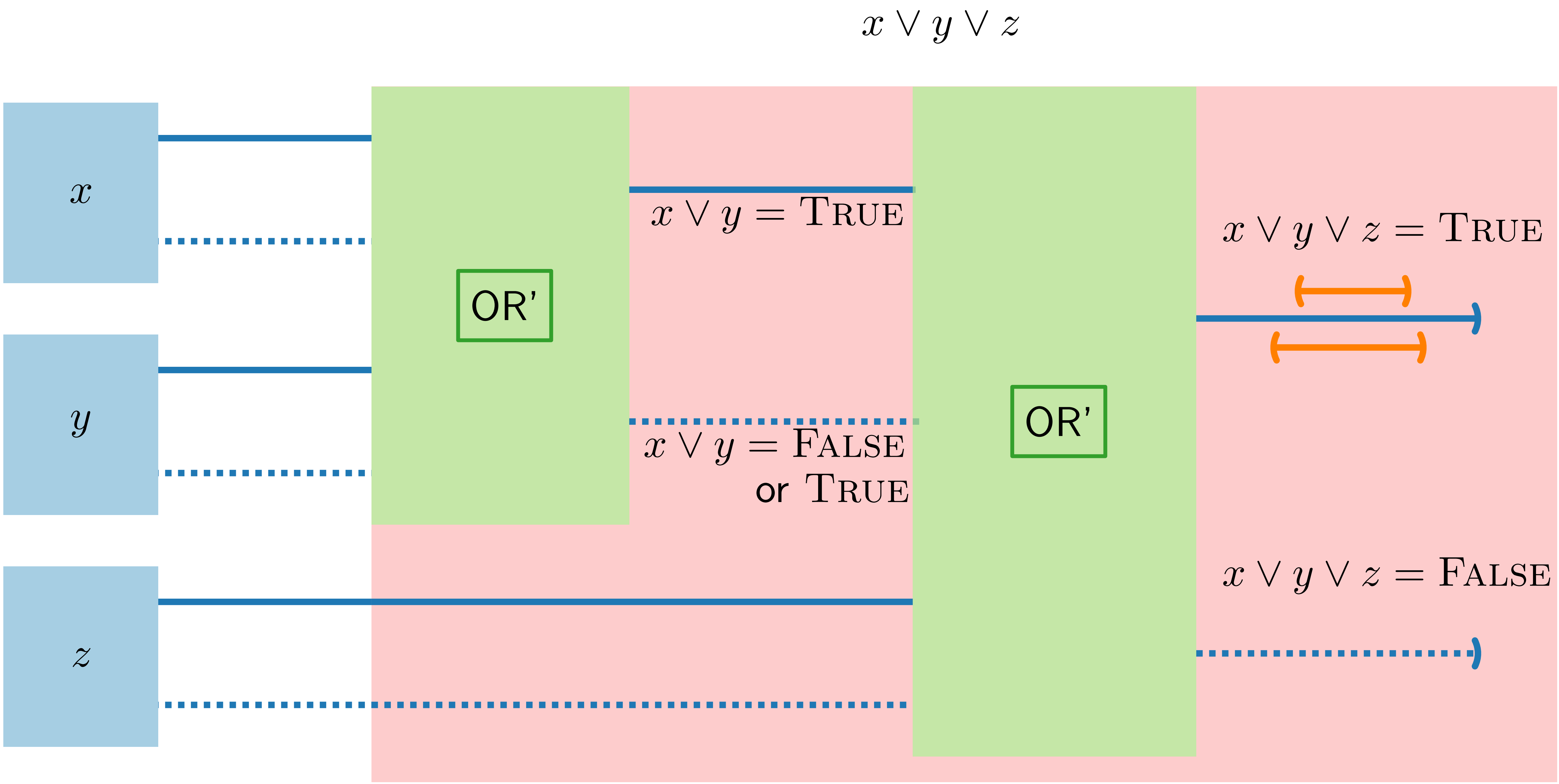
$x \vee y \vee z$



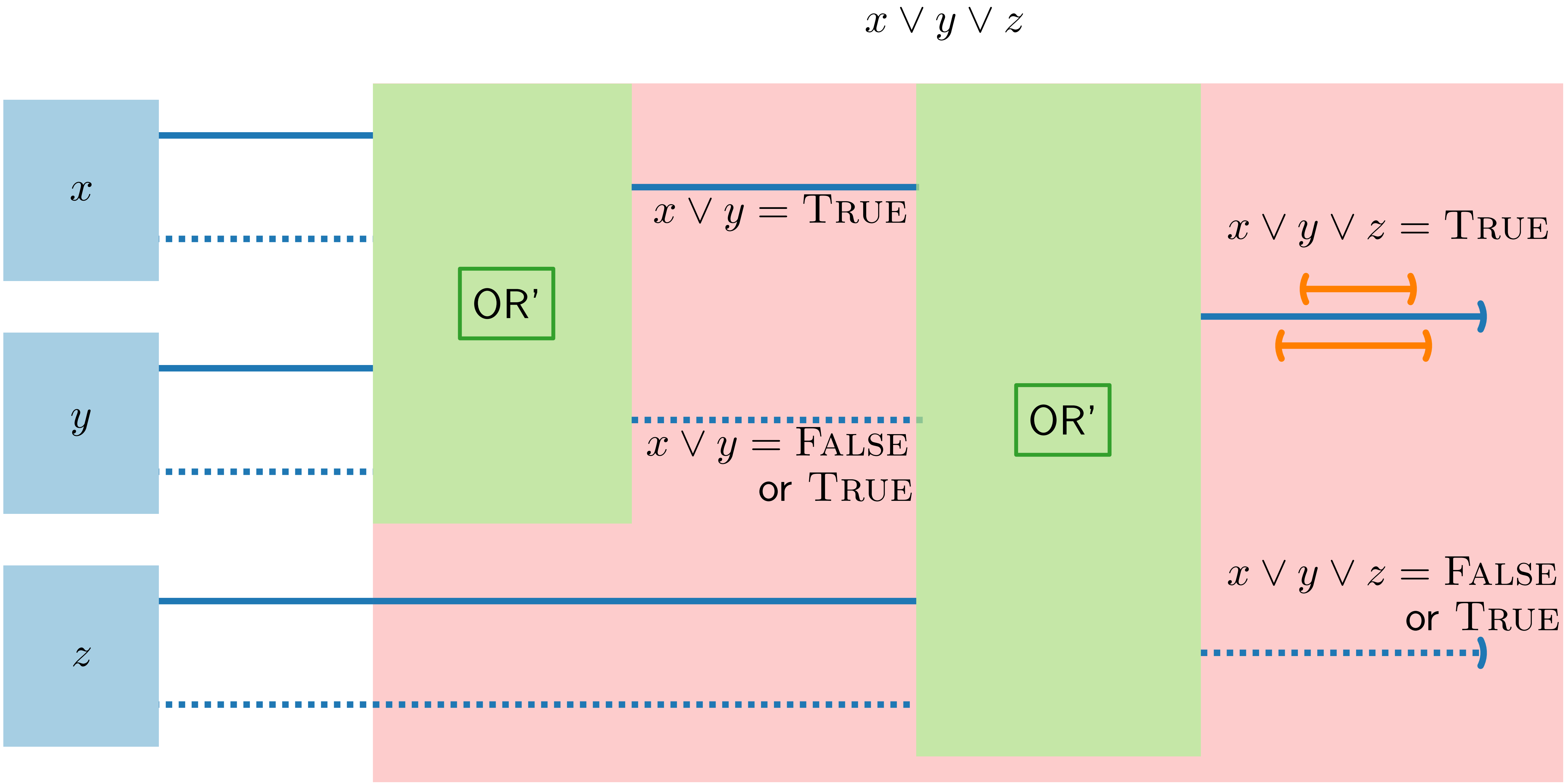
Clause Gadget



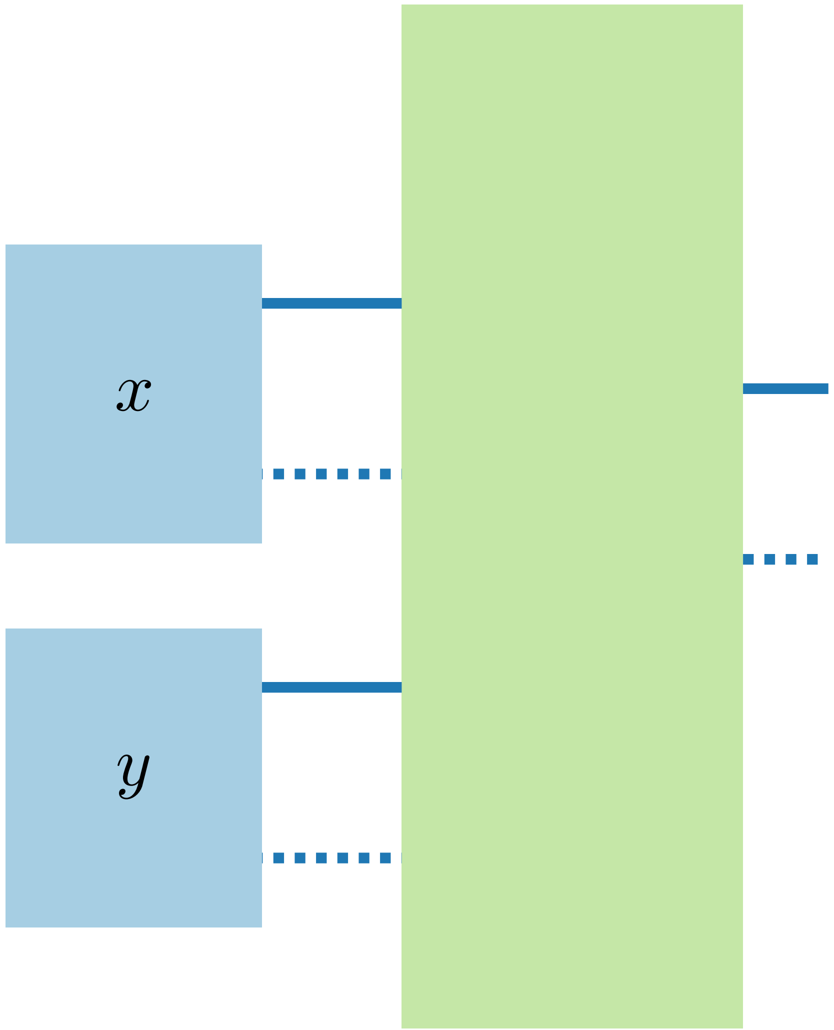
Clause Gadget



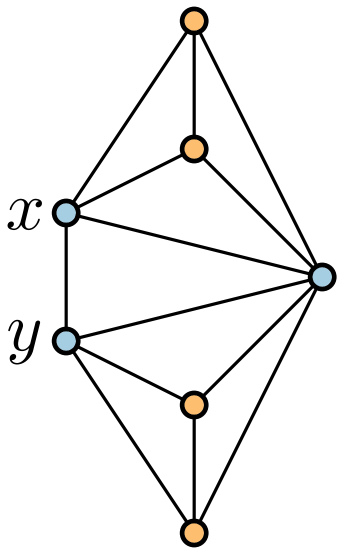
Clause Gadget

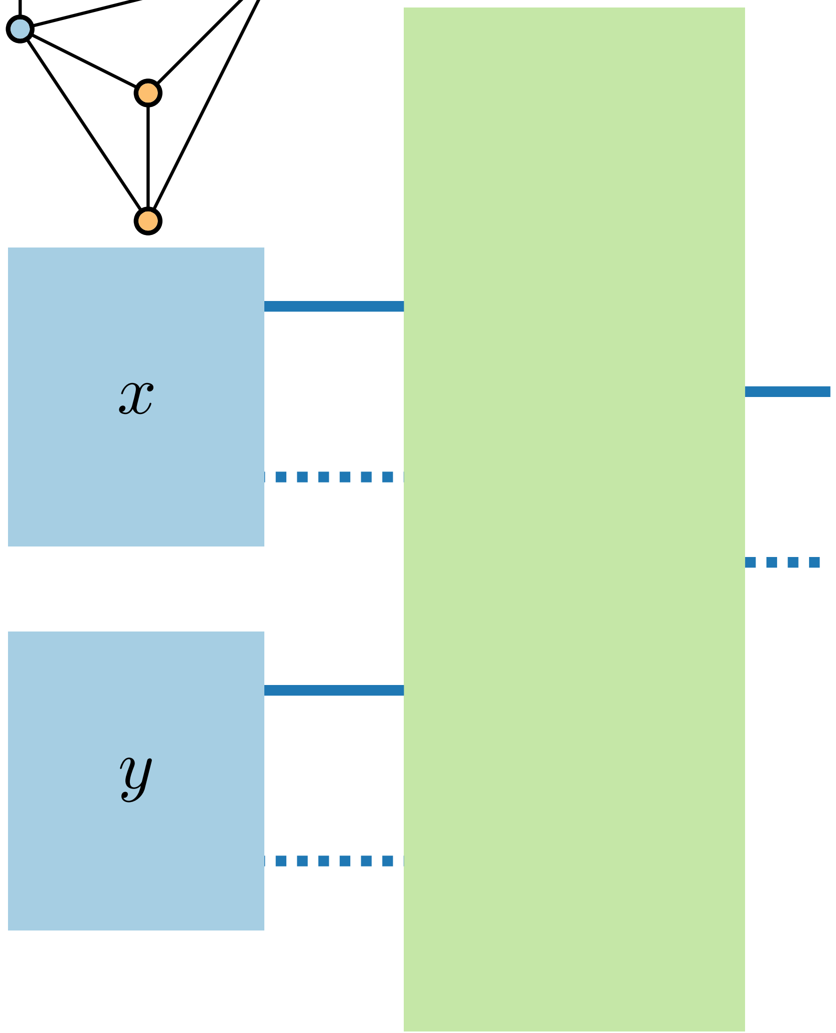


OR' Gadget

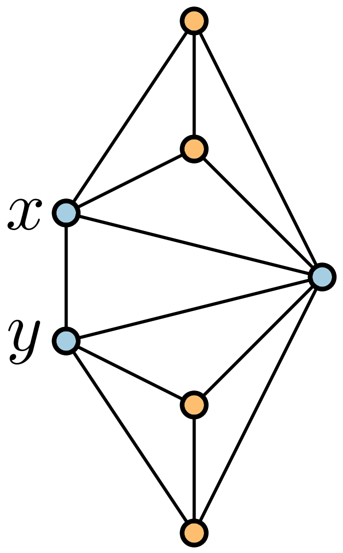


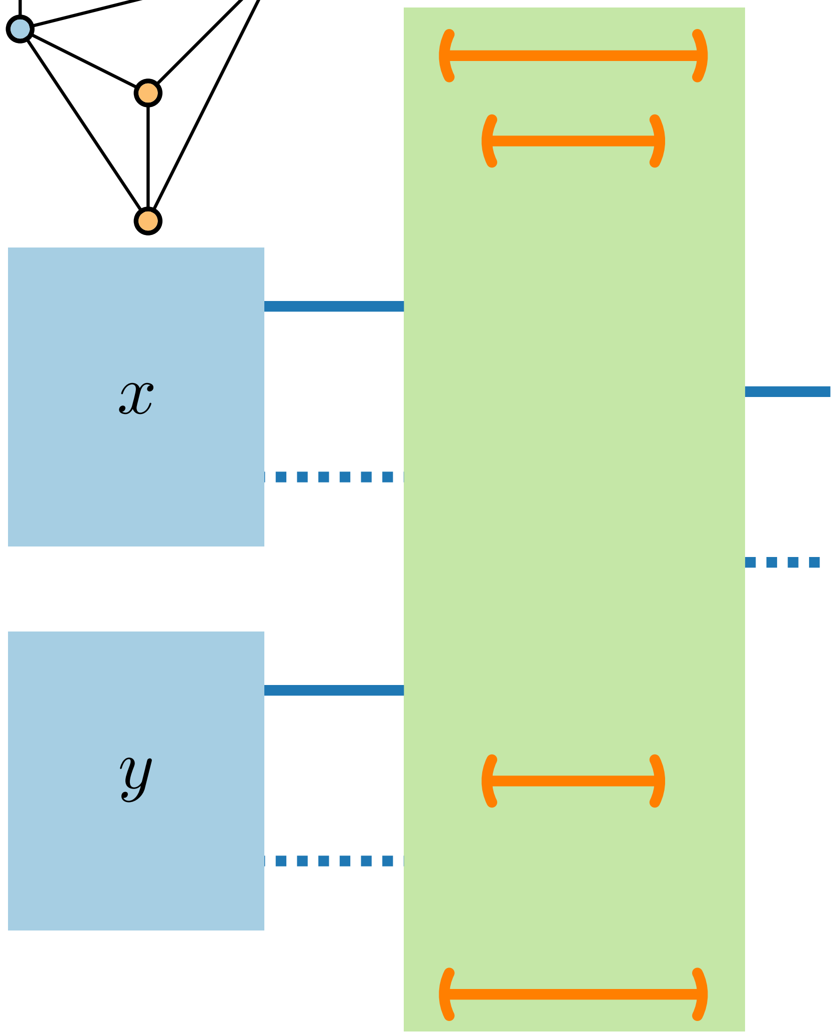
OR' Gadget



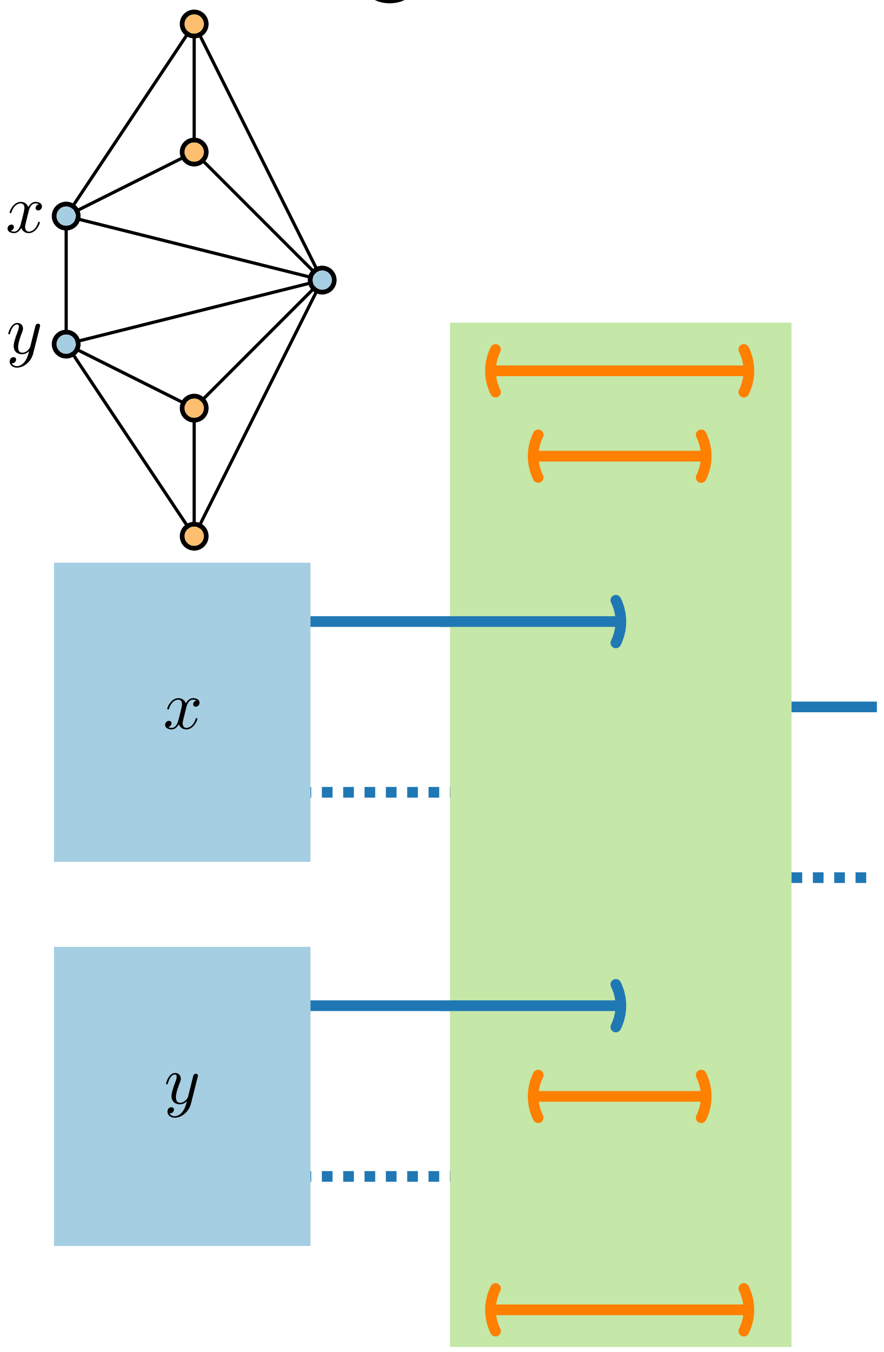


OR' Gadget

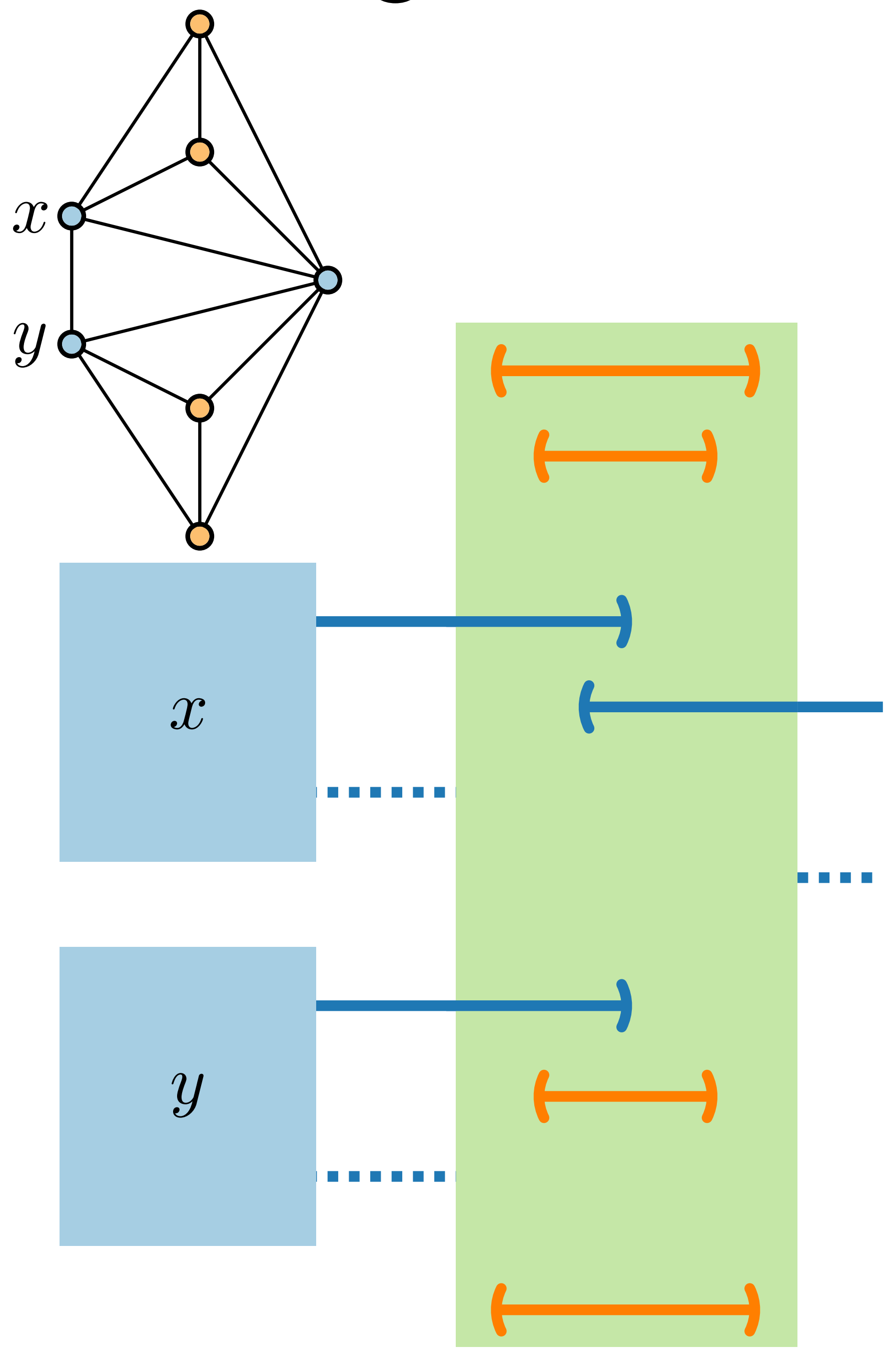




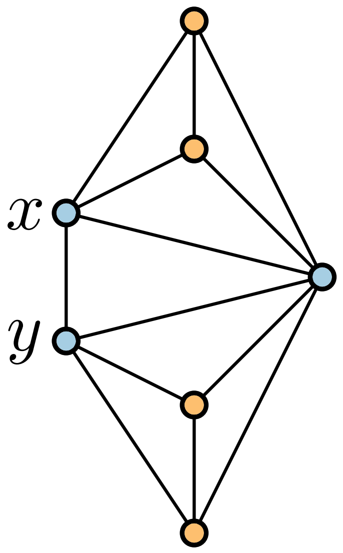
OR' Gadget

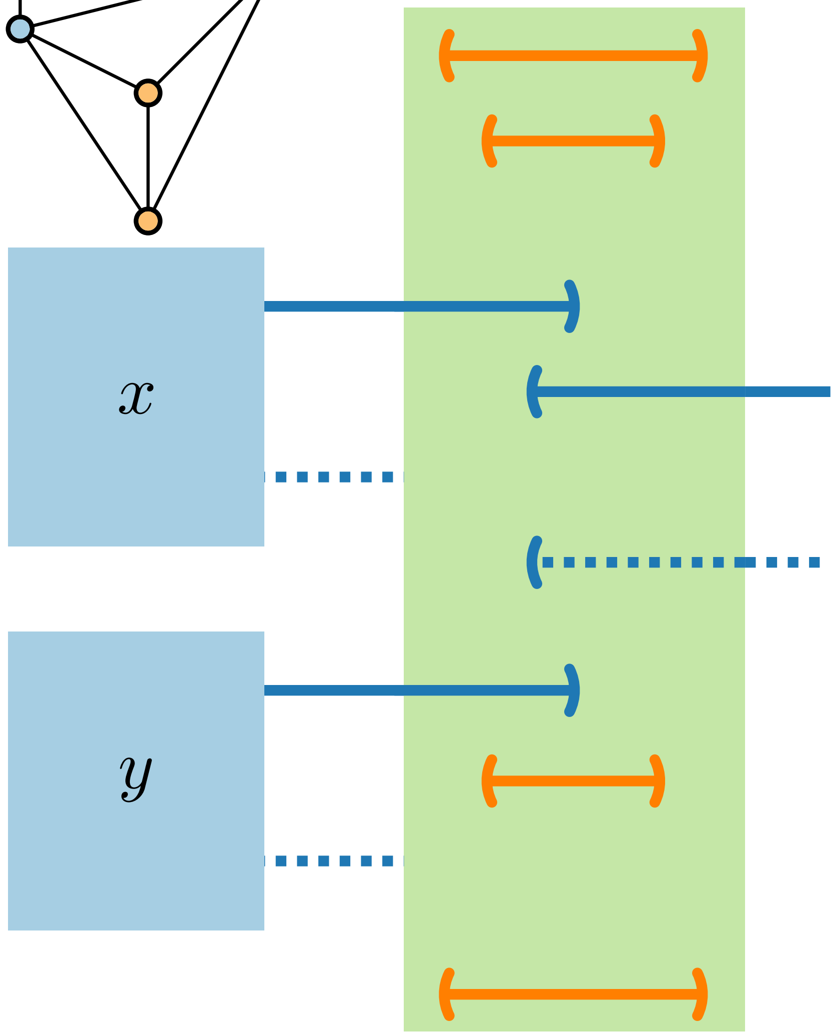


OR' Gadget

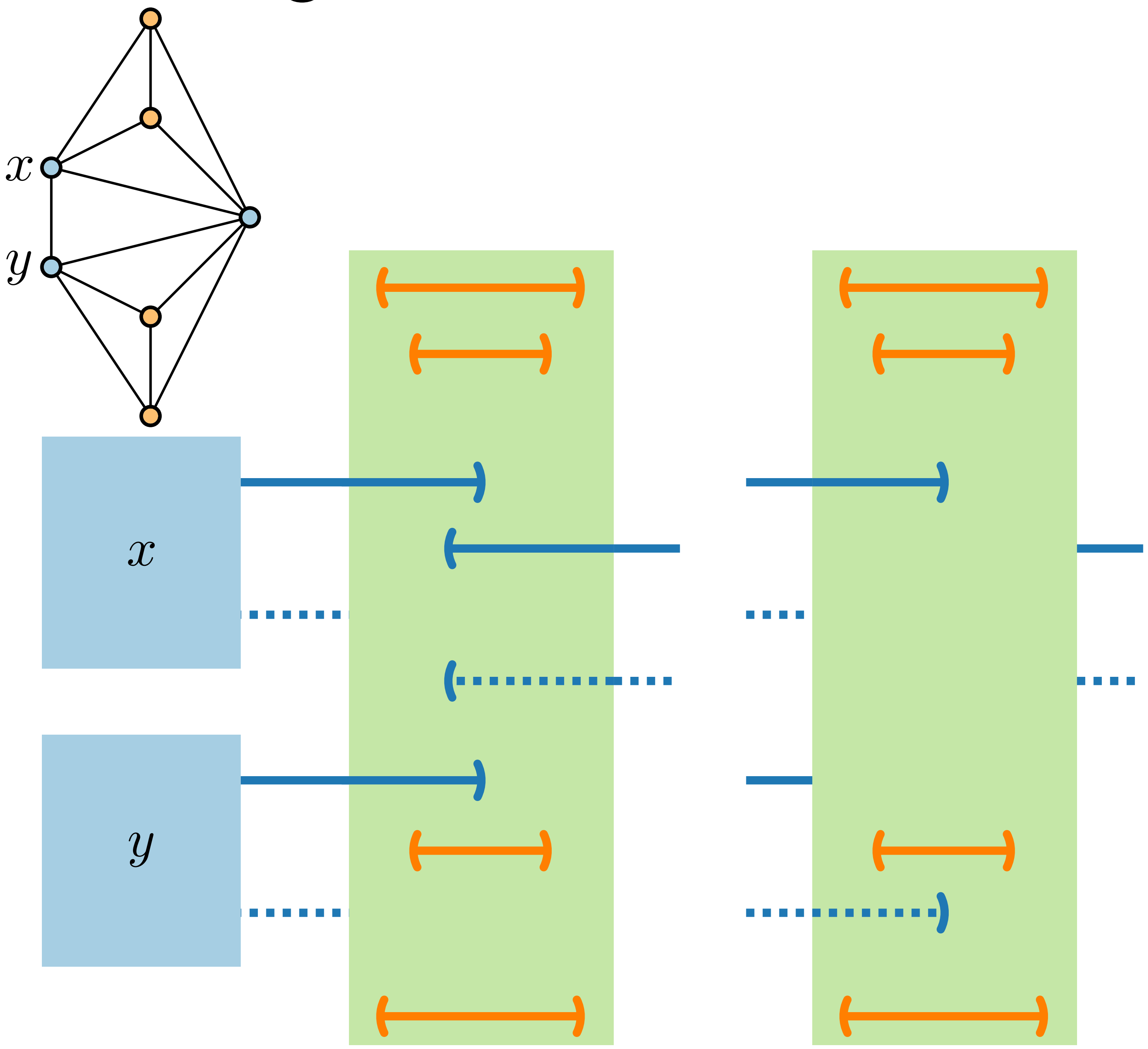


OR' Gadget

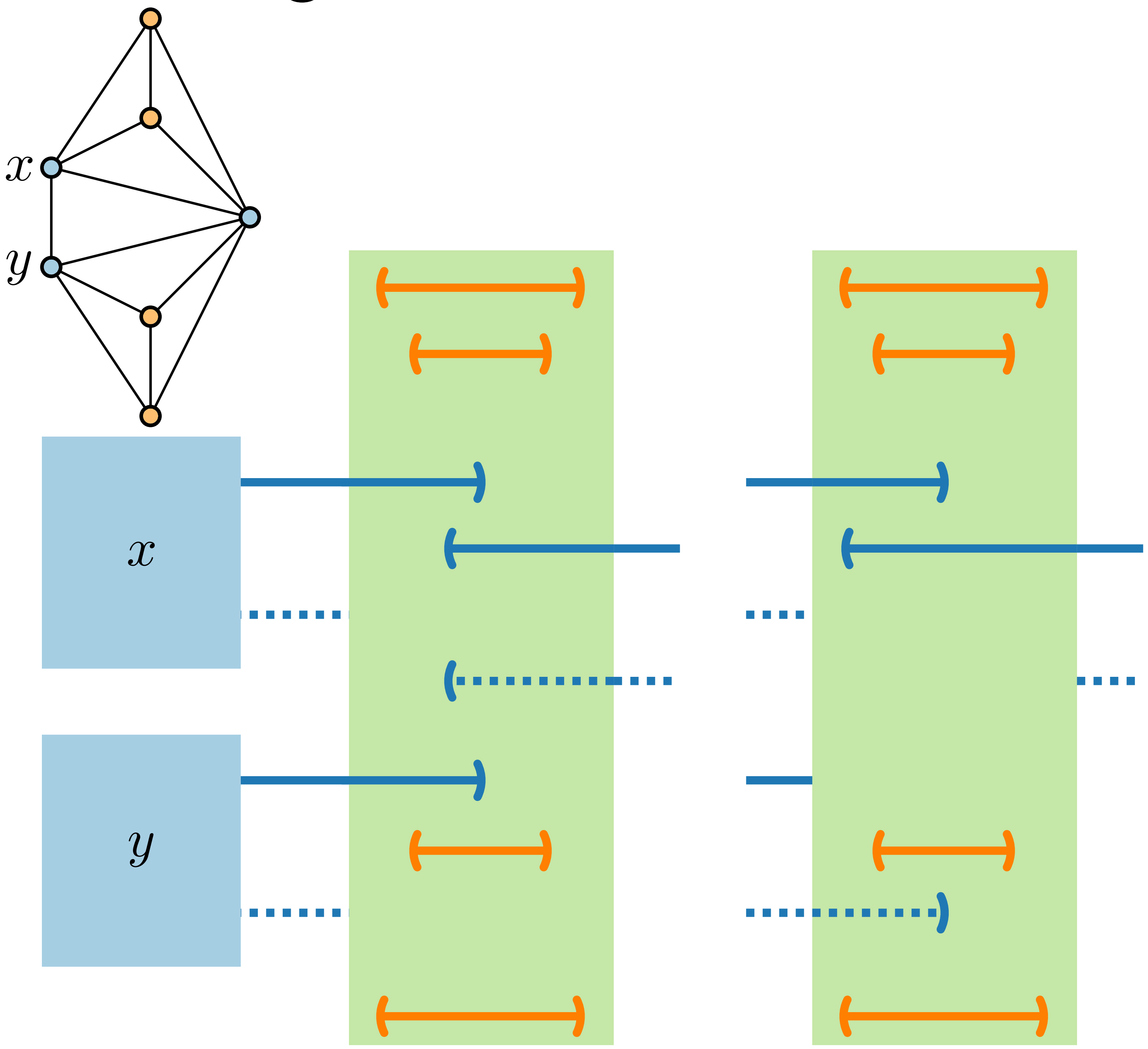




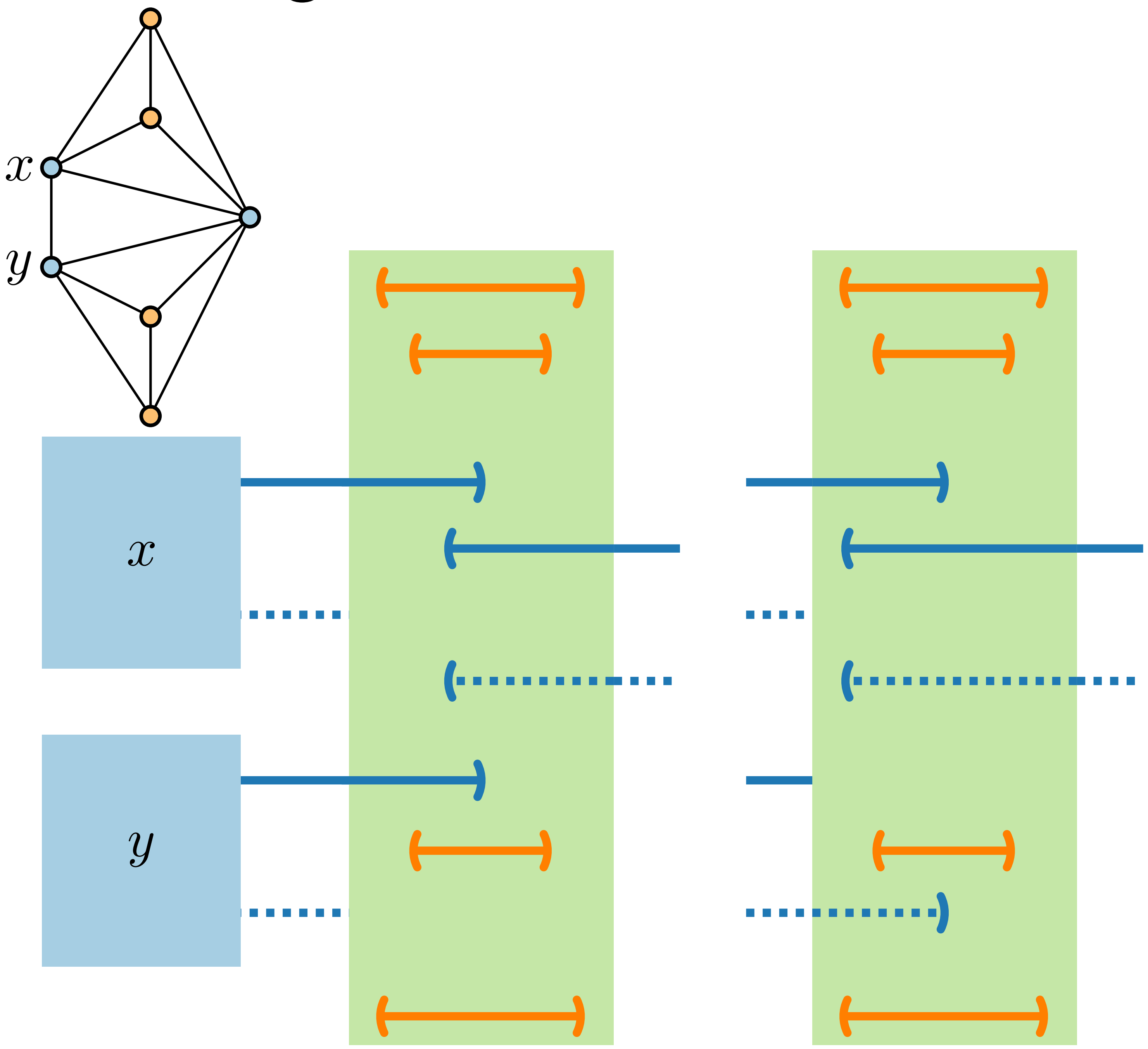
OR' Gadget



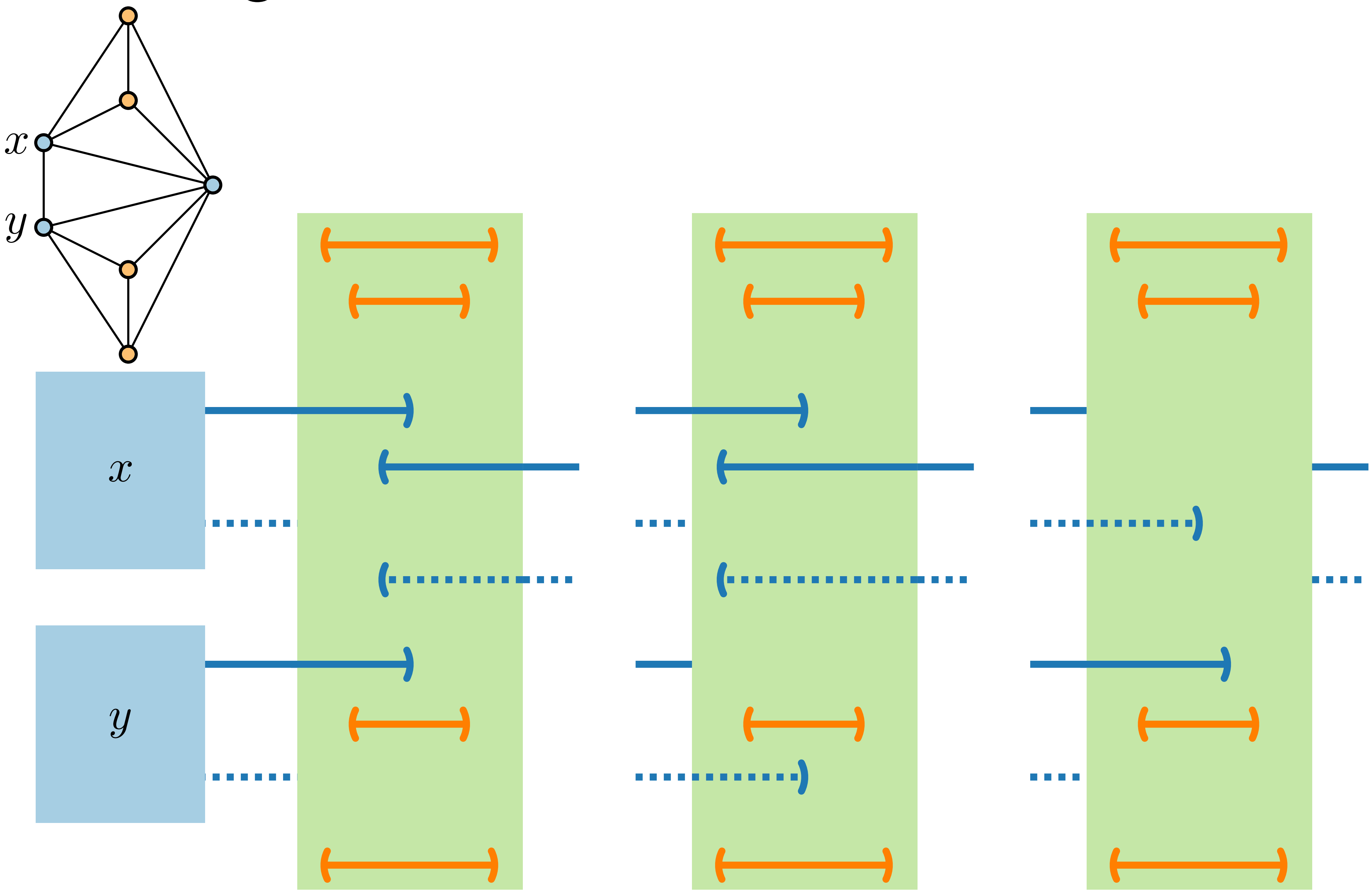
OR' Gadget



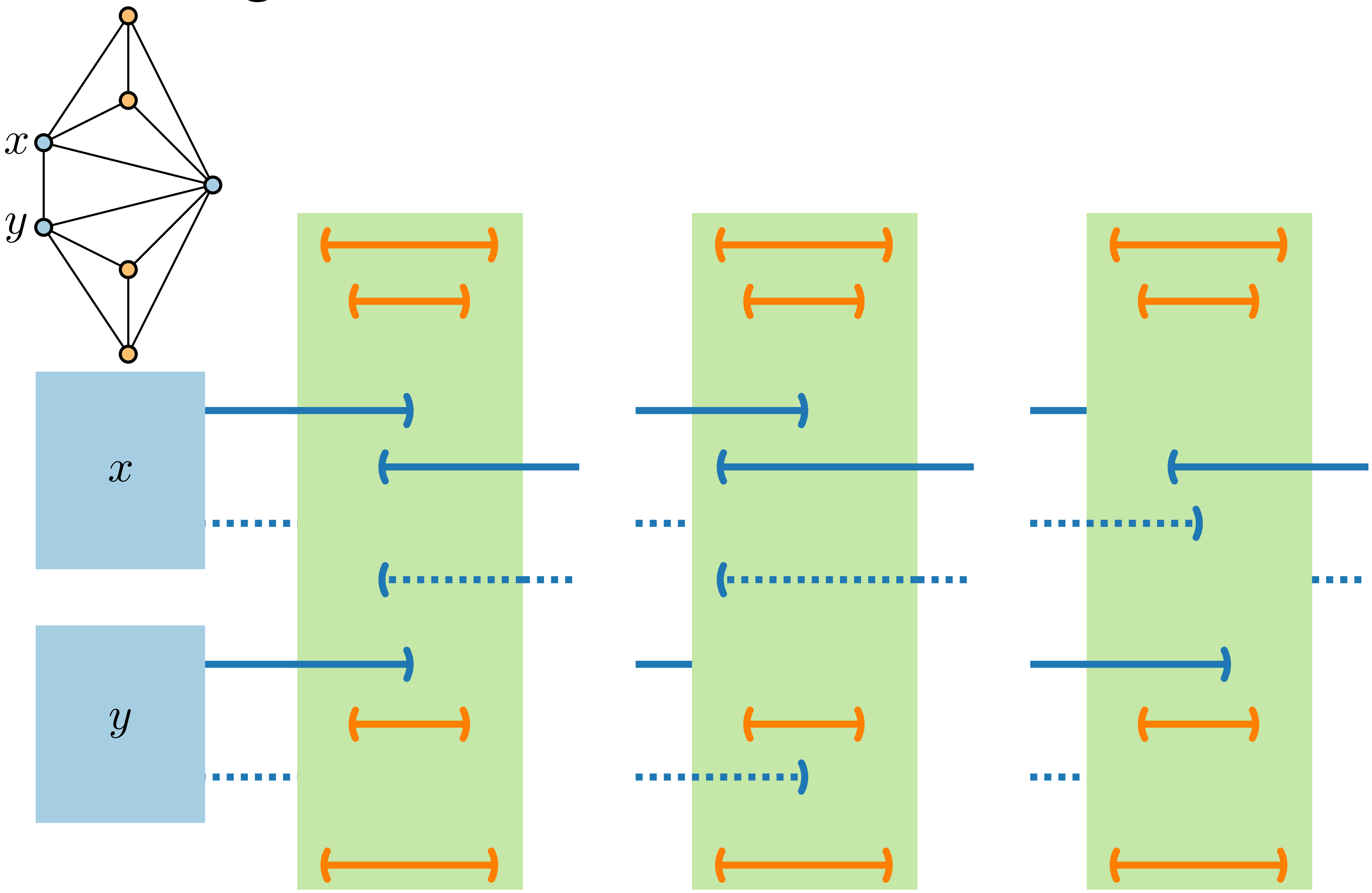
OR' Gadget



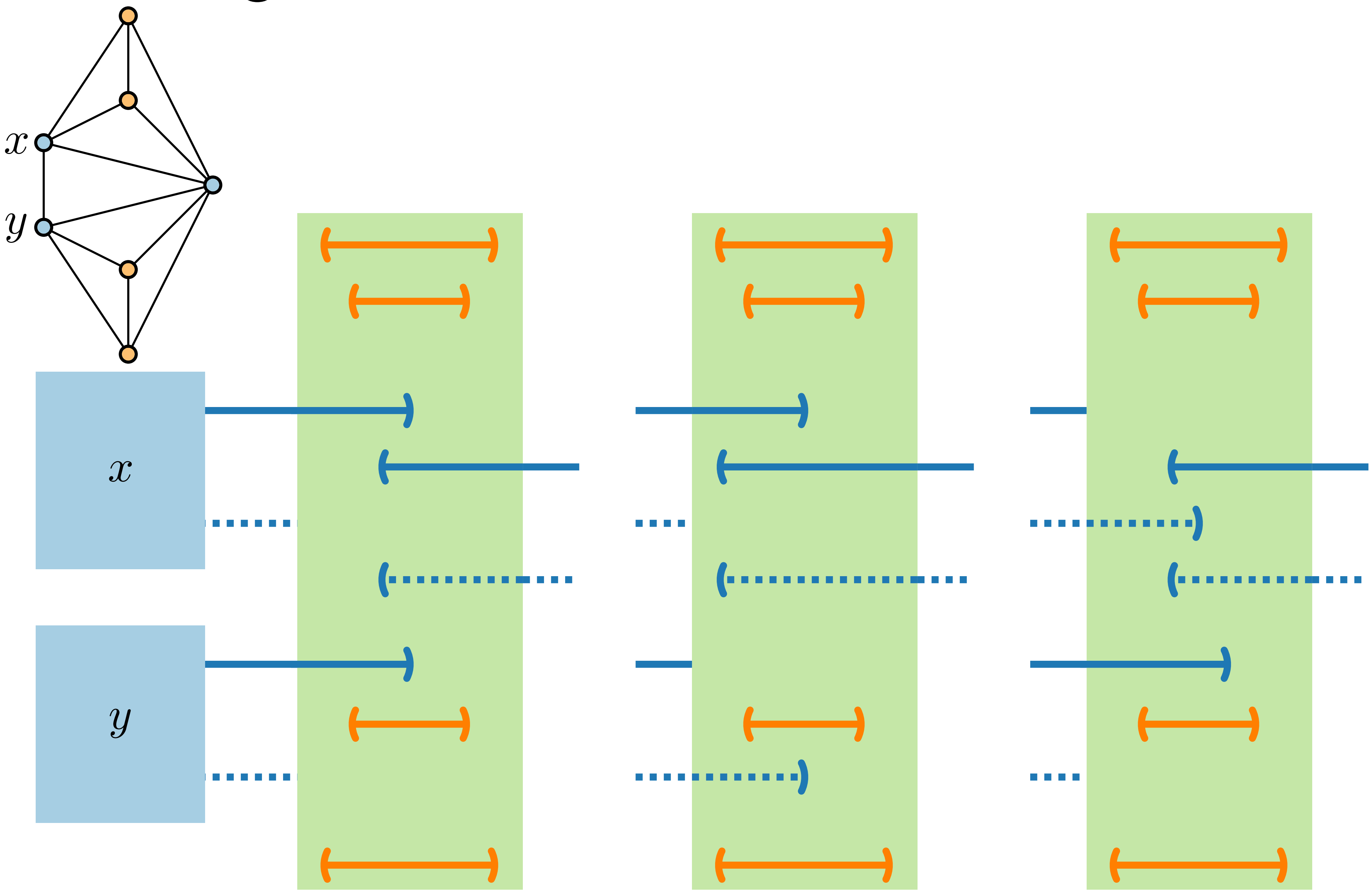
OR' Gadget



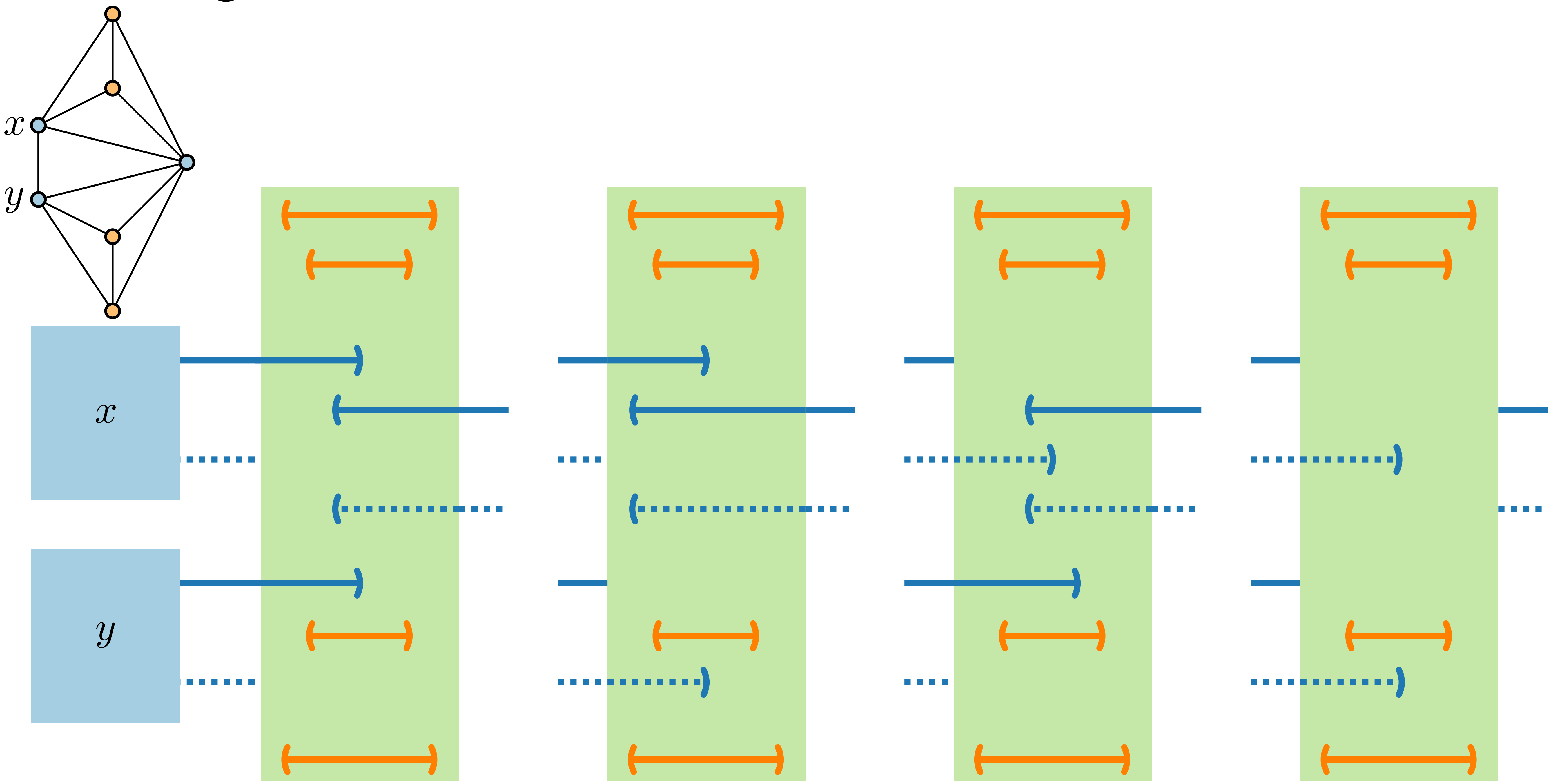
OR' Gadget



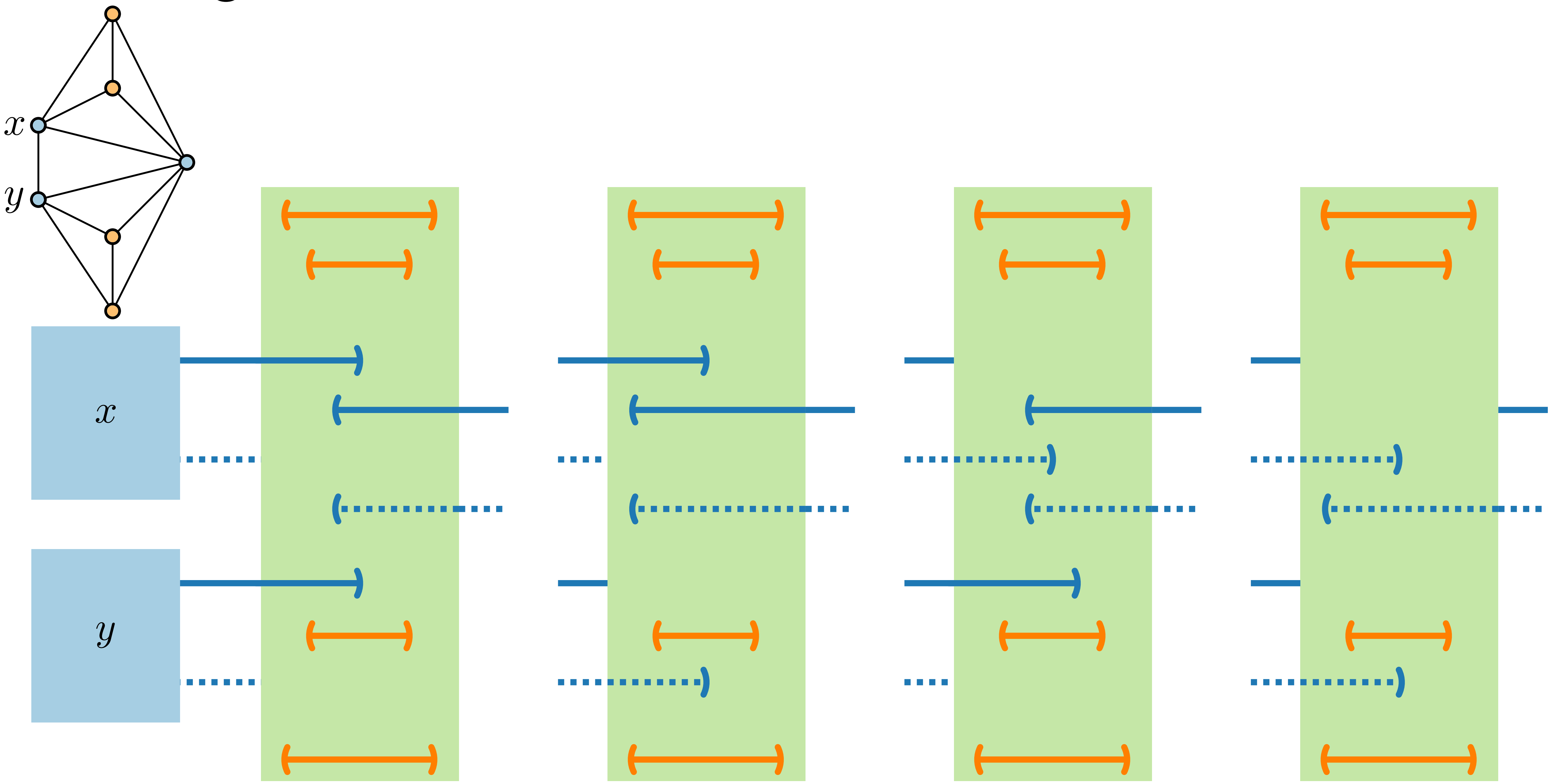
OR' Gadget



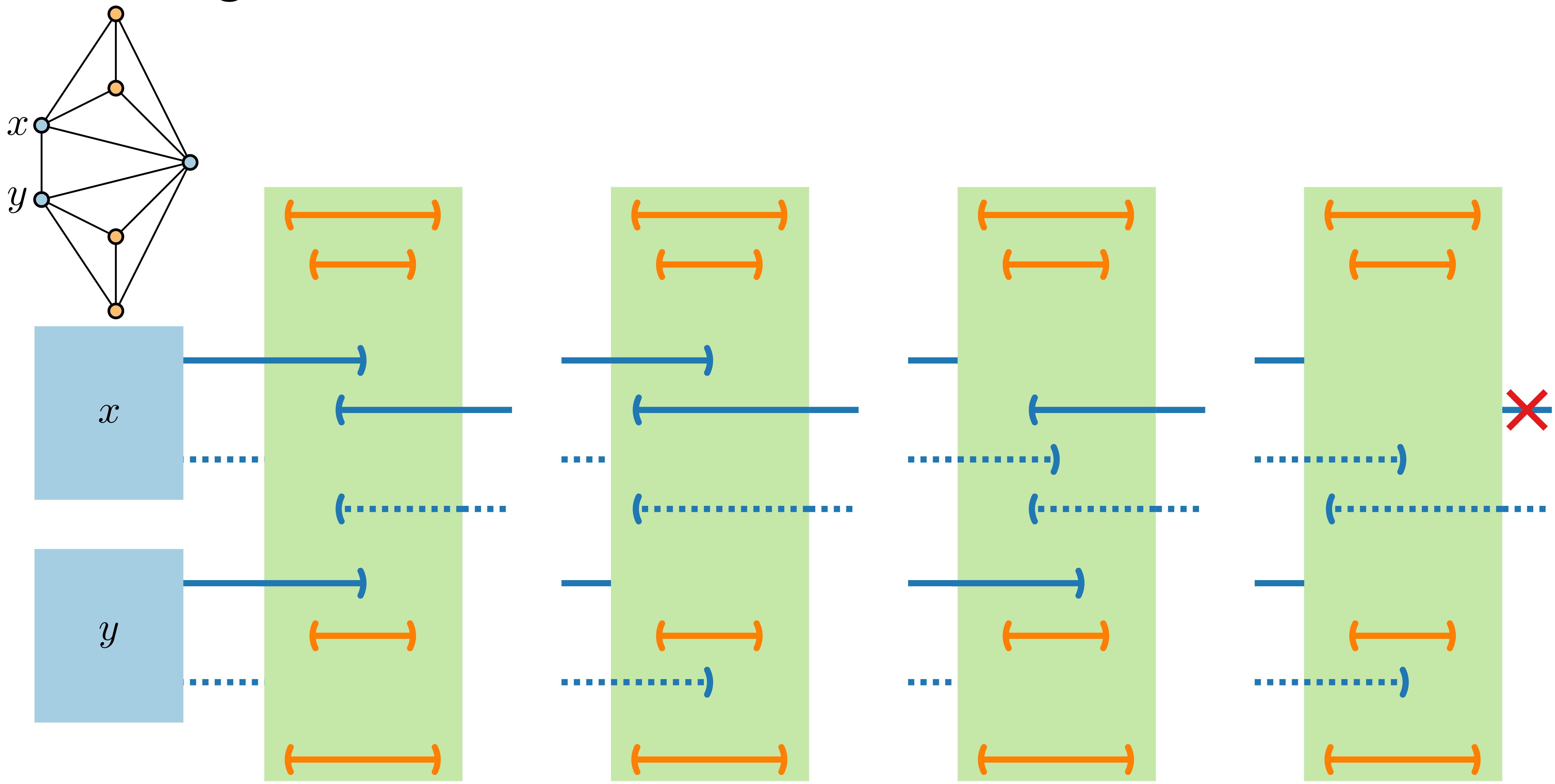
OR' Gadget



OR' Gadget



OR' Gadget



Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.

Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.
- ε-bar visibility representation extension is NP-complete.

Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.
- ε-bar visibility representation extension is NP-complete.
- ε-bar visibility representation extension is NP-complete for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.
- ε-bar visibility representation extension is NP-complete.
- ε-bar visibility representation extension is NP-complete for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

Open Problems:

- Can ~~*rectangular*~~ ε-bar visibility representation extension be solved in polynomial time for st-graphs? For DAGs?

Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.
- ε-bar visibility representation extension is NP-complete.
- ε-bar visibility representation extension is NP-complete for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

Open Problems:

- Can ~~*rectangular*~~ ε-bar visibility representation extension be solved in polynomial time for st-graphs? For DAGs?

Discussion

- *Rectangular* ε-bar visibility representation extension can be solved in $O(n \log^2 n)$ time *for st-graphs*.
- ε-bar visibility representation extension is NP-complete.
- ε-bar visibility representation extension is NP-complete for (series-parallel) st-graphs when restricted to the *integer grid* (or if any fixed $\varepsilon > 0$ is specified).

Open Problems:

- Can ~~*rectangular*~~ ε-bar visibility representation extension be solved in polynomial time for st-graphs? For DAGs?
- Can *strong* bar visibility recognition / representation extension be solved in polynomial time for st-graphs?

Literature

Main source:

- [Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta '18]
  The Partial Visibility Representation Extension Problem

Referenced papers:

- [Tamassia, Tollis '86] Algorithms for visibility representations of planar graphs
- [Wismath '85] Characterizing bar line-of-sight graphs
- [Chaplick, Dorbec, Kratochvíl, Montassier, Stacho '14]
  Contact representations of planar graphs: Extending a partial representation is hard
- [Andreae '92] Some results on visibility graphs
- [Garg, Tamassia '01]
  On the Computational Complexity of Upward and Rectilinear Planarity Testing
- [Gutwenger, Mutzel '01] A Linear Time Implementation of SPQR-Trees
- [de Berg, Khosravi '10] Optimal Binary Space Partitions in the Plane