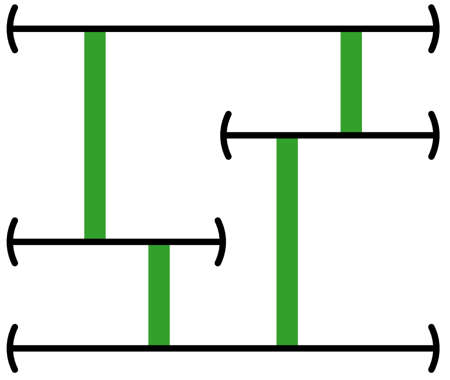
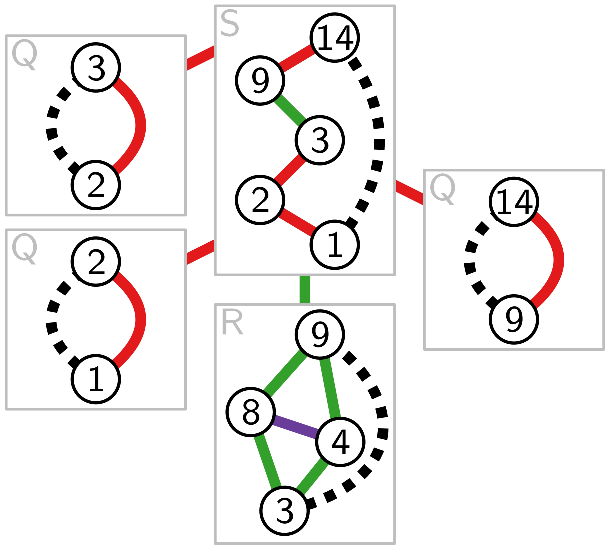
Visualization of graphs

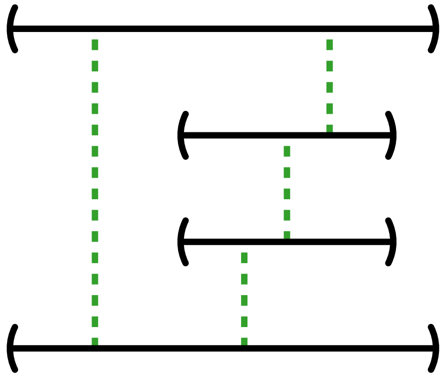
Partial visibility representation extension

With SPQR-trees

Jonathan Klawitter · Summer semester 2020





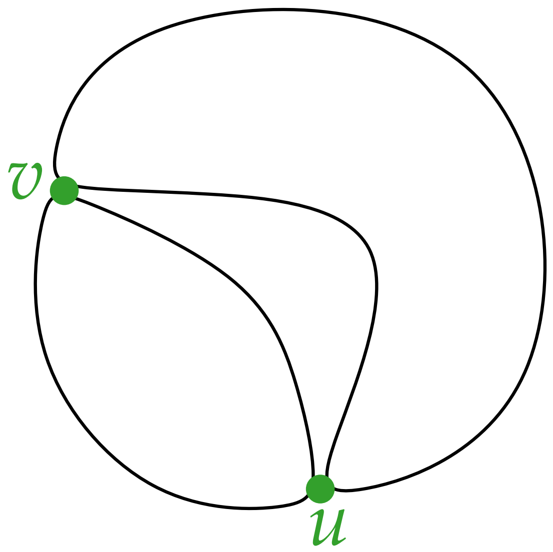


SPQR-tree

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

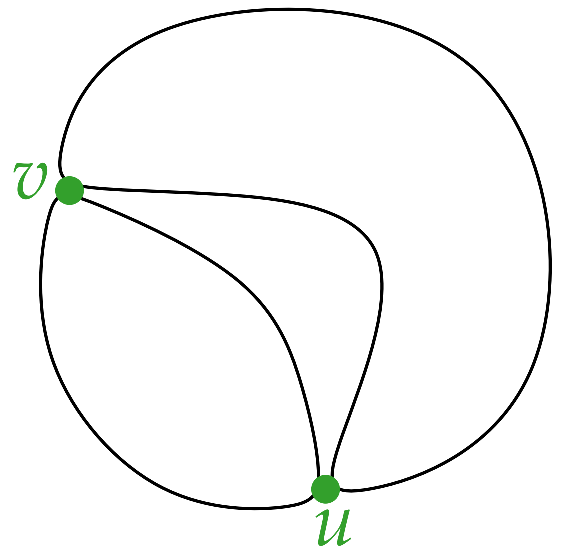
SPQR-tree

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



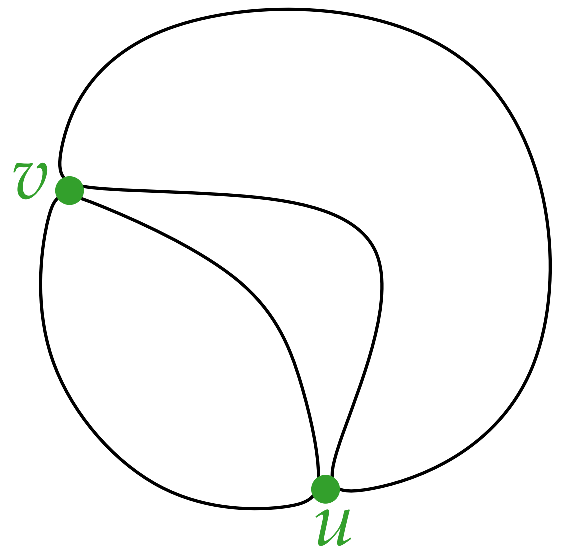
SPQR-tree

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



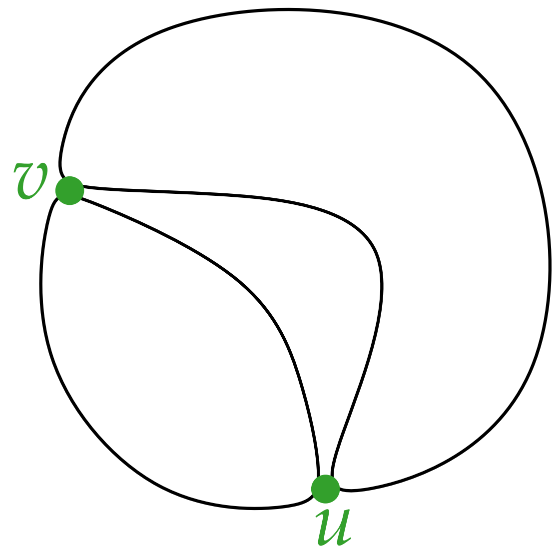
SPQR-tree

- 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.



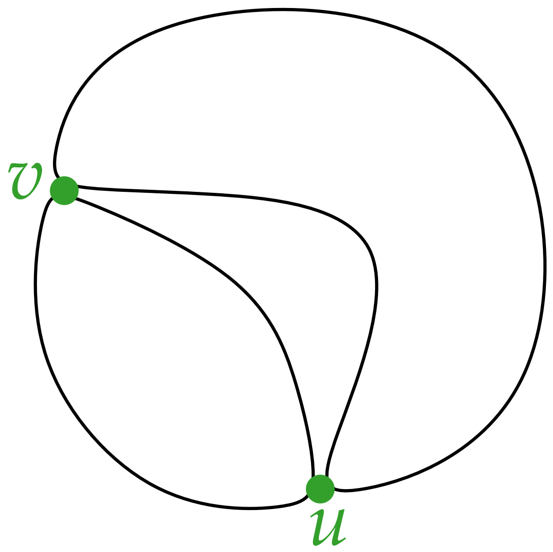
SPQR-tree

- 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.

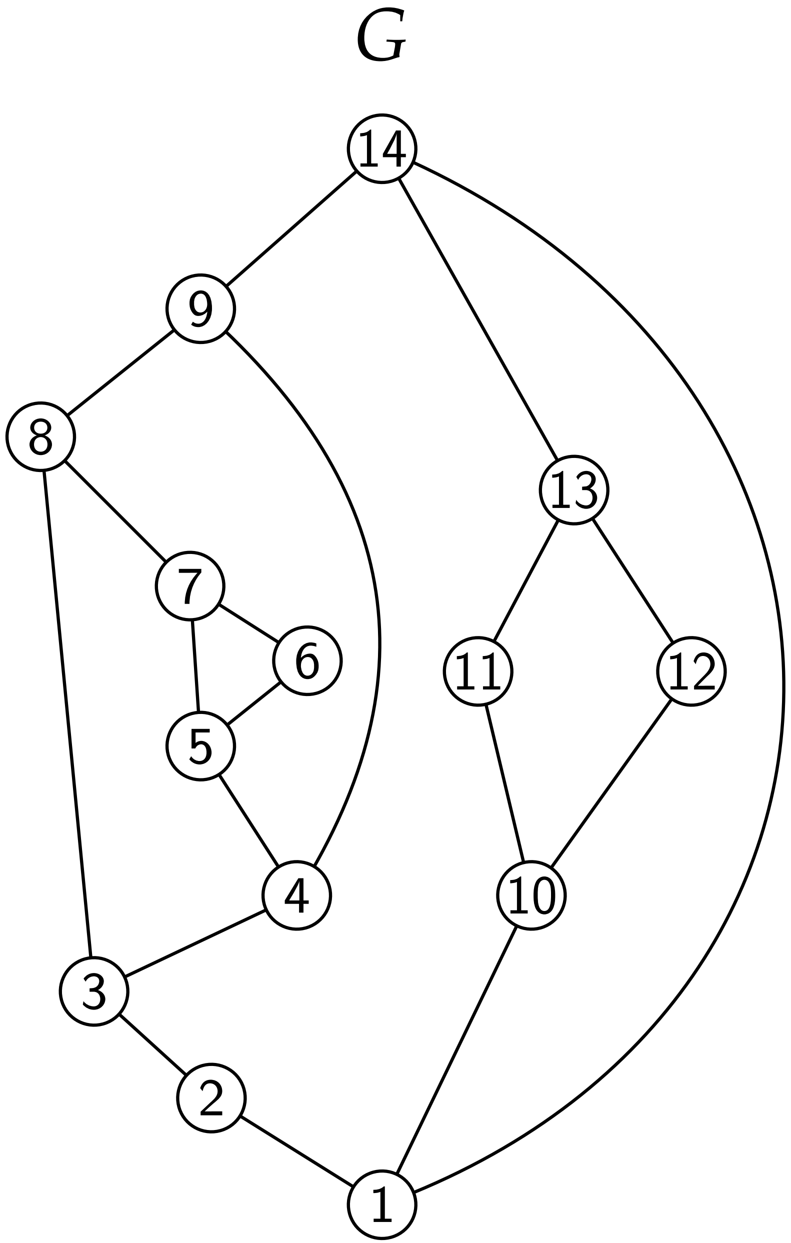


SPQR-tree

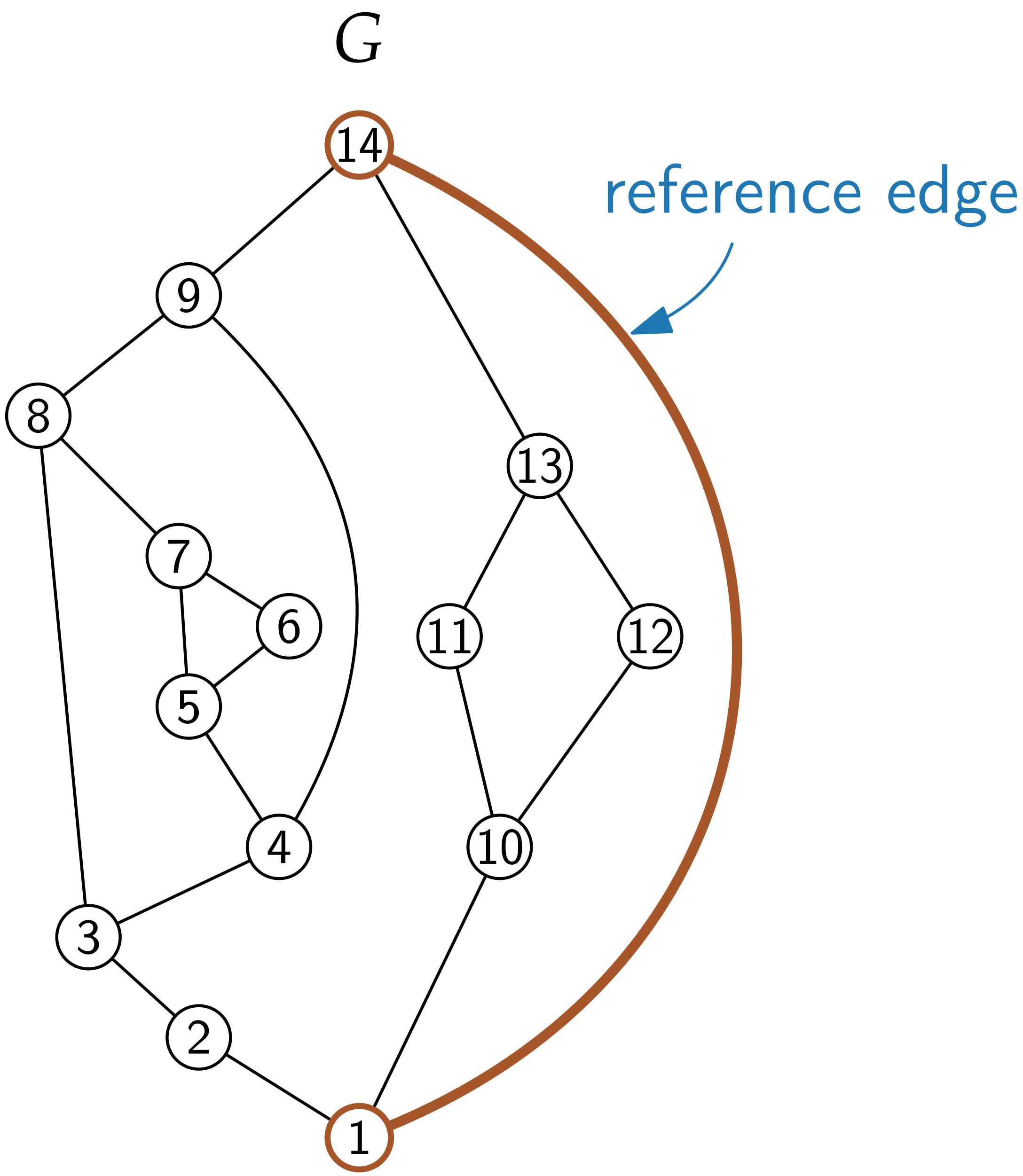
- An **SPQR-tree** T is a decomposition of a planar graph G by **separation pairs**.
- The nodes of T are of four types:
  - **S** nodes represent a series composition
  - **P** nodes represent a parallel composition
  - **Q** nodes represent a single edge
  - **R** nodes represent 3-connected (*rigid*) 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 $\mathcal{O}(n)$ time. [Gutwenger, Mutzel '01]

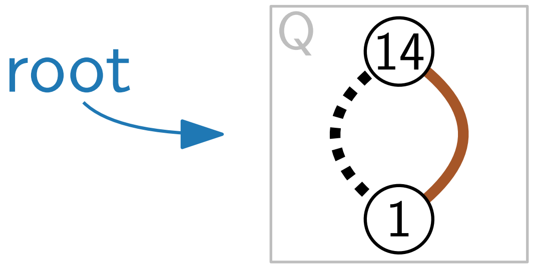


SPQR-tree example

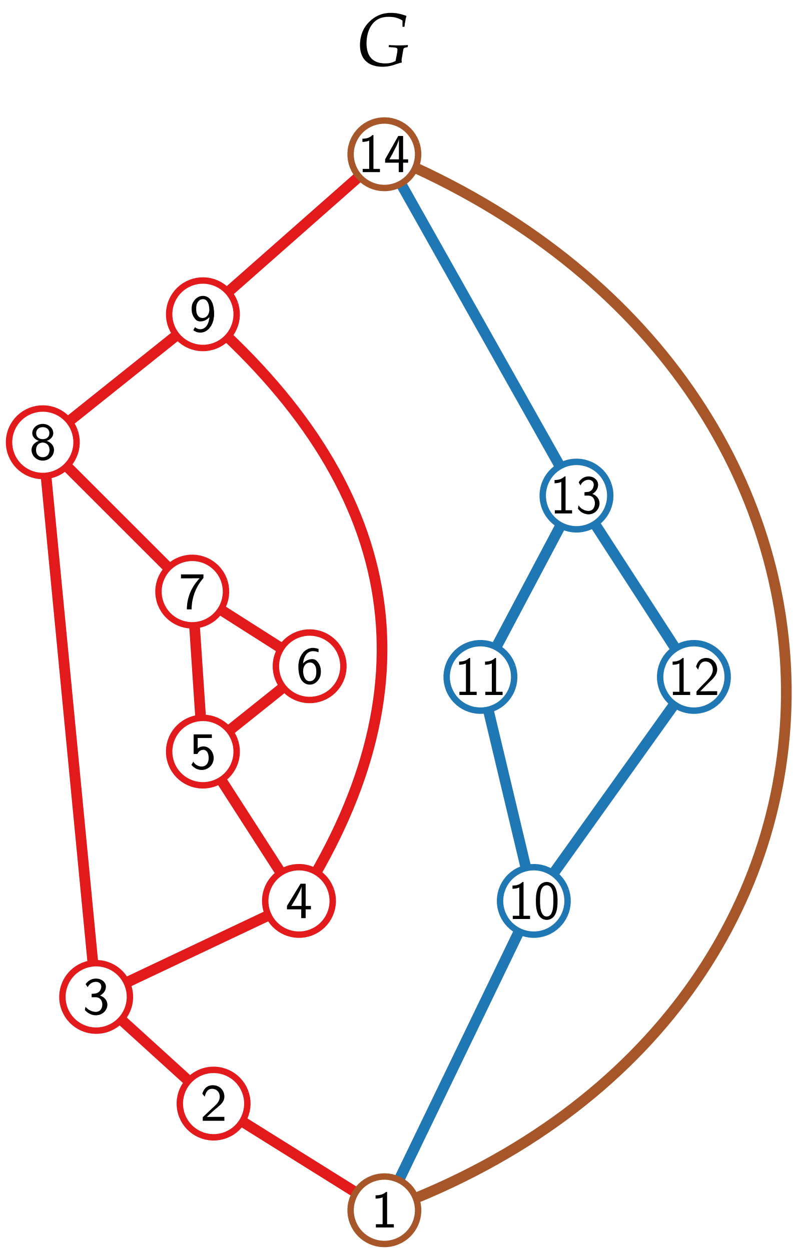


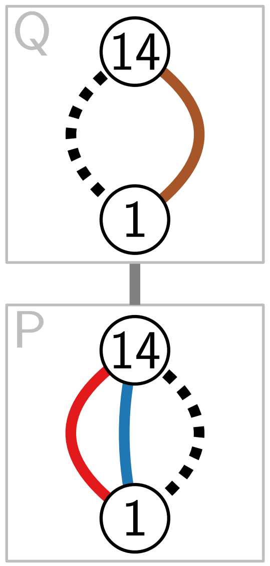
SPQR-tree example



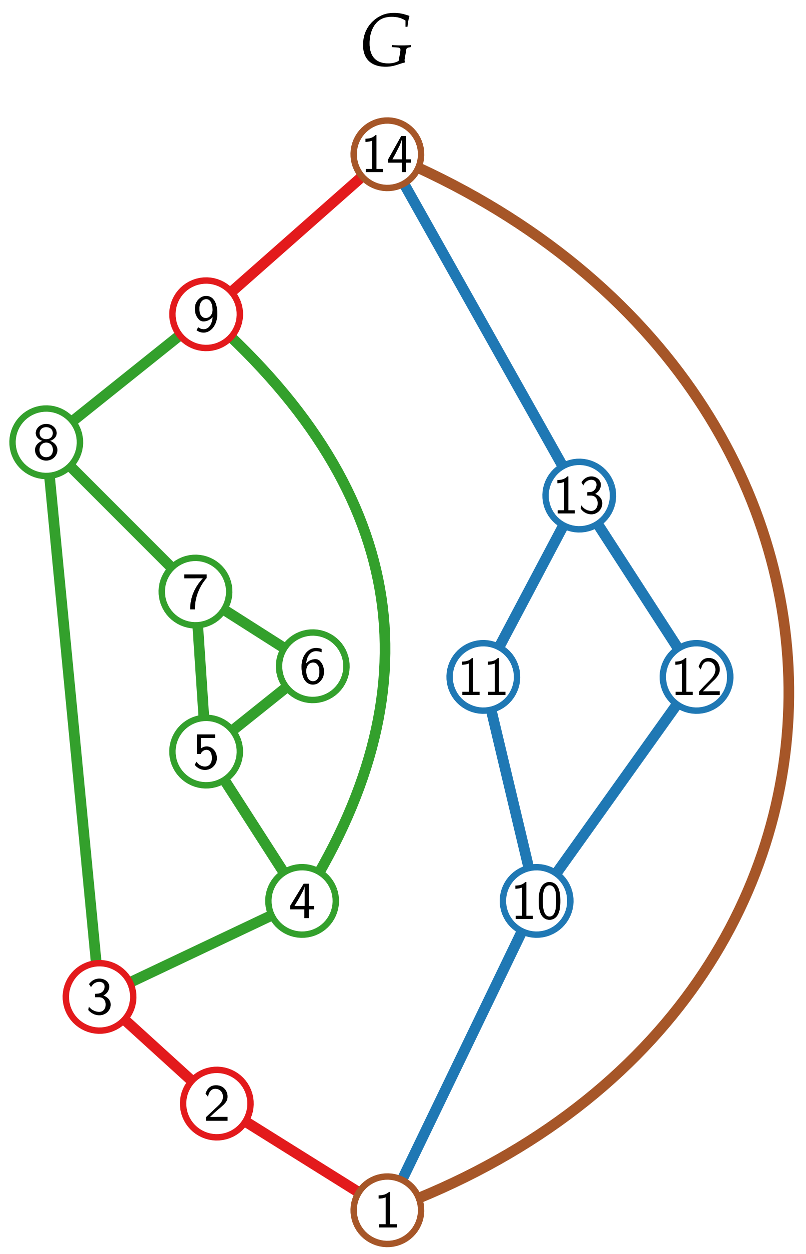


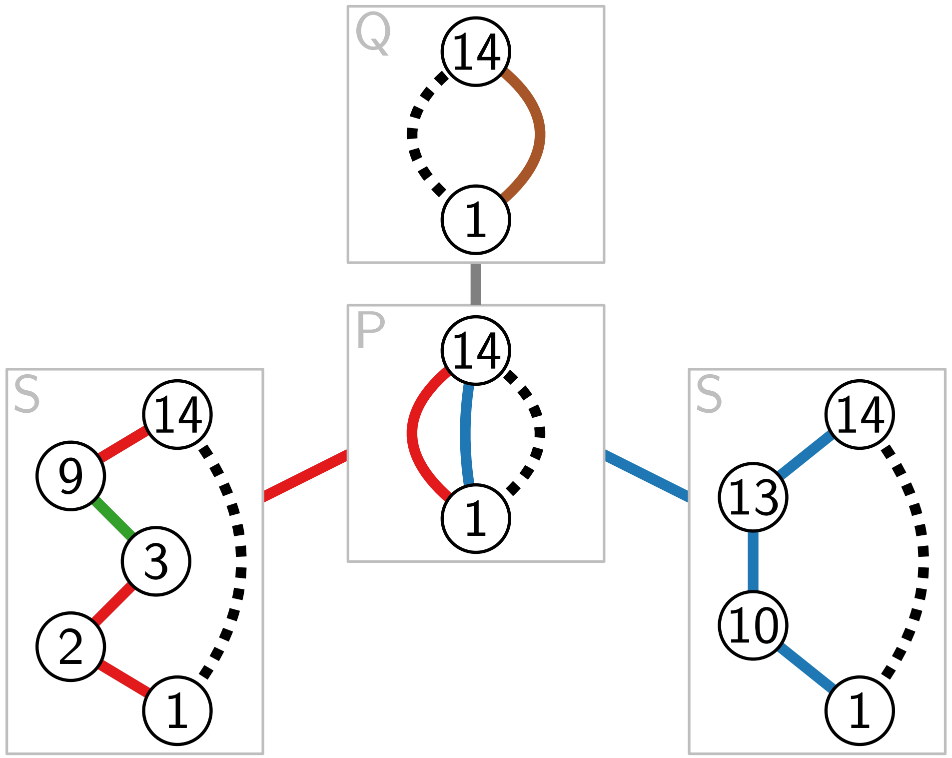
SPQR-tree example



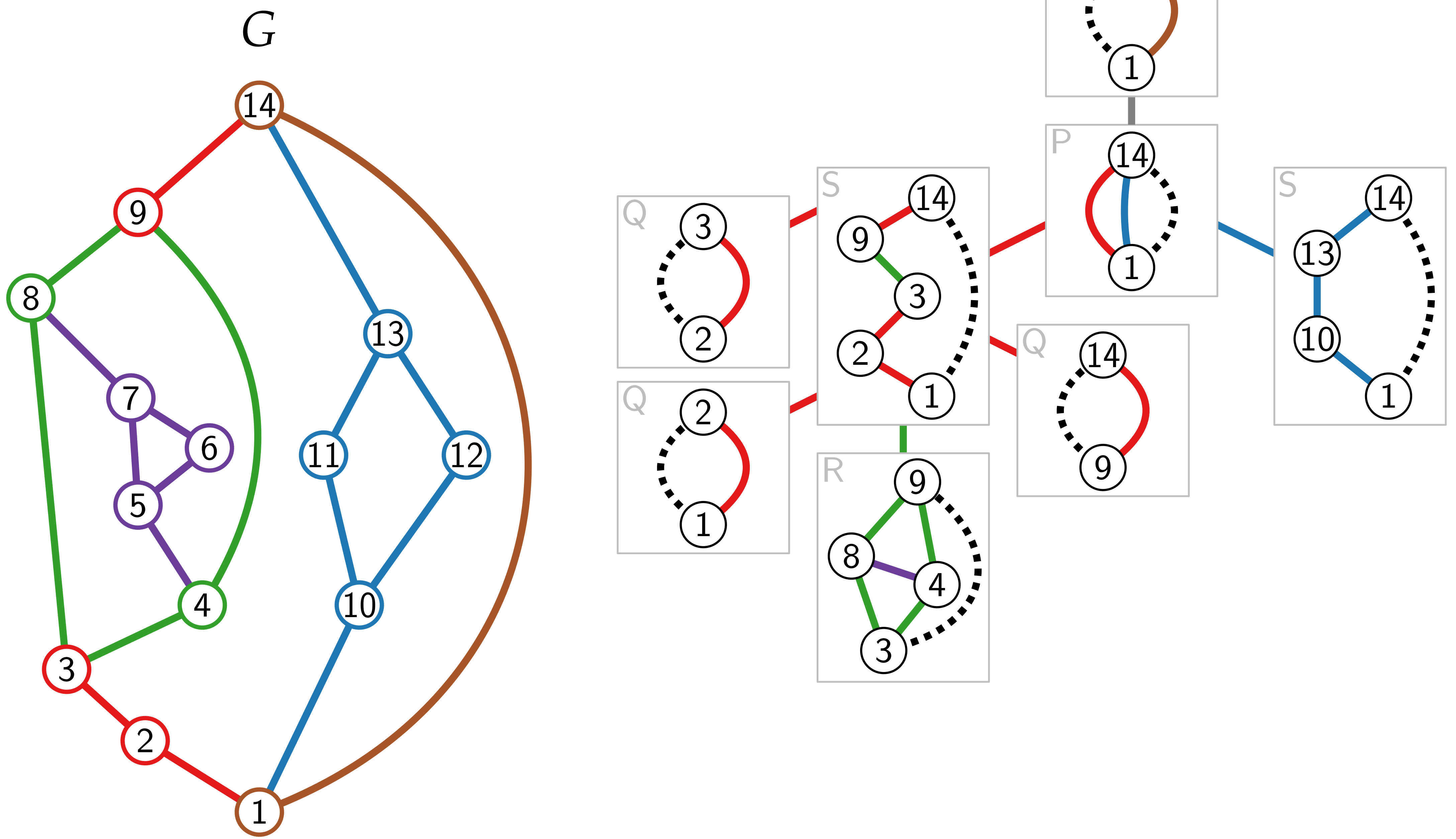


SPQR-tree example

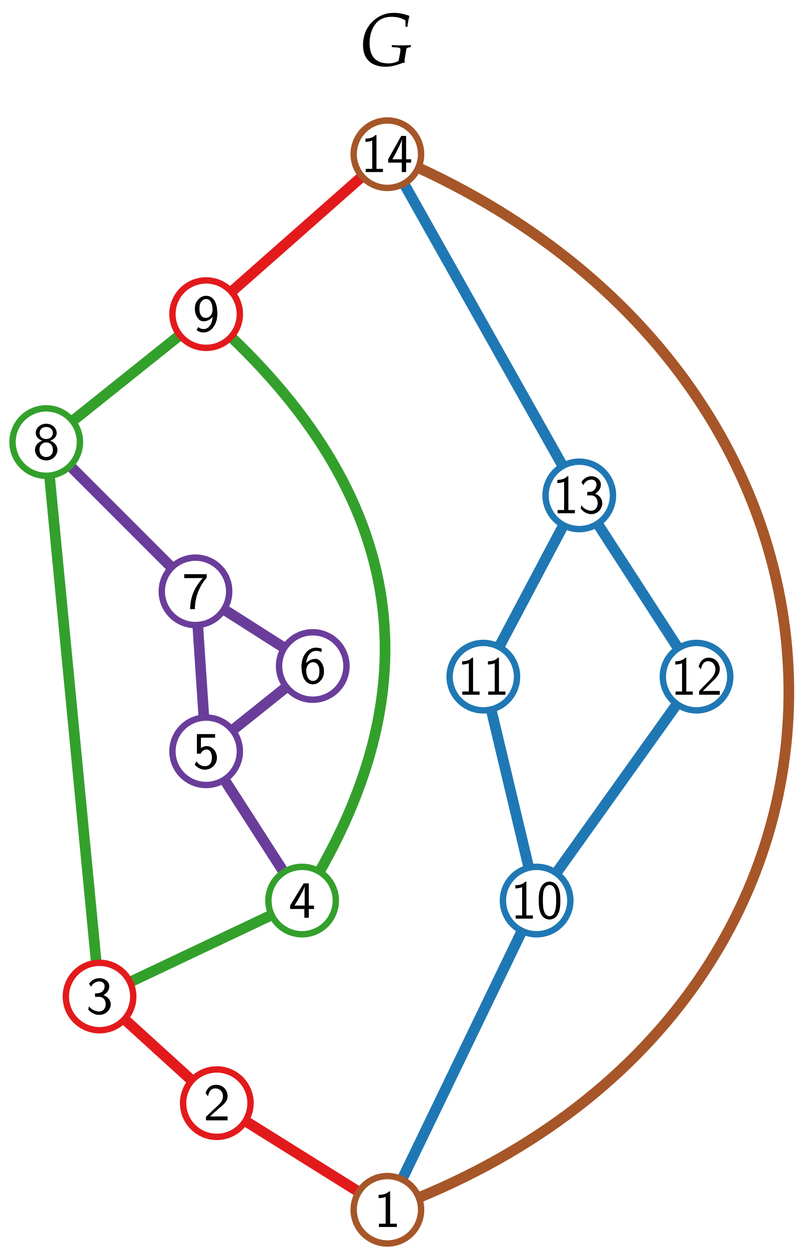


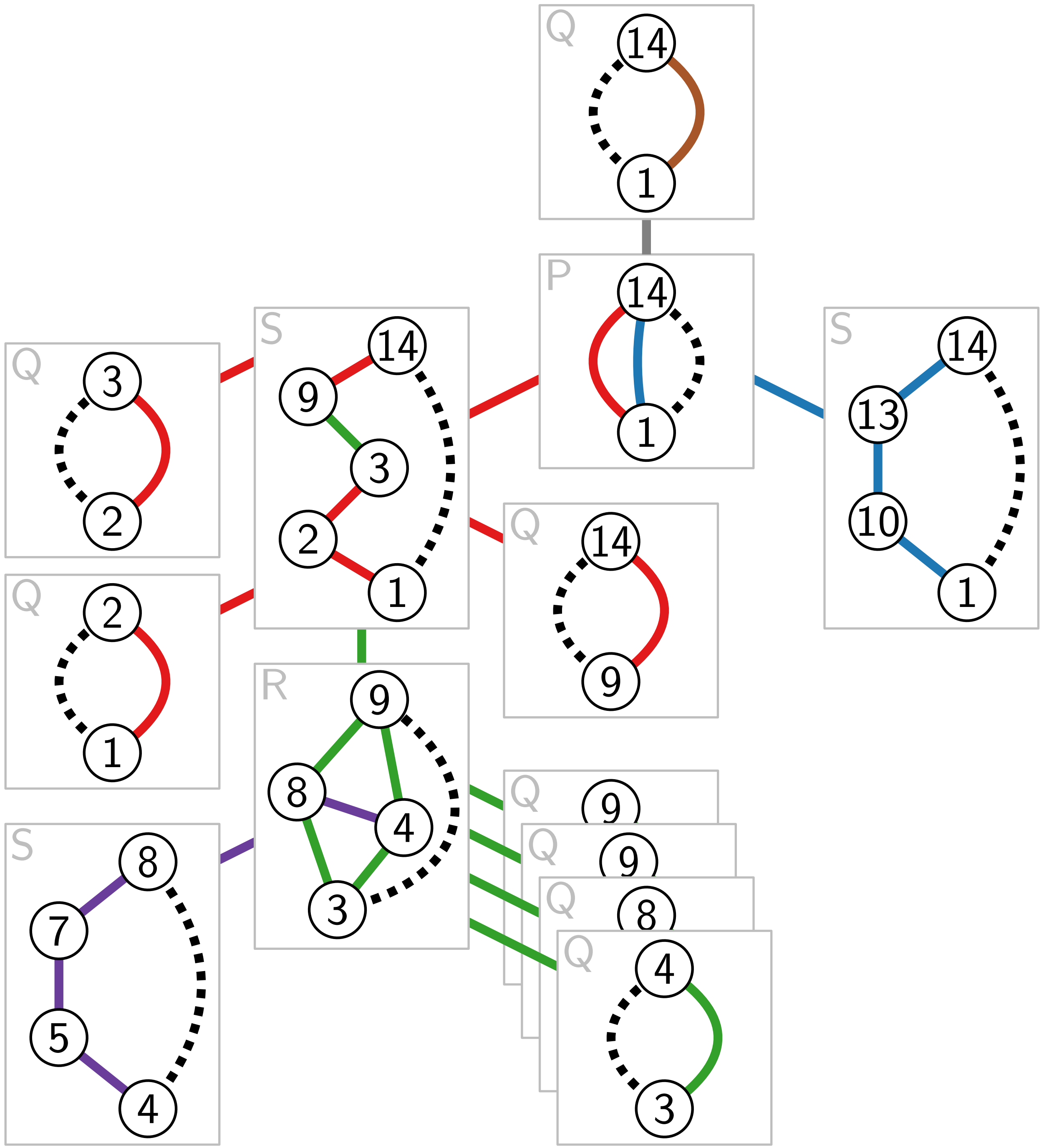


SPQR-tree example

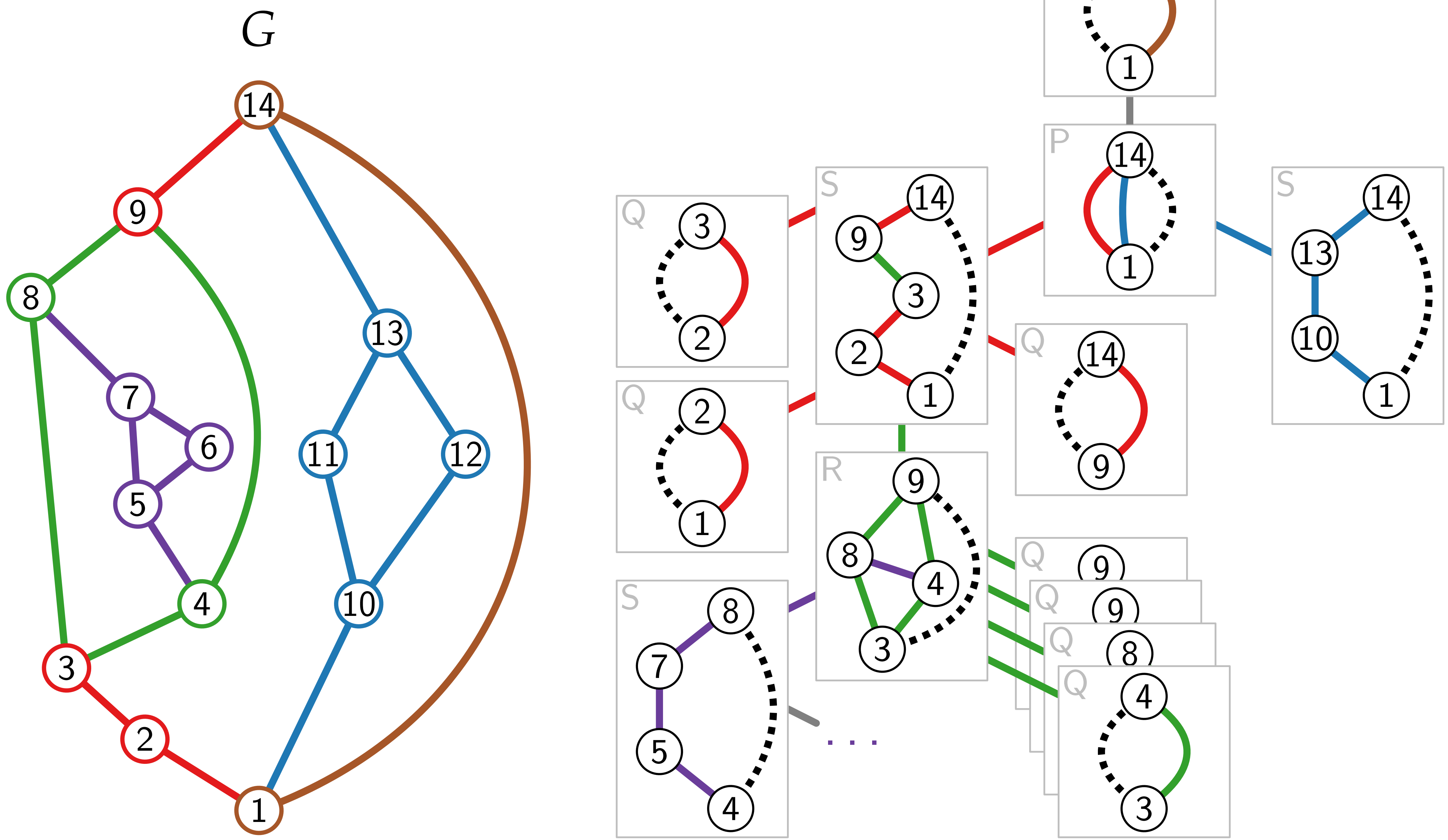


SPQR-tree example

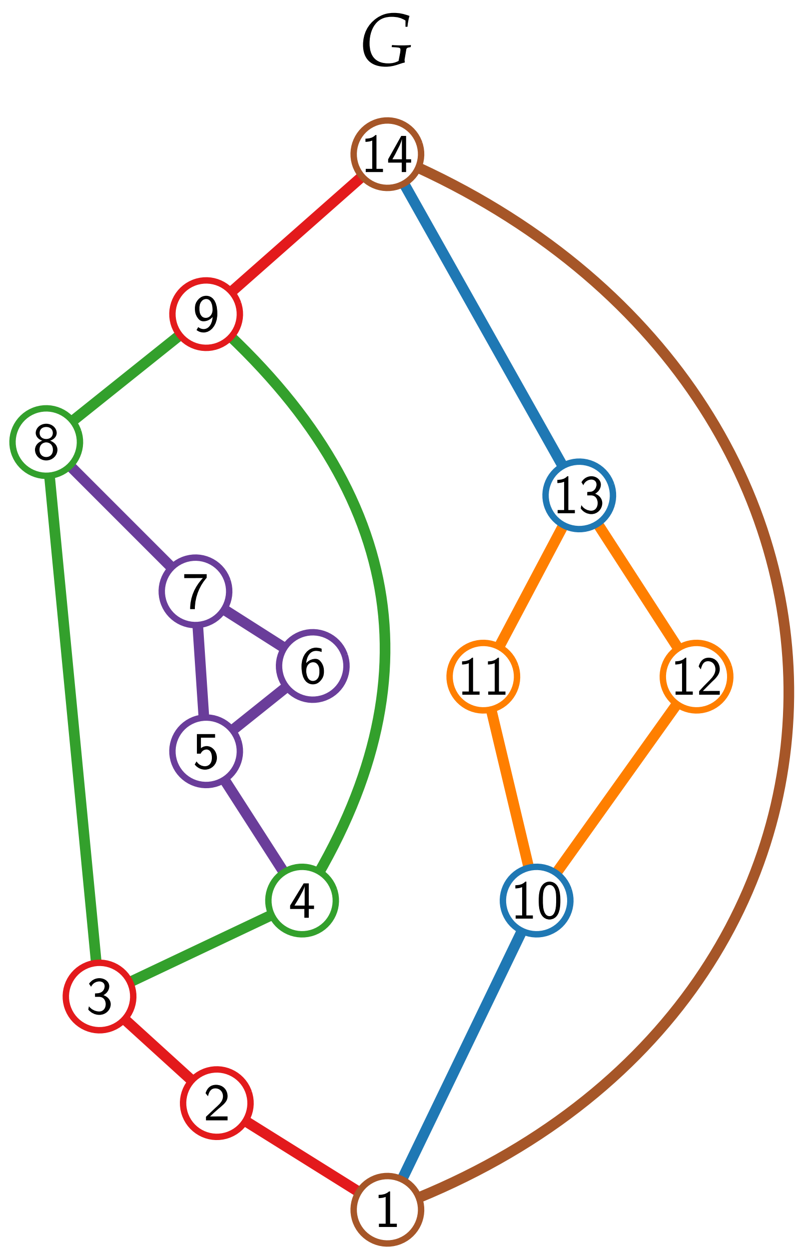


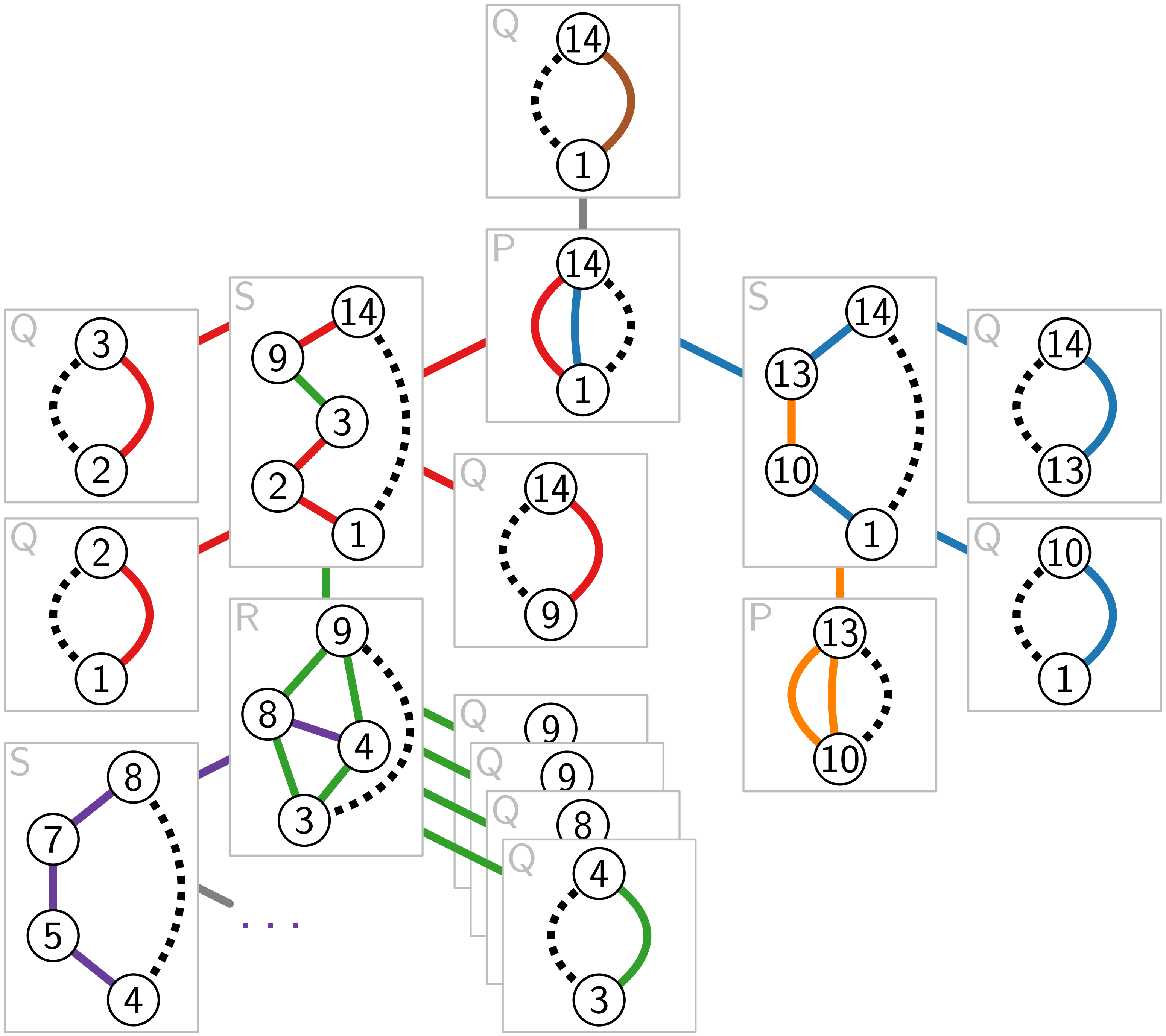


SPQR-tree example

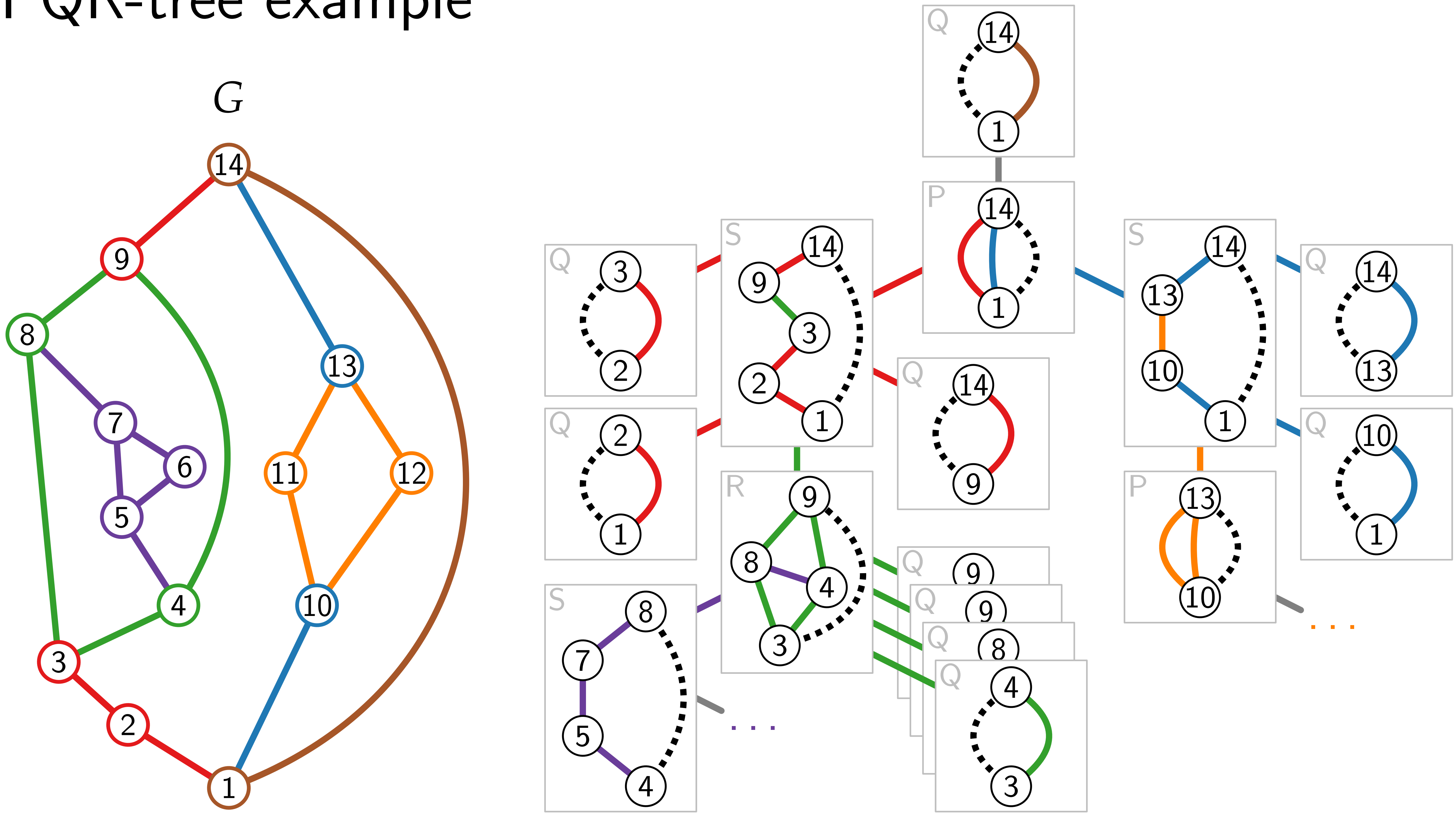


SPQR-tree example



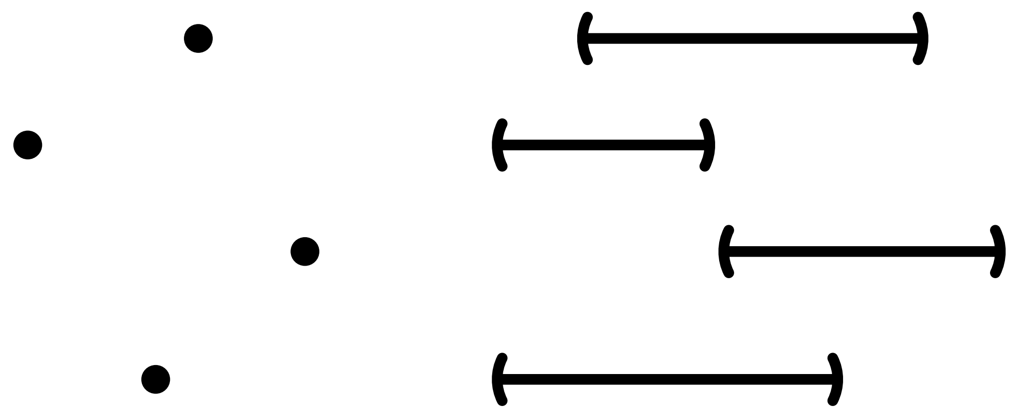


SPQR-tree example



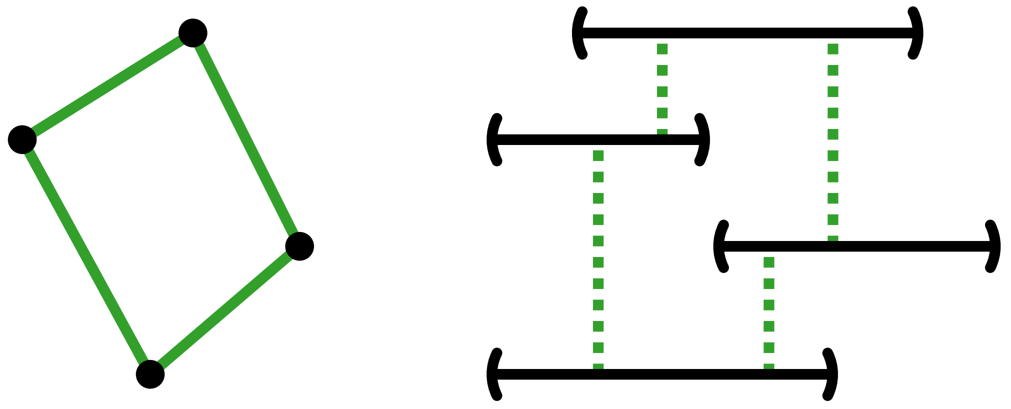
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**



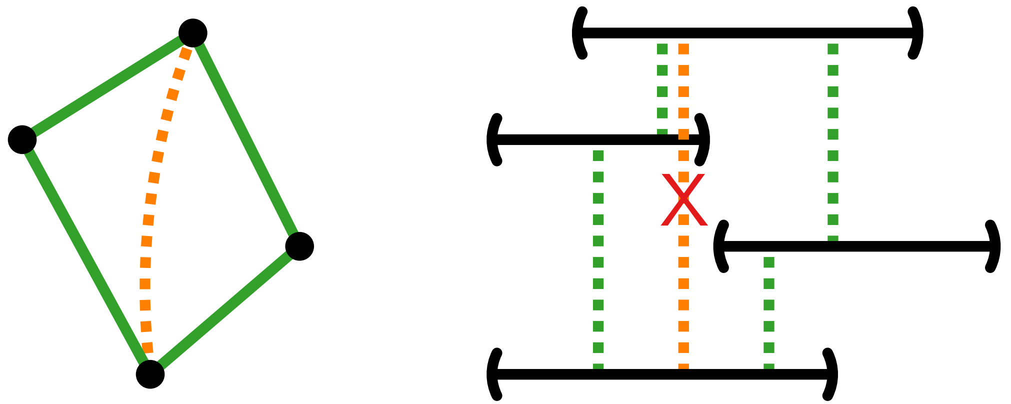
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines



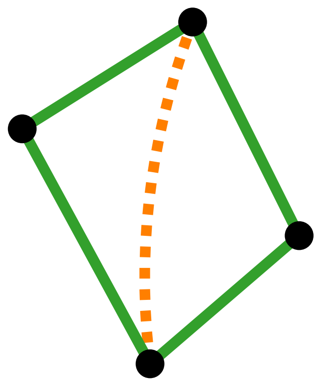
Bar visibility representation

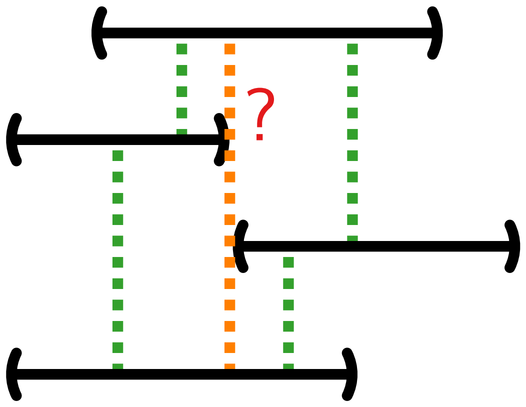
- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines



Bar visibility representation

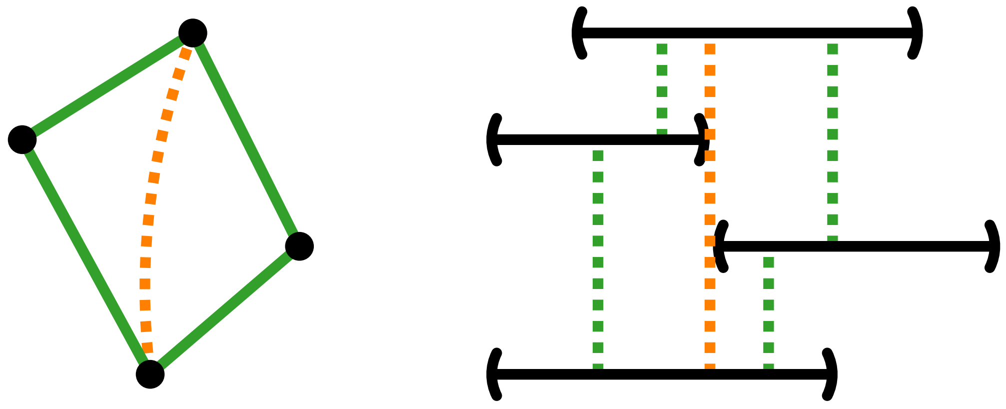
- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?





Bar visibility representation

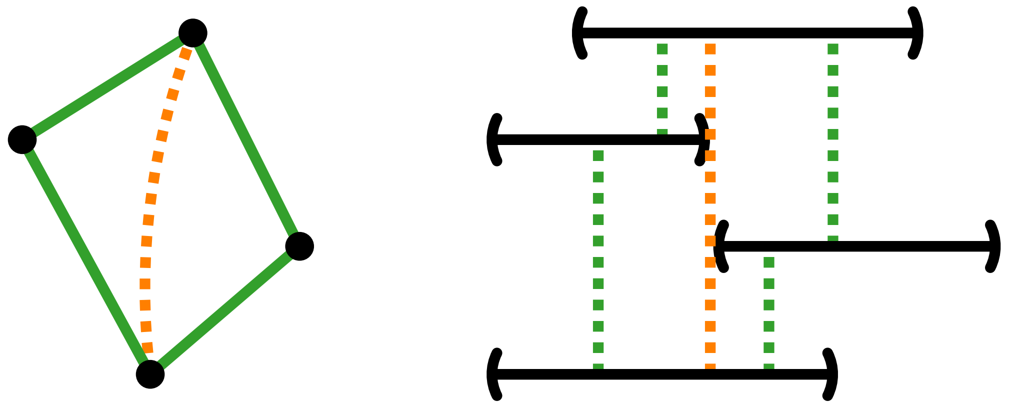
- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?



Models.

Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?

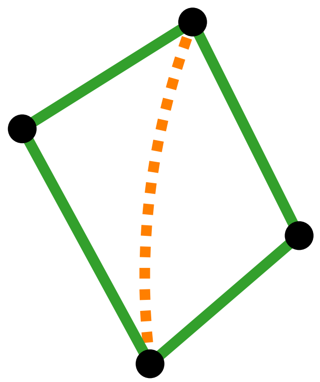


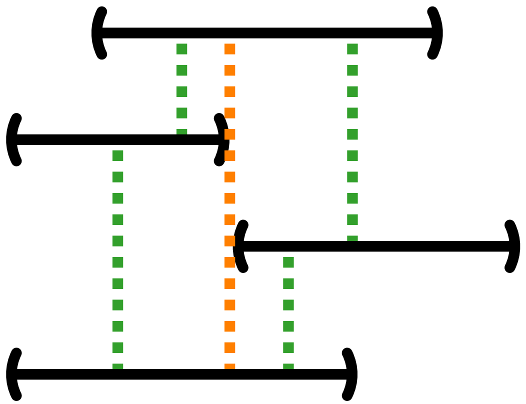
Models.

- **Strong:** Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical sightlines

Bar visibility representation

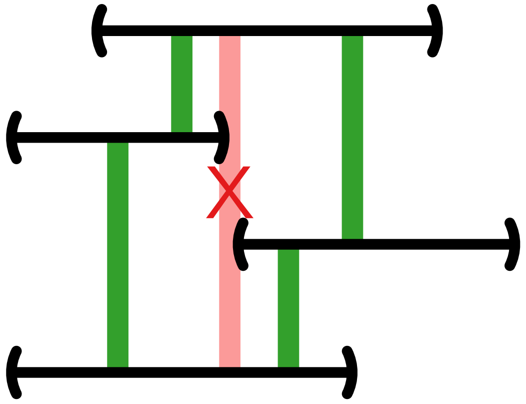
- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?





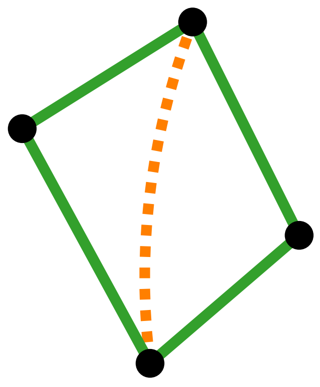
Models.

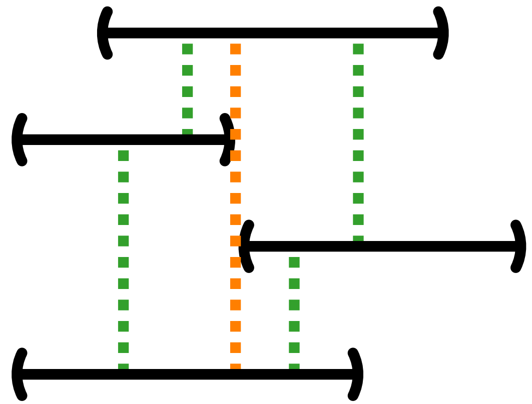
- **Strong:** Edge $uv \Leftrightarrow$ unobstructed **0-width** vertical sightlines
- ε**:** Edge $uv \Leftrightarrow \epsilon$ wide vertical sightlines for $\varepsilon > 0$



Bar visibility representation

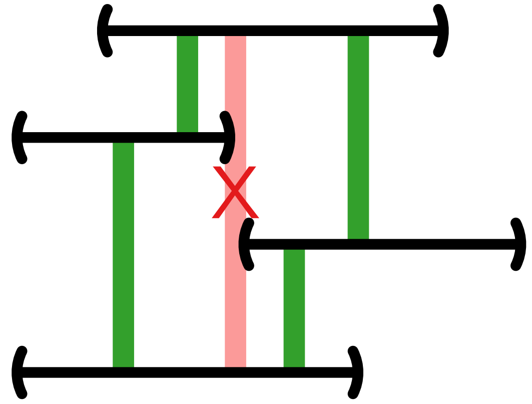
- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?



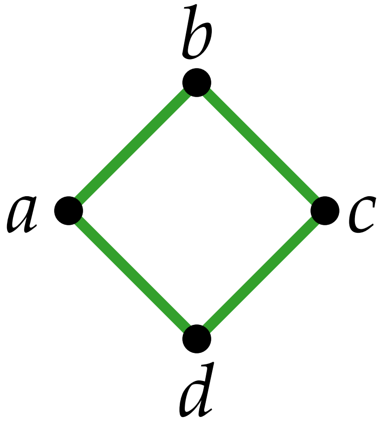


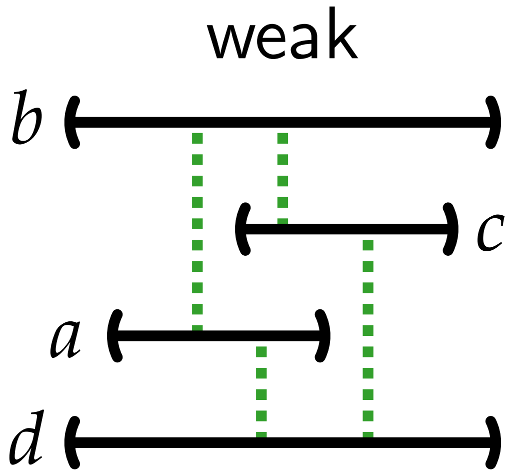
Models.

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

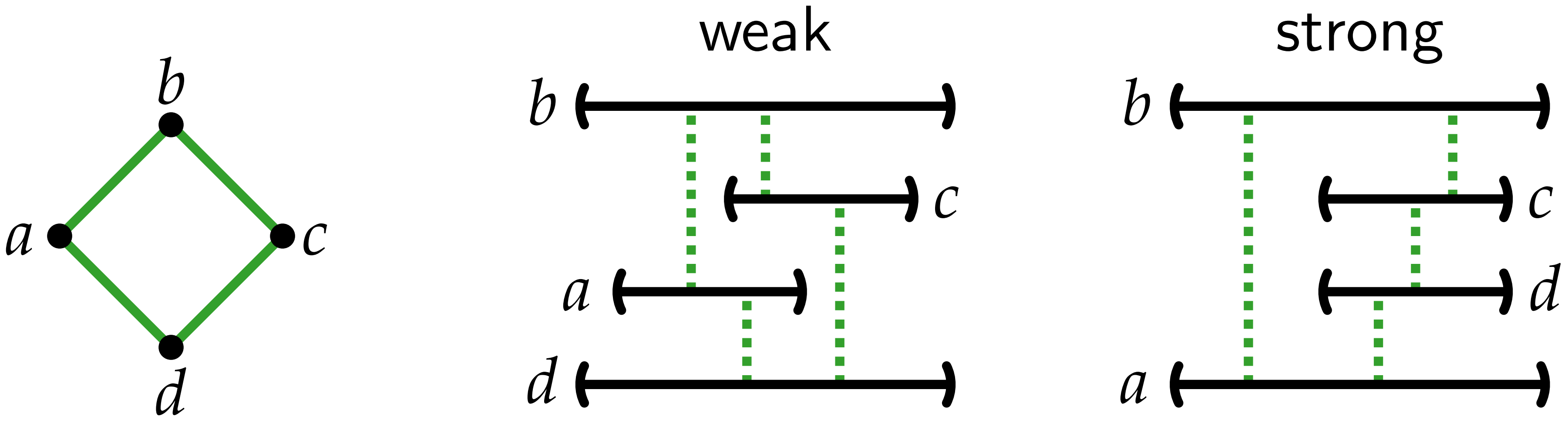


Problems

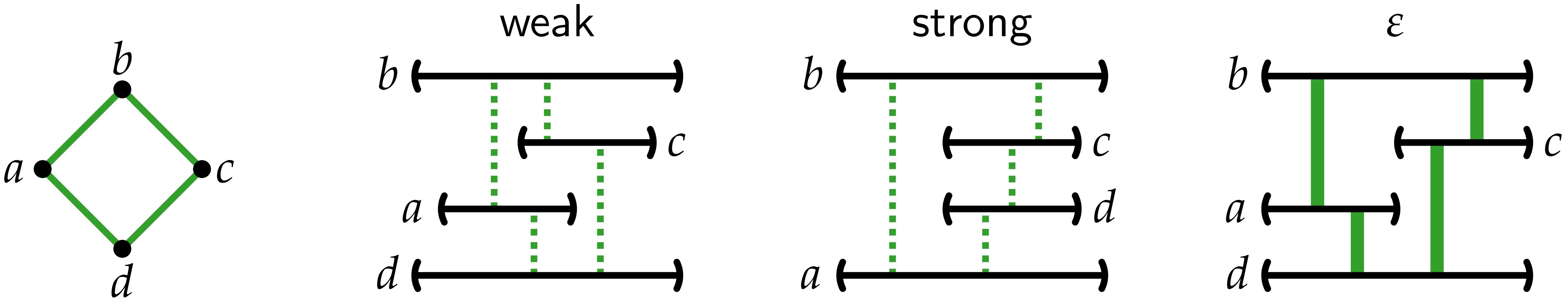




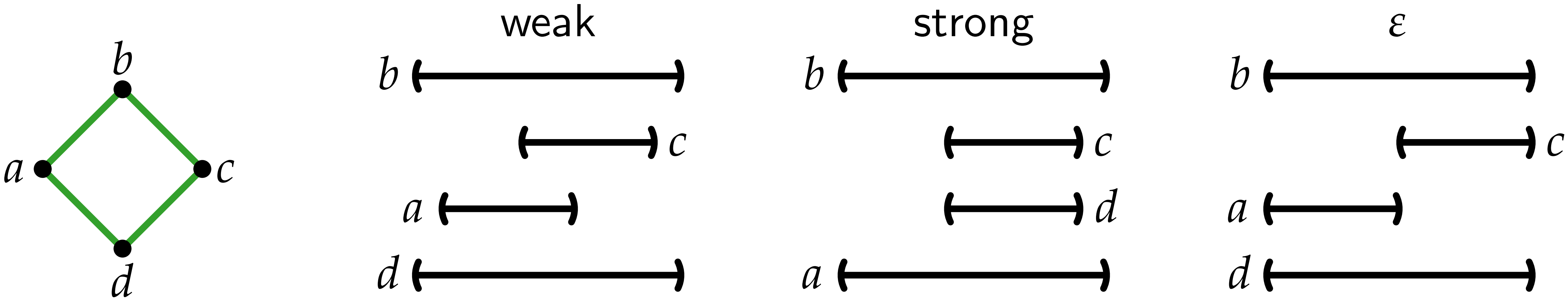
Problems



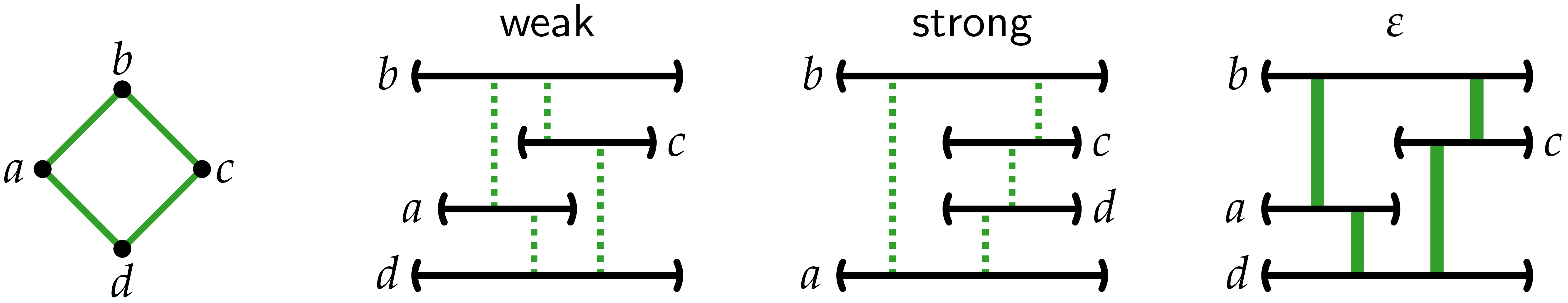
Problems



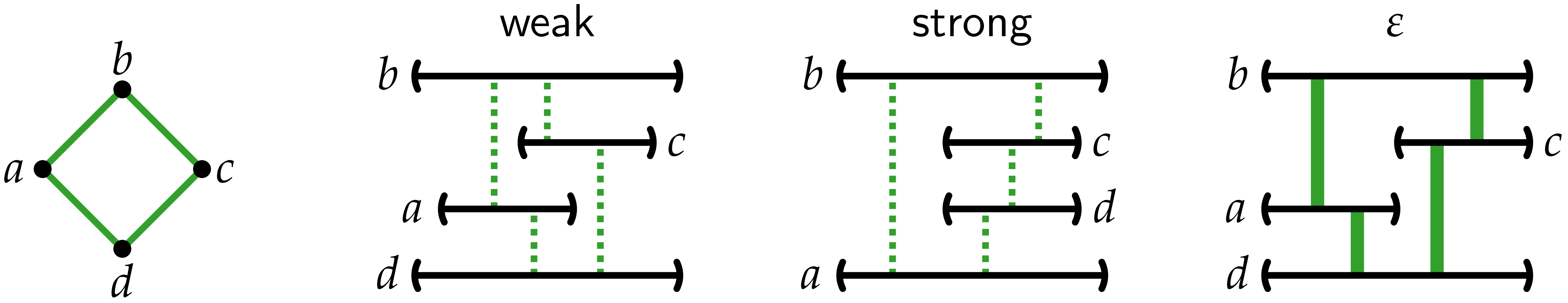
Problems



Problems



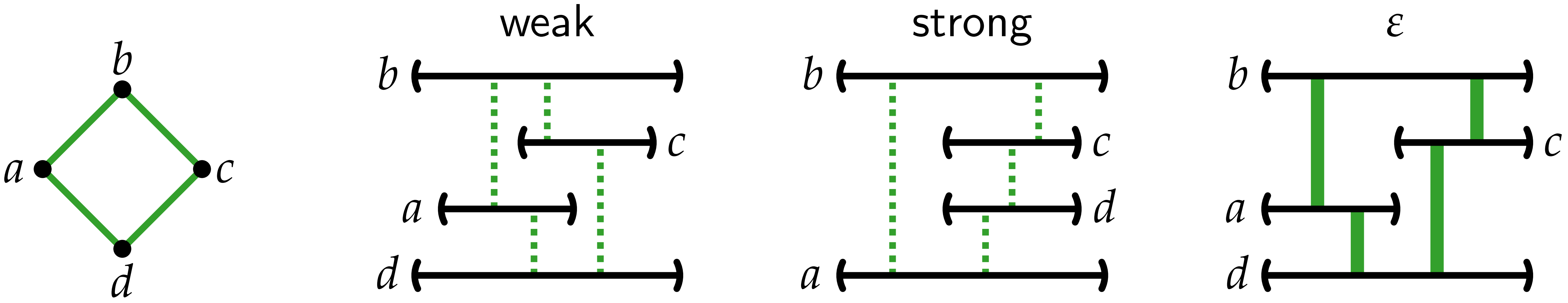
Problems



Recognition problem.

Given a graph G, **decide** if there exists a weak/strong/ε bar visibility representation ψ of G.

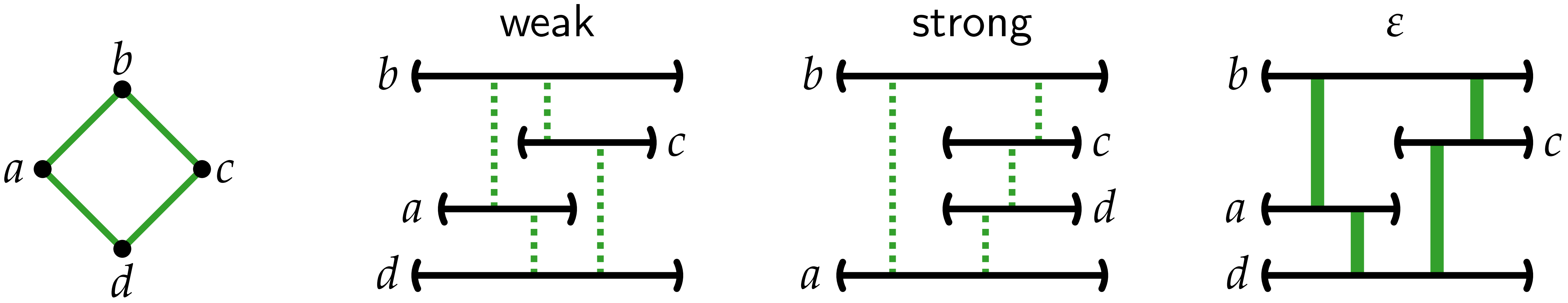
Problems



Recognition problem.
Given a graph G, **decide** if 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 when one exists.

Problems

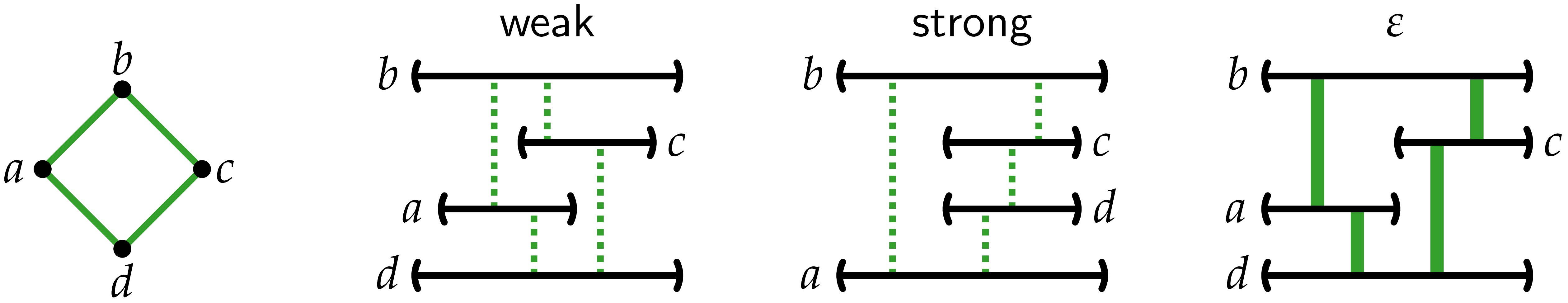


Recognition problem.
Given a graph G, **decide** if there exists a weak/strong/ε bar visibility representation ψ of G.

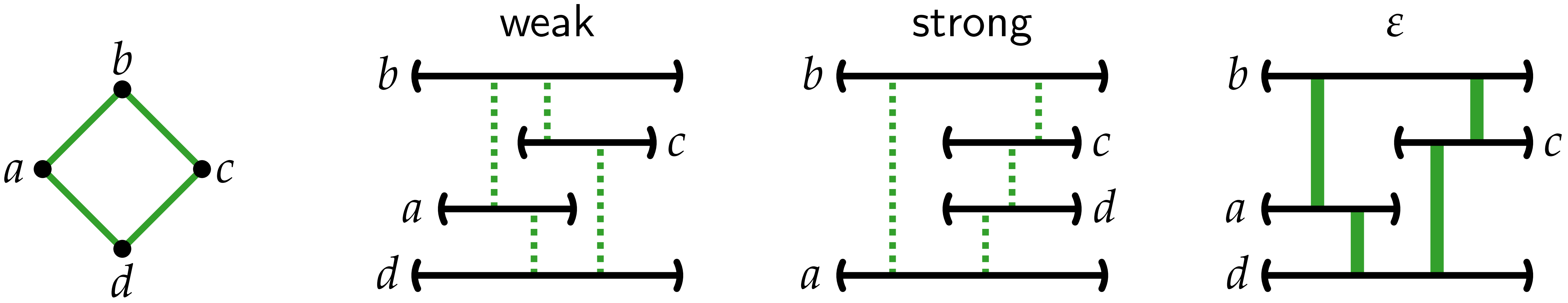
Construction problem.
Given a graph G, **construct** a weak/strong/ε bar visibility representation ψ of G when one exists.

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

Background



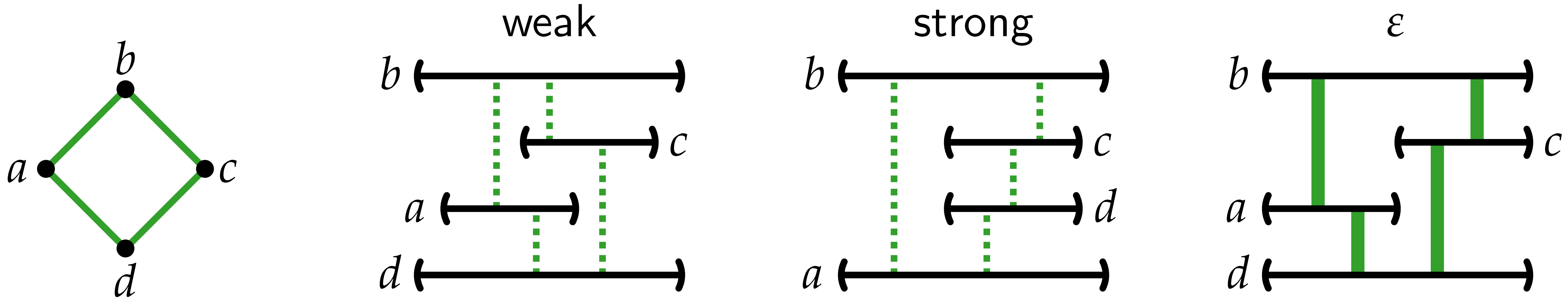
Background



Weak Bar Visibility.

- All planar graphs. [Tamassia & Tollis 1986; Wismath 1985]
- Linear time recognition and construction [T&T '86]
- Representation Extension is NP-complete [Chaplick et al. '14]

Background



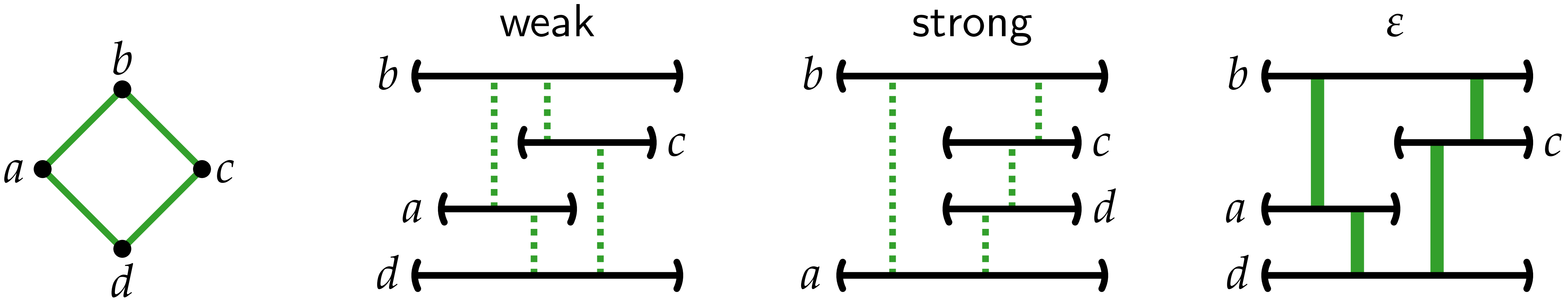
Weak Bar Visibility.

- All planar graphs. [Tamassia & Tollis 1986; Wismath 1985]
- 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]

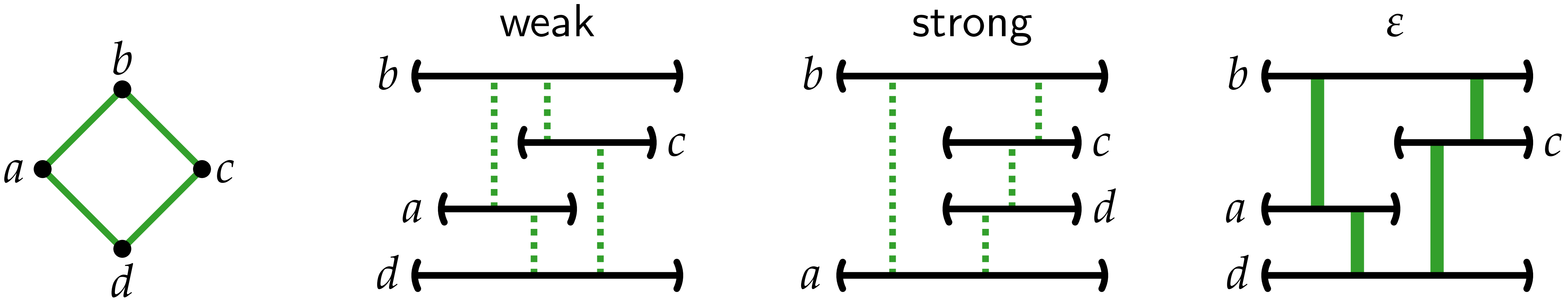
Background



ε-Bar Visibility.

- Planar graphs that can be embedded with all **cut vertices** on the outerface. [T&T 1986, Wismath '85]
- Linear time recognition and construction [T&T '86]
- What about Representation Extension?

Background



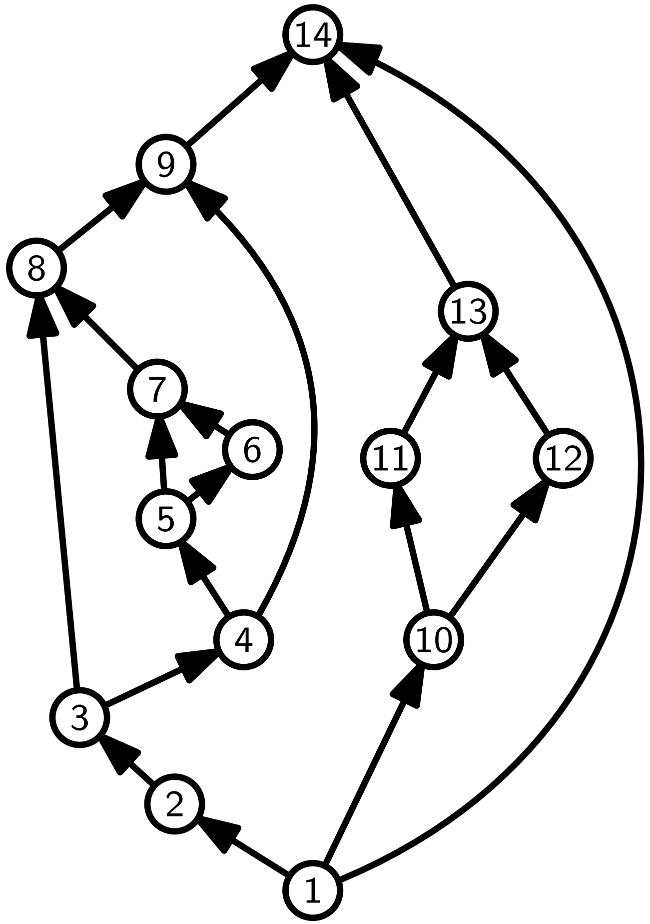
ε-Bar Visibility.

- Planar graphs that can be embedded with all **cut vertices** on the outerface. [T&T 1986, Wismath '85]
- Linear time recognition and construction [T&T '86]
- What about Representation Extension?

Let's see!

ε-bar visiblity and st-graphs

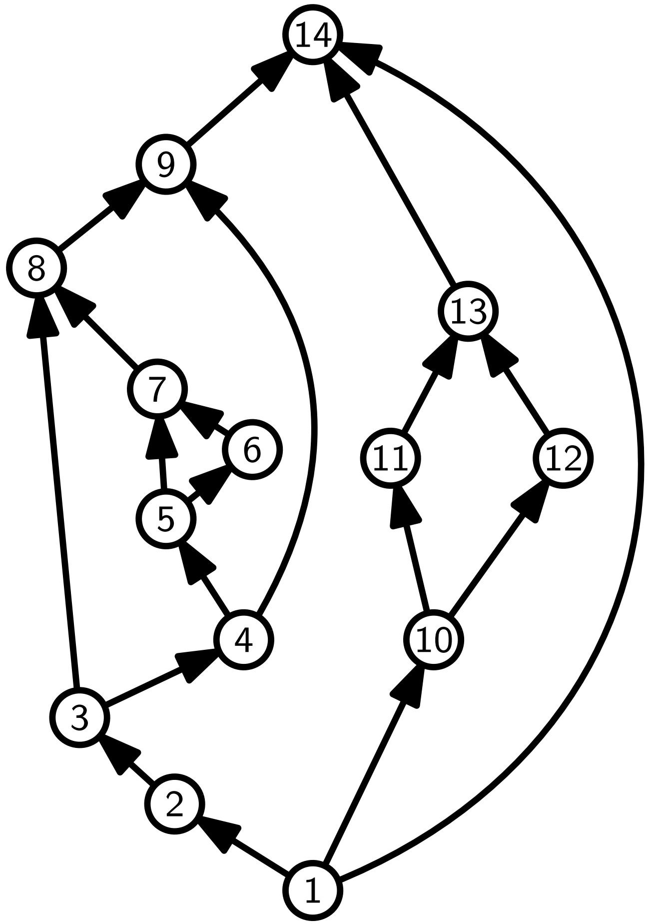
Recall that an **st-graph** is a planar digraph G with exactly one soure s and one sink t where s and t occur on the outer face of an embedding of G.



ε-bar visiblity and st-graphs

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

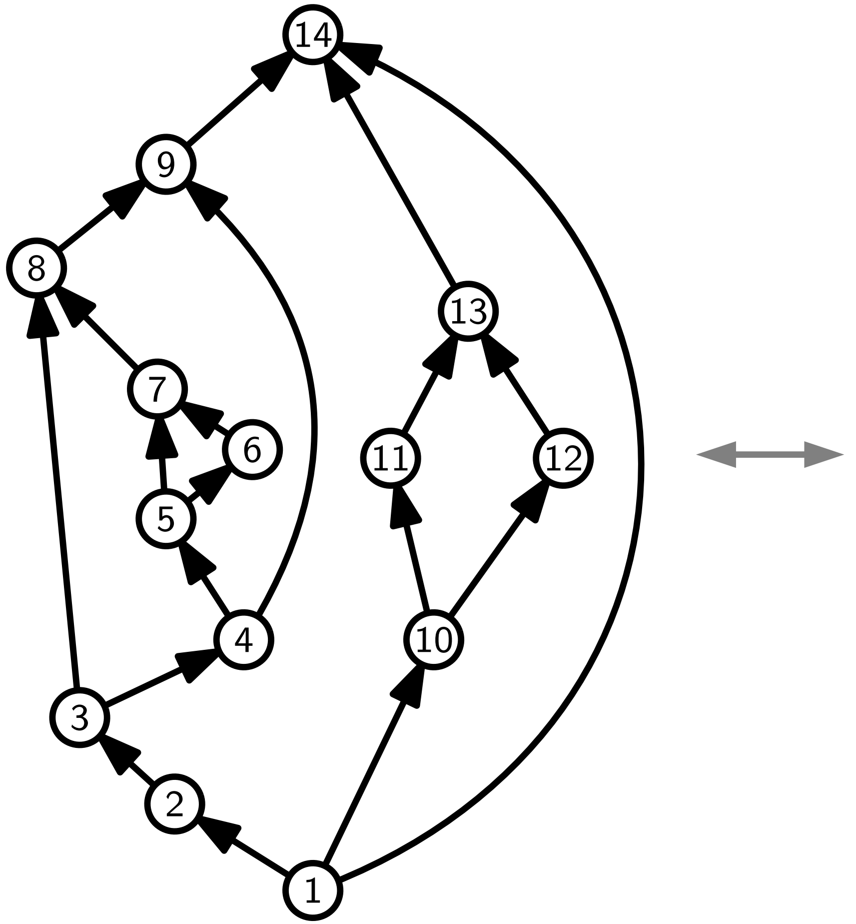
Observation.
st-orientations correspond to ε-bar visibility representations.

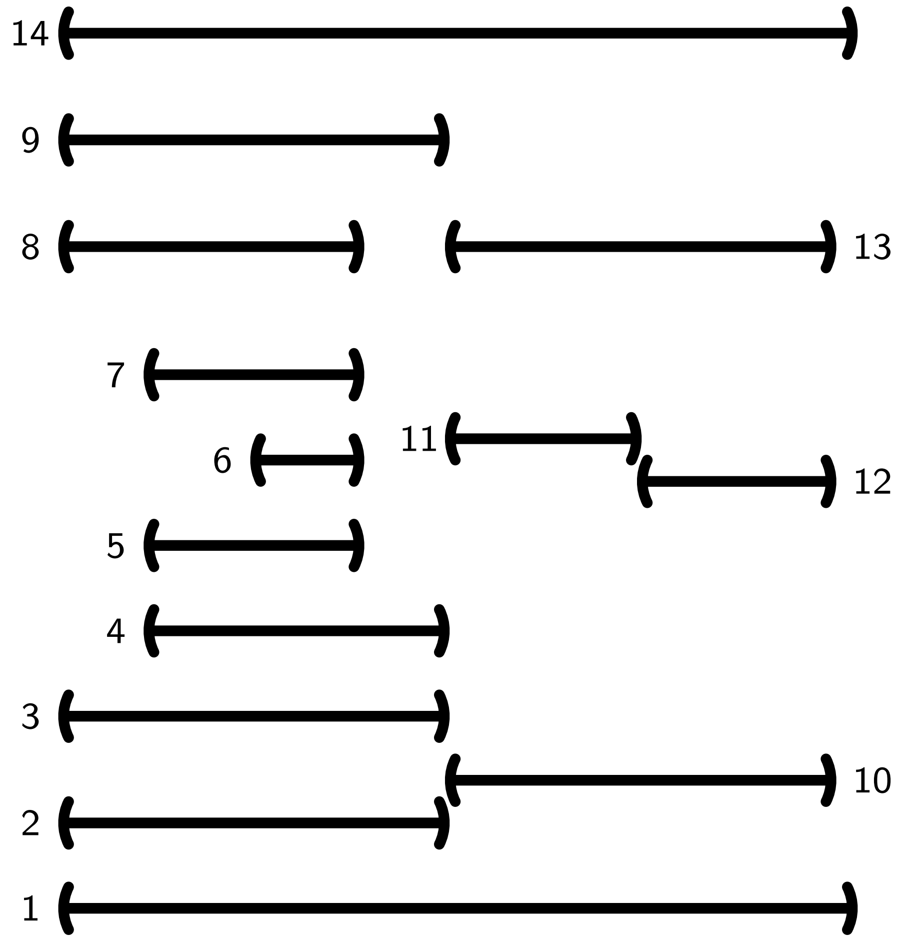


ε-bar visiblity and st-graphs

Recall that an **st-graph** is a planar digraph G with exactly one soure 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.



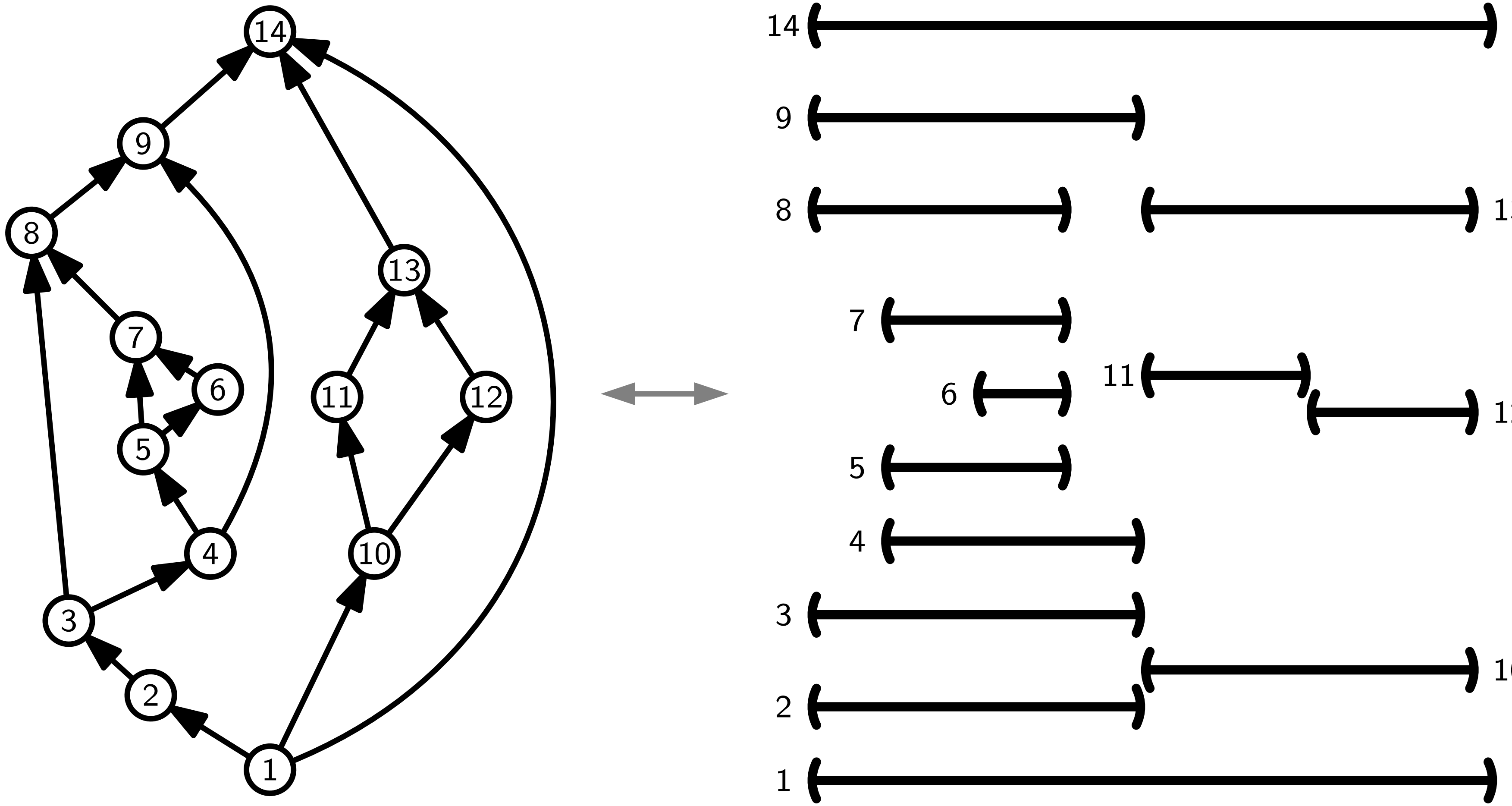


ε-bar visiblity and st-graphs

Recall that an **st-graph** is a planar digraph G with exactly one soure 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.

Note that testing whether an acyclic planar **di**graph has a weak bar visibility representation is NP-complete.



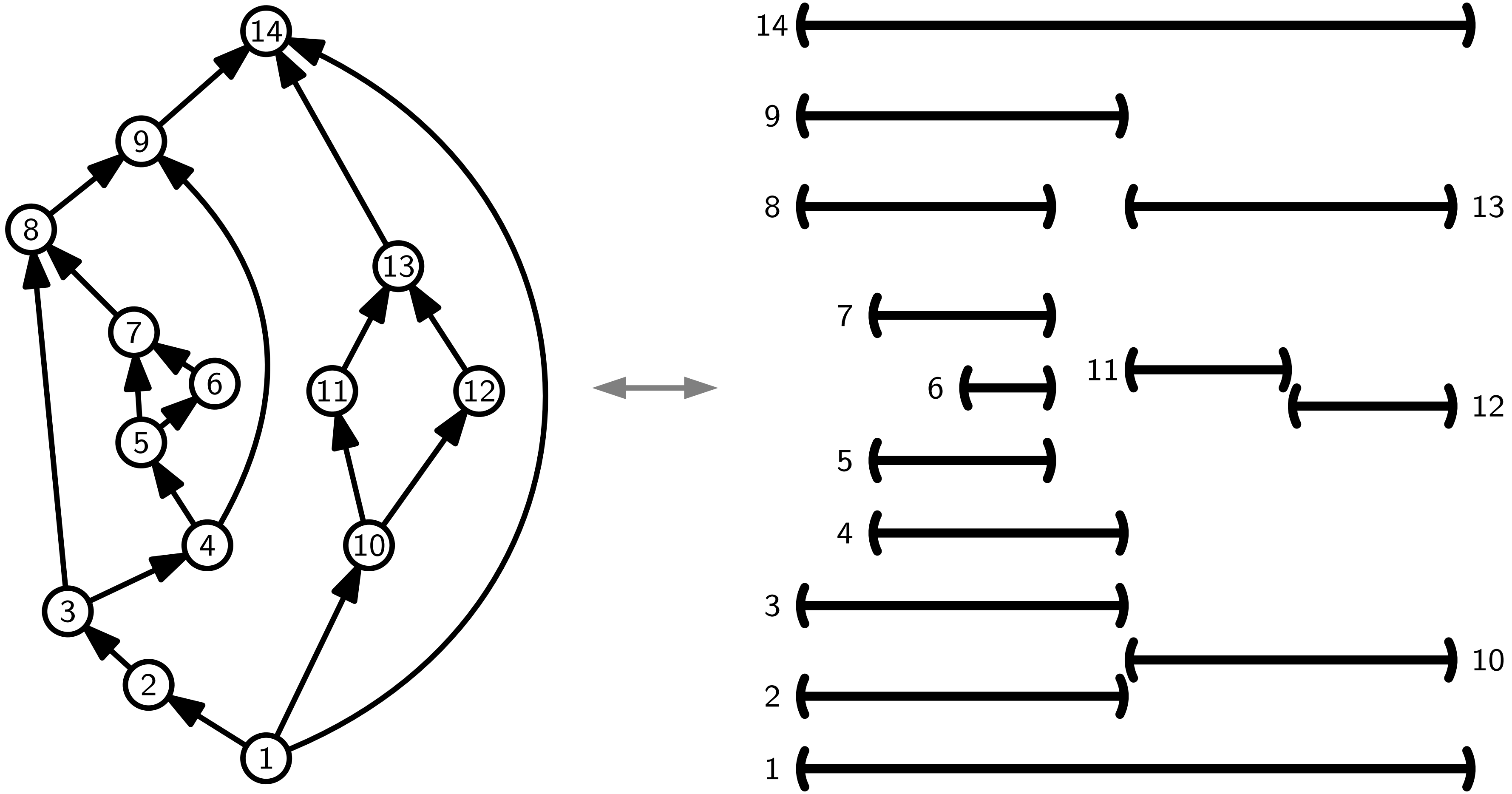
ε-bar visiblity and st-graphs

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

Note that testing whether an acyclic planar **di**graph has a weak bar visiblity representation is NP-complete.

- This is upward planarity testing! [Garg & Tamassia '01]

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

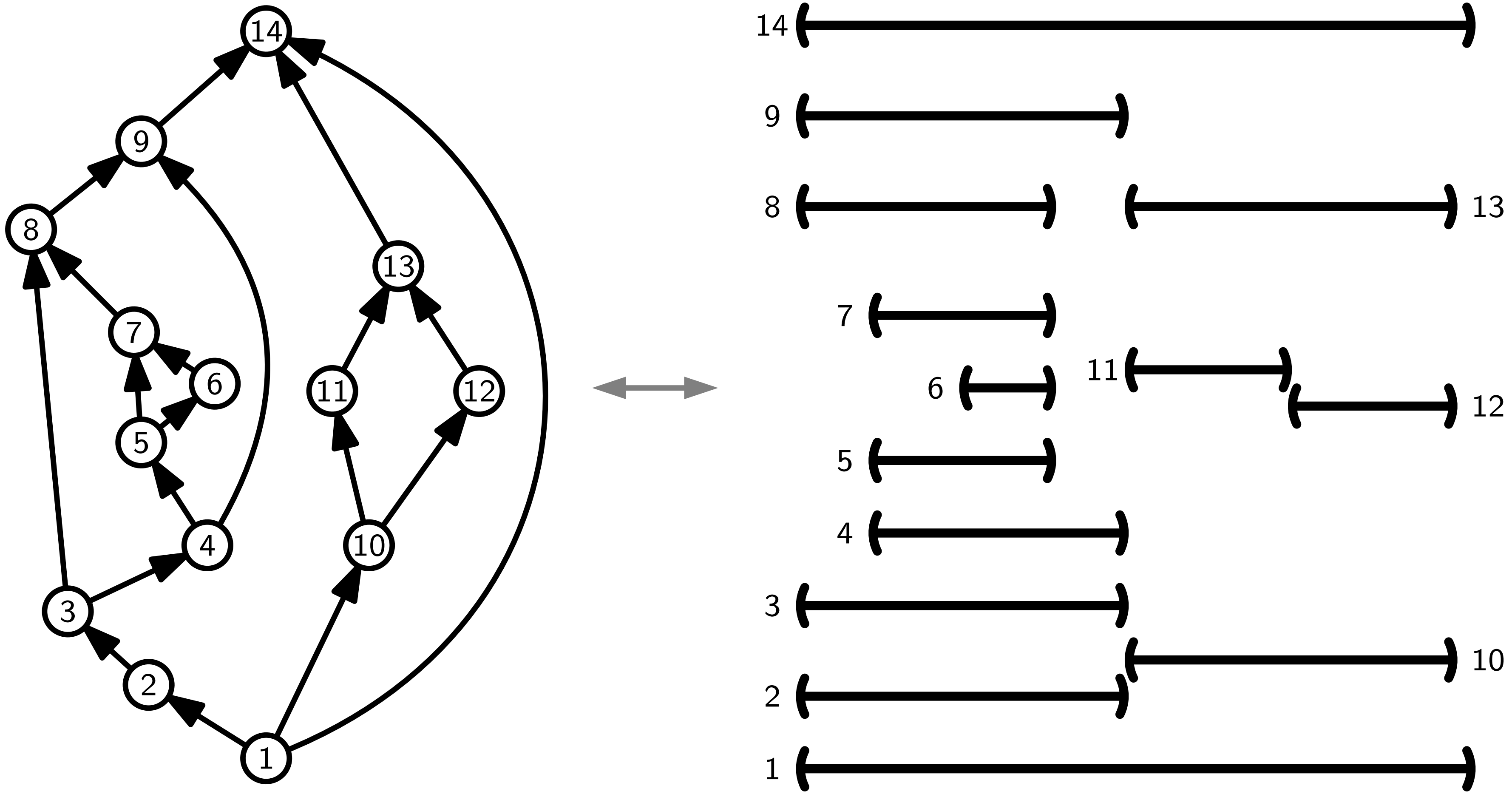


ε-bar visiblity and st-graphs

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

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

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

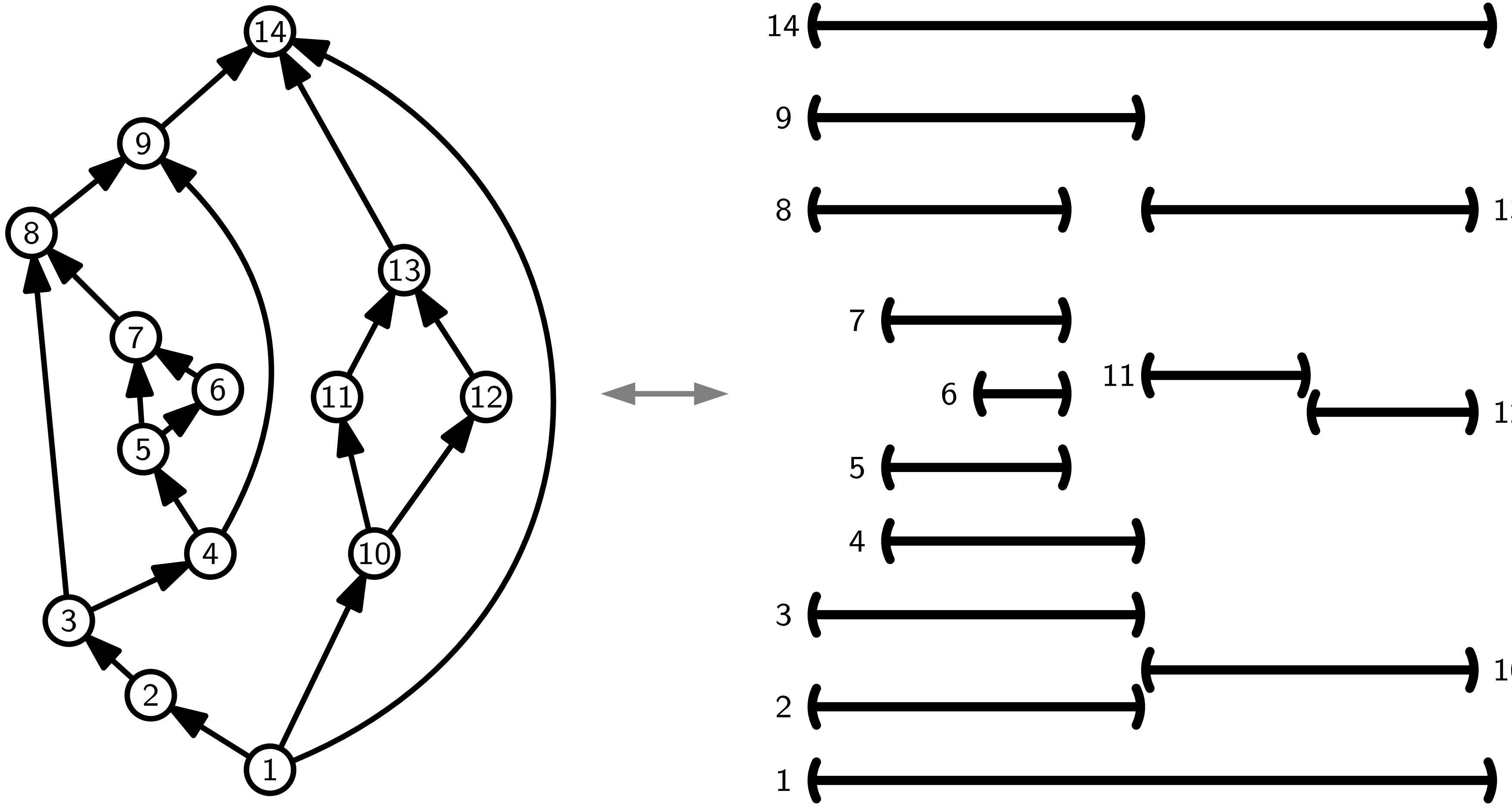


ε-bar visiblity and st-graphs

Recall that an **st-graph** is a planar digraph G with exactly one soure 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 visability testing is easily done via st-graph recognition.
- Strong bar visability recognition. . . open?

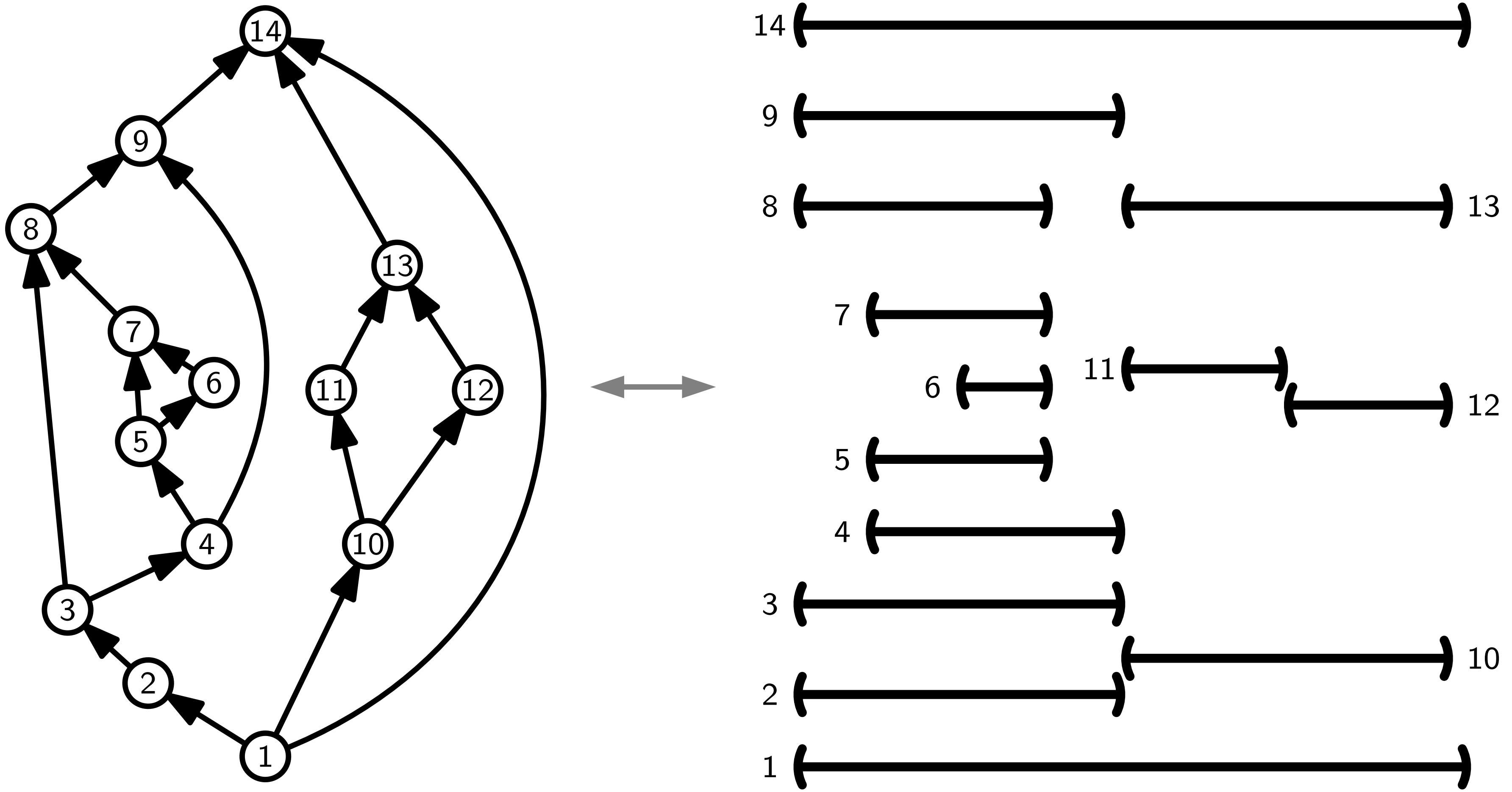


ε-bar visiblity and st-graphs

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

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

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



Results and outline

Theorem 1. [Chaplick et al. '18]
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

Theorem 1. [Chaplick et al. '18]
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

Theorem 1. [Chaplick et al. '18]
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. [Chaplick et al. '18]
ε-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT

Results and outline

Theorem 1. [Chaplick et al. '18]
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. [Chaplick et al. '18]
ε-Bar Visibility Representation Ext. is NP-complete.

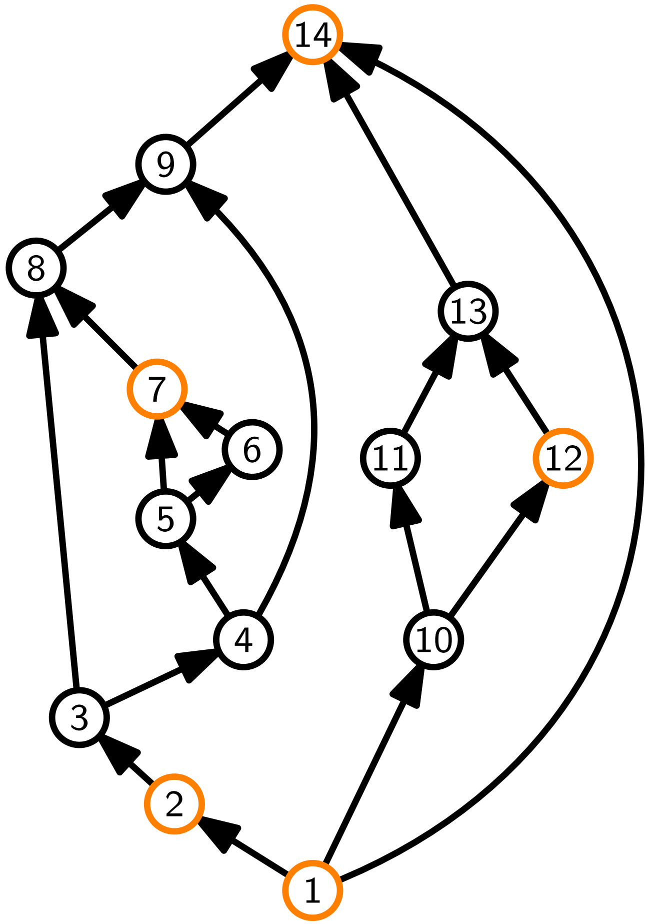
- Reduction from Planar Monotone 3-SAT

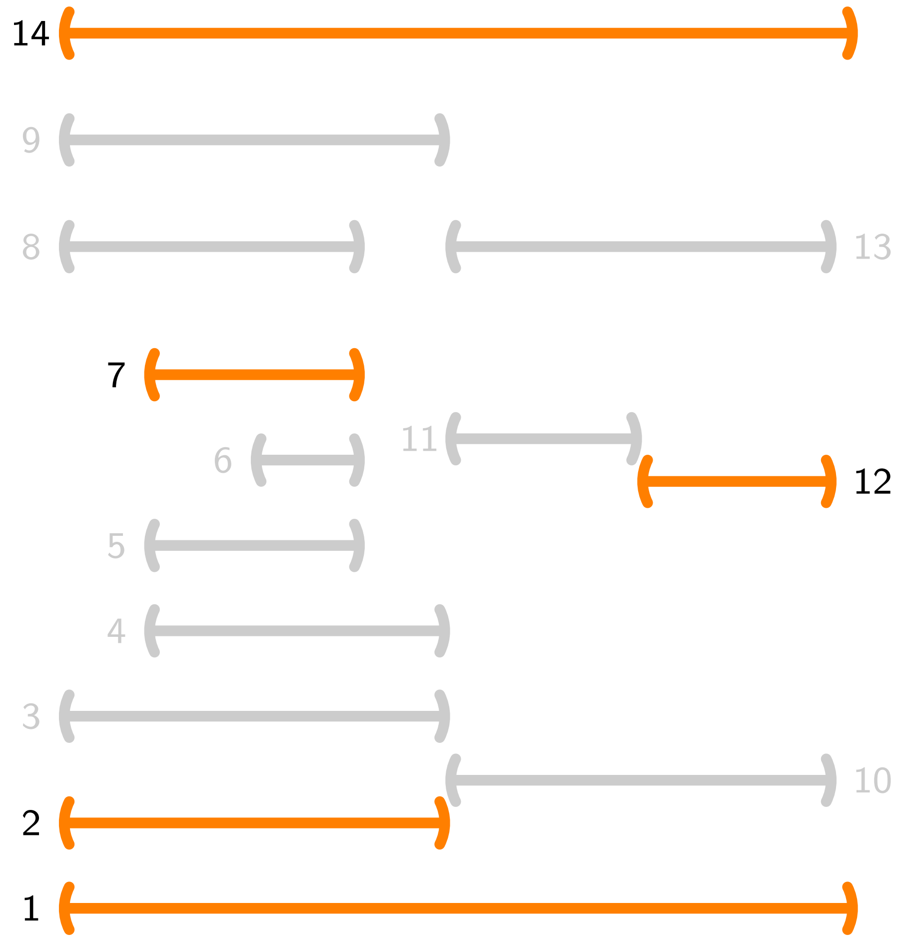
Theorem 3. [Chaplick et al. '18]
ε-Bar Visibility Representation Ext. is NP-complete for (series-parallel) st-graphs when restricted to the **integer grid** (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from 3-Partition

Representation extension for st-graphs

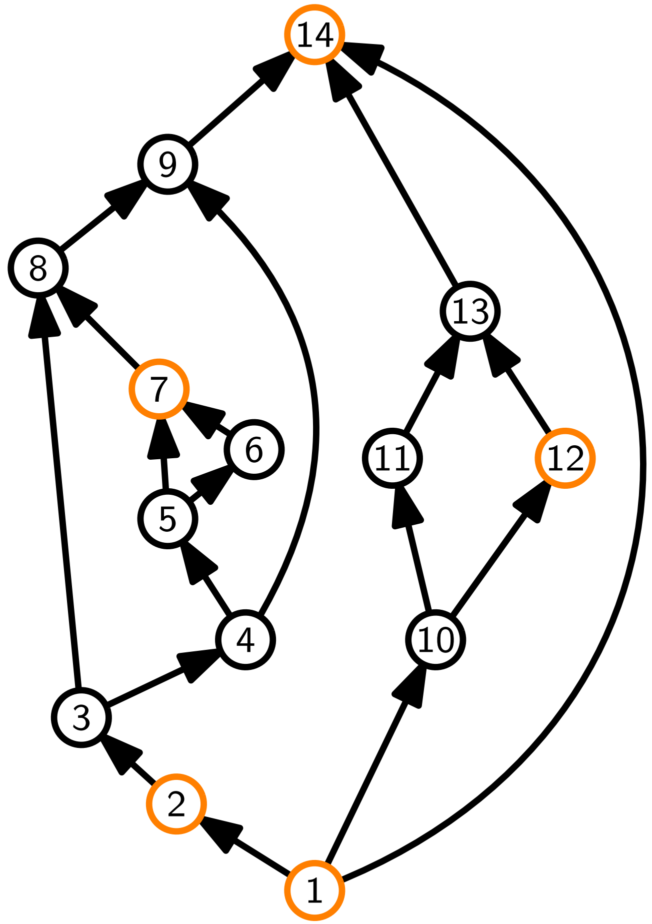
Theorem 1'.
Rectangular ε-Bar Visibility Representation Extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.



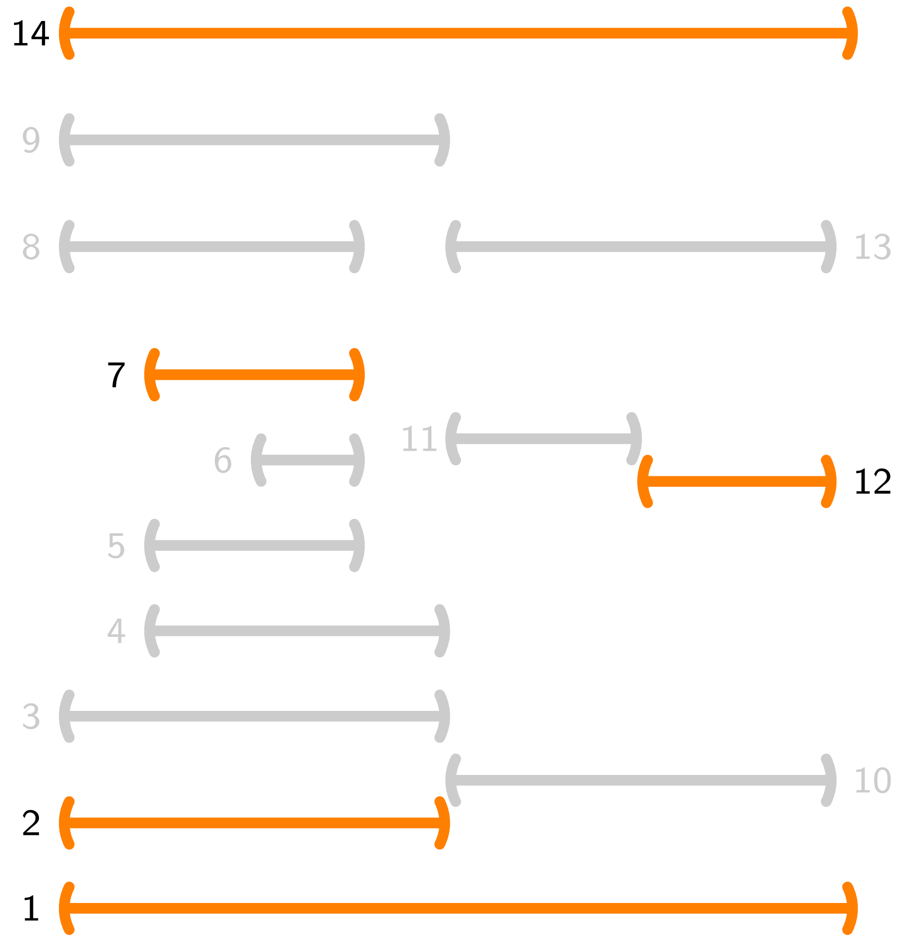


Representation extension for st-graphs

Theorem 1'.
Rectangular ε-Bar Visibility Representation Extension can be solved in $\mathcal{O}(n^2)$ time for st-graphs.



- Simplify with assumption on y-coordinates
- Look at connection to SPRQ-trees – tiling
- Solve problems for S, P and R nodes
- Dynamic program via SPQR-tree



y-coordinate invariant

- Let G be an st-graph, and ψ' be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \to \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the y-coordinate of $\psi'(v)$.
  - for each edge (u, v), $y(u) < y(v)$.

y-coordinate invariant

- Let G be an st-graph, and ψ' be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \to \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the y-coordinate of $\psi'(v)$.
  - for each edge (u, v), $y(u) < y(v)$.

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

y-coordinate invariant

- Let G be an st-graph, and ψ' be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \rightarrow \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the y-coordinate of $\psi'(v)$.
  - for each edge (u, v), $y(u) < y(v)$.

Lemma 1.
G has a representation extending ψ' iff
G has a representation ψ 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

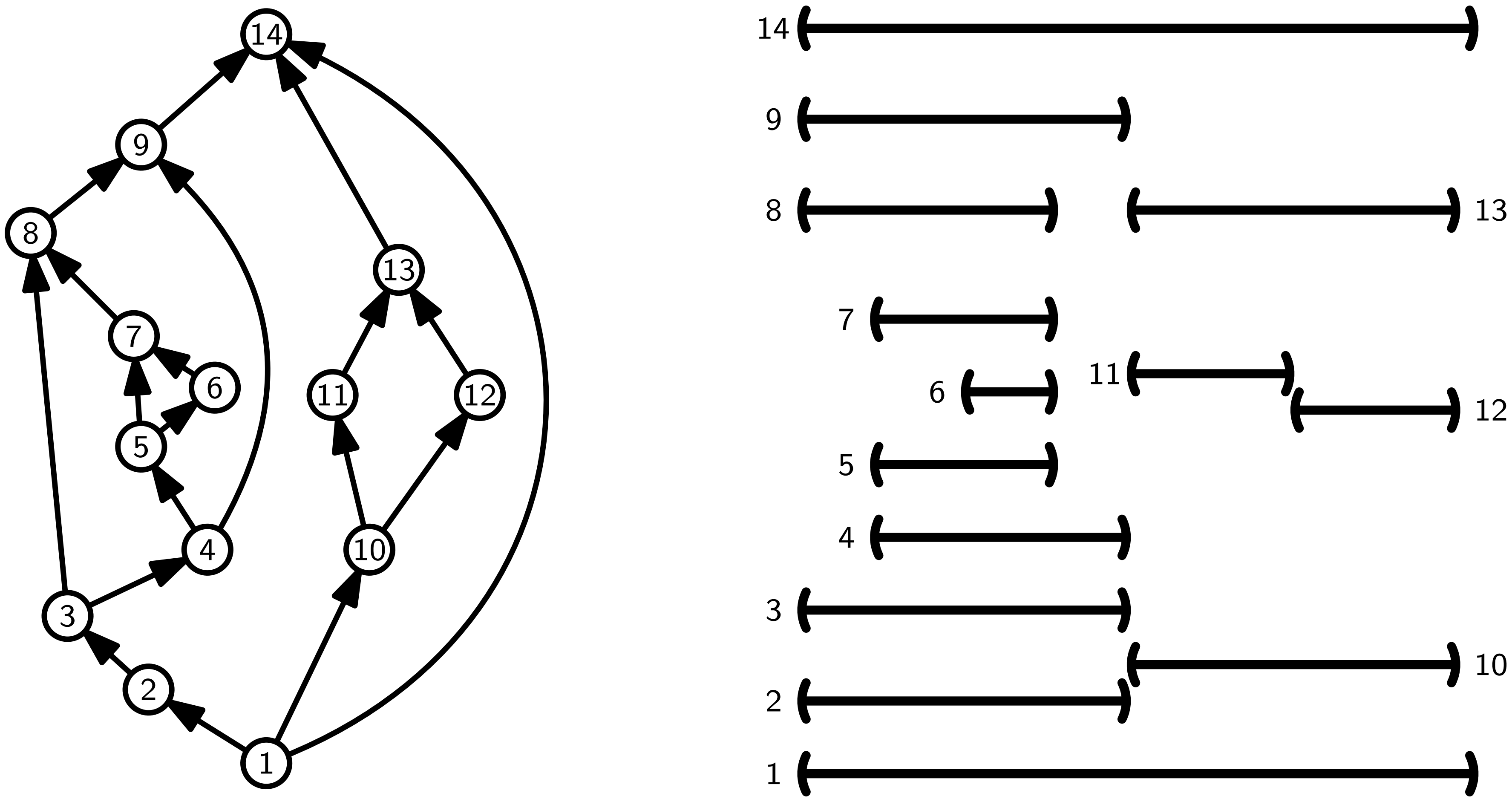
- Let G be an st-graph, and ψ' be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \to \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the y-coordinate of $\psi'(v)$.
  - for each edge (u, v), $y(u) < y(v)$.

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

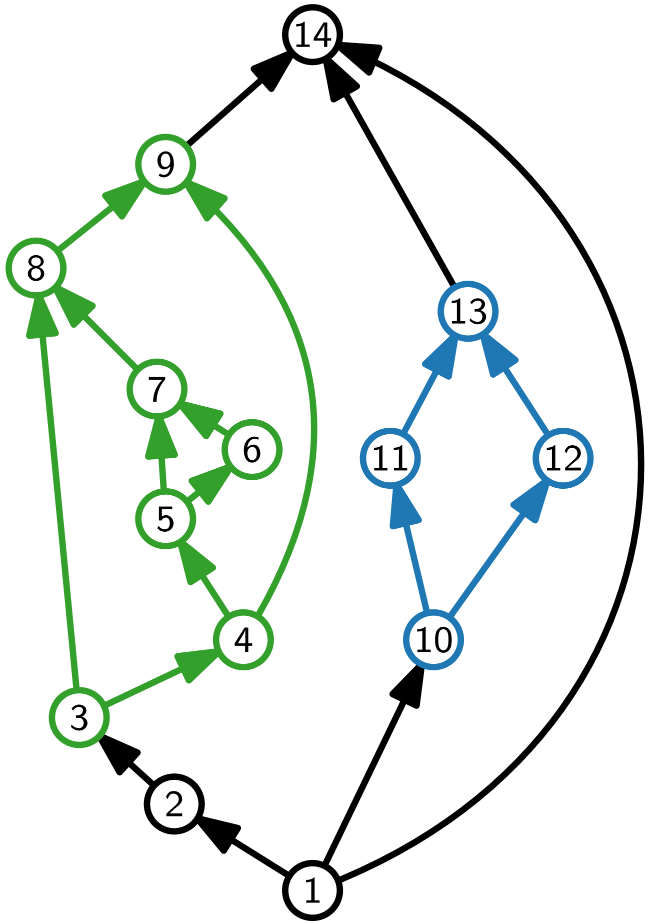
We can now assume all y-coordinates are given!

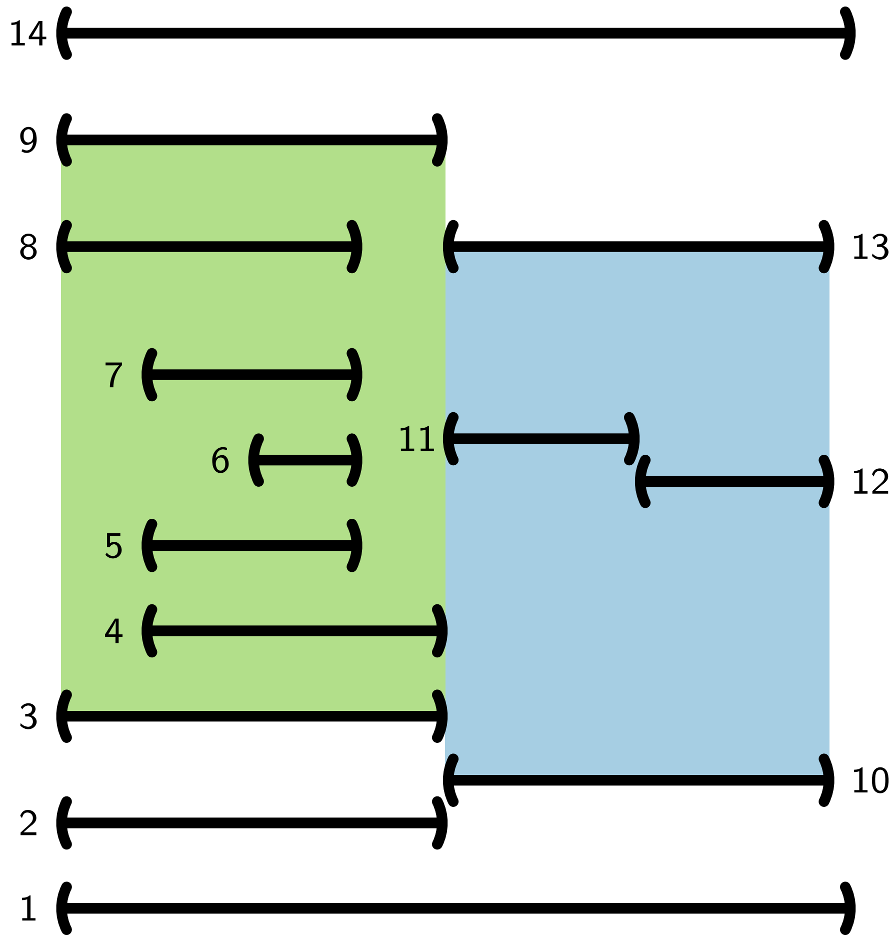
Proof idea. The relative positions of **adjacent** bars must match the order given by y.
So, we can adjust the y-coordinates of any solution to be as in y by sweeping from bottom-to-top.

But why do SPQR-trees help?



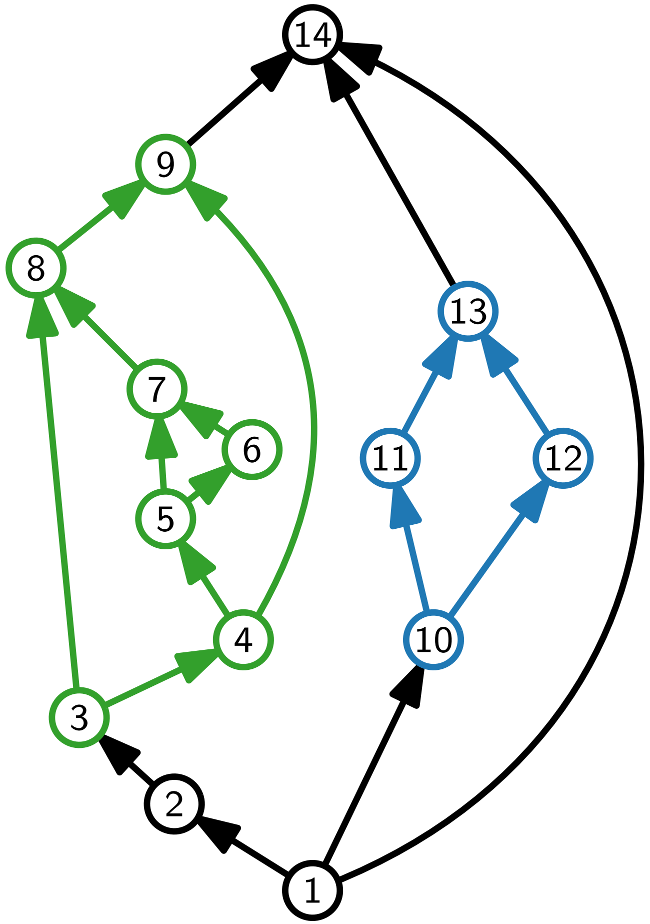
But why do SPQR-trees help?

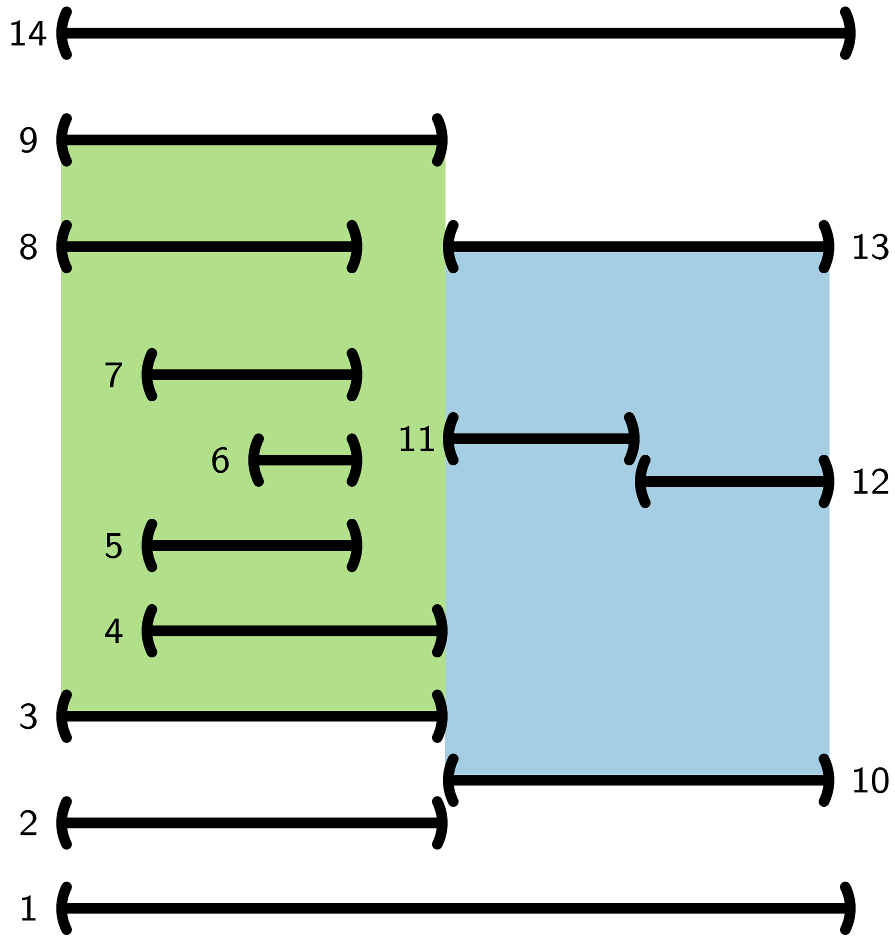




But why do SPQR-trees help?

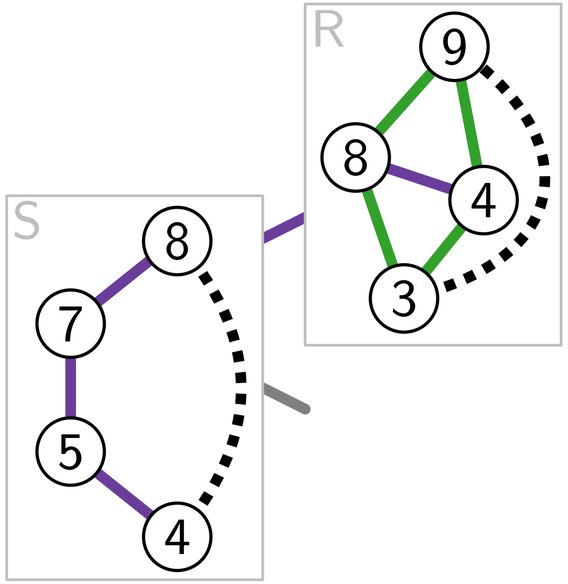
Lemma 2. The SPQR-tree of an st-graph G induces a recursive **tiling** of any ε-bar visibility representation of G.



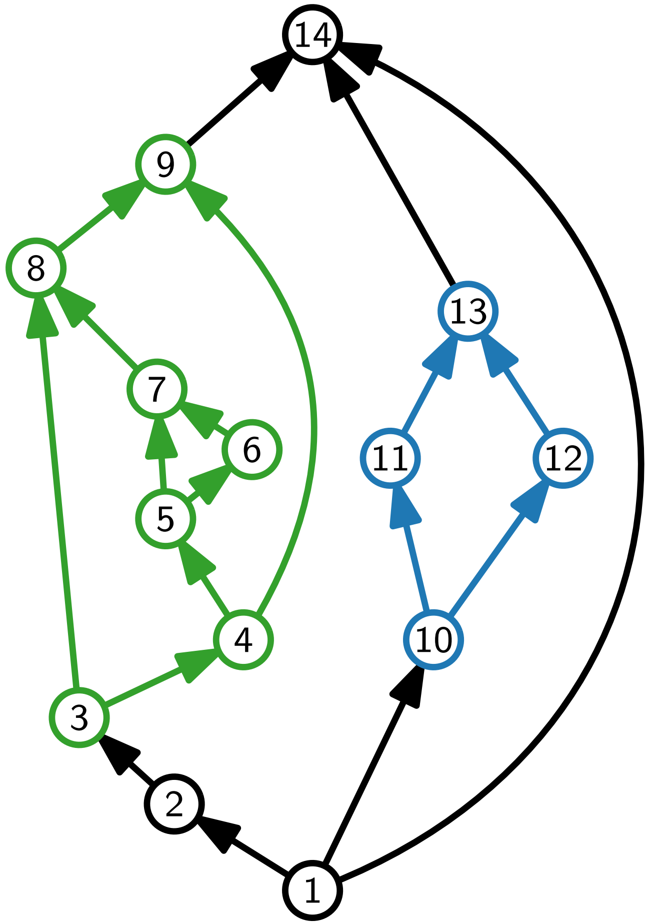


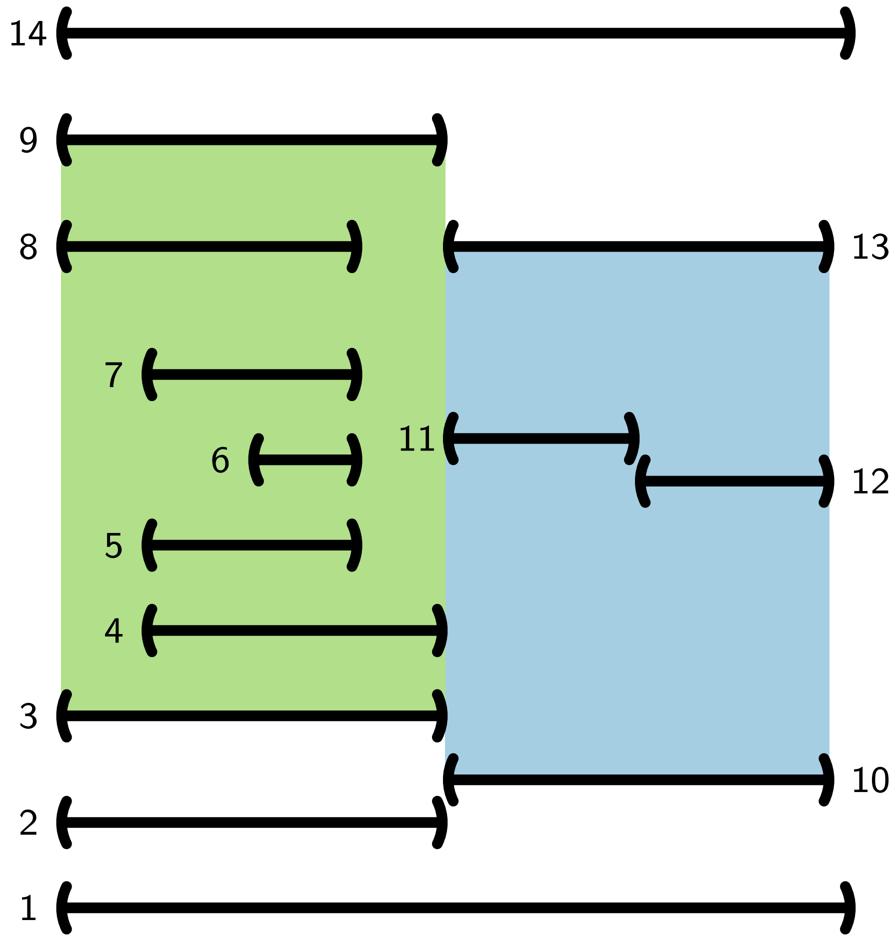
But why do SPQR-trees help?

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



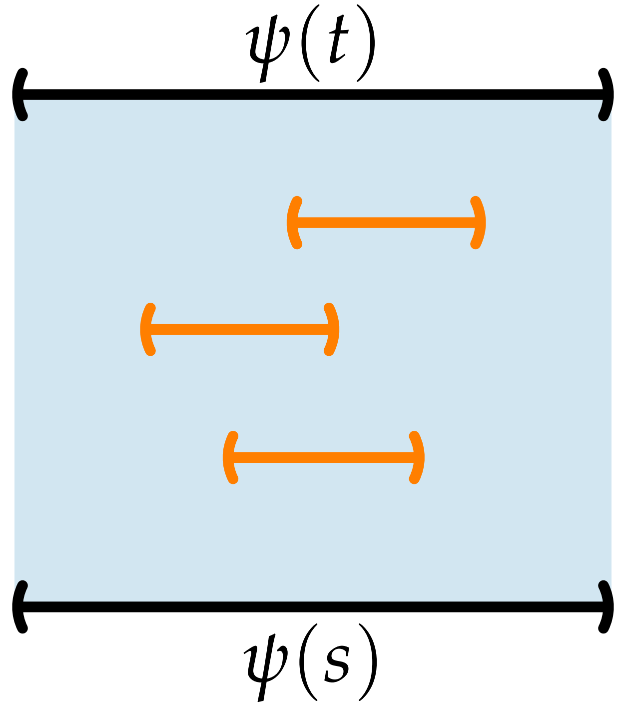
Solve tiles bottom-up





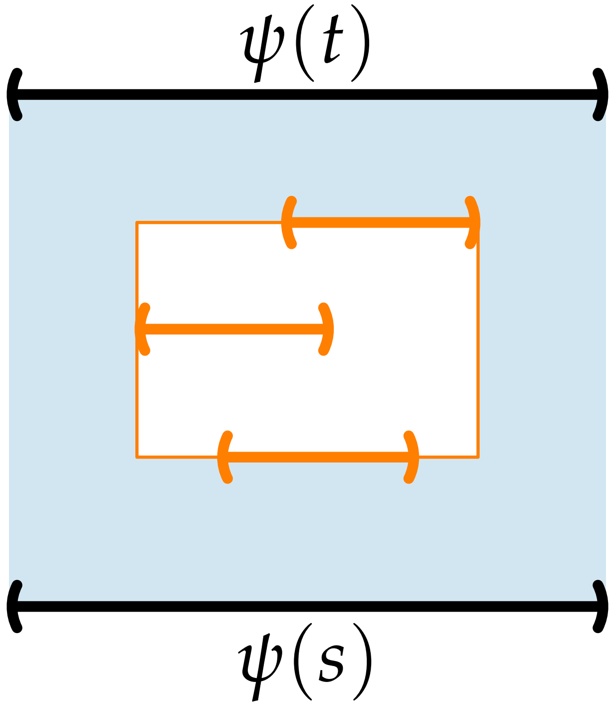
Tiles

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



Tiles

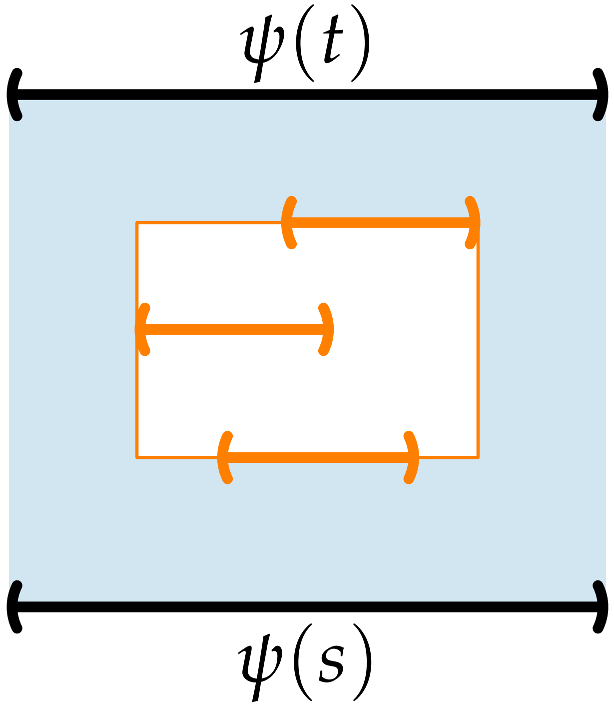
Convention. **Orange** bars are from the partial representation



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

Tiles

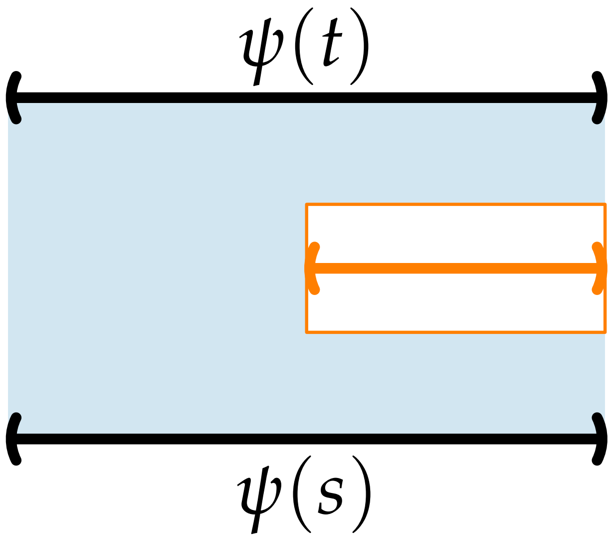
Convention. **Orange** bars are from the partial representation



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

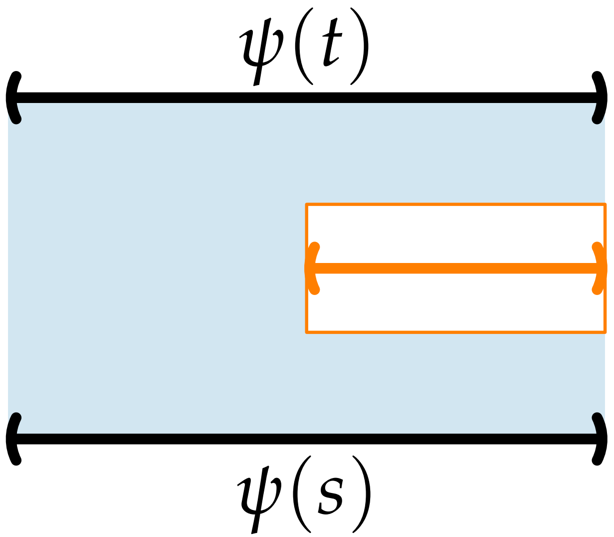
How many **different** tiles can we really have?

Types of tiles



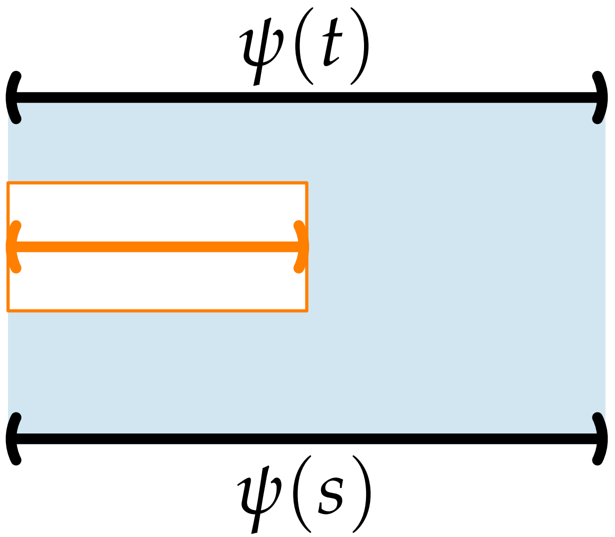
- Right **F**ixed – due to the orange bar
- Left **L**oose – due to the orange bar

Types of tiles

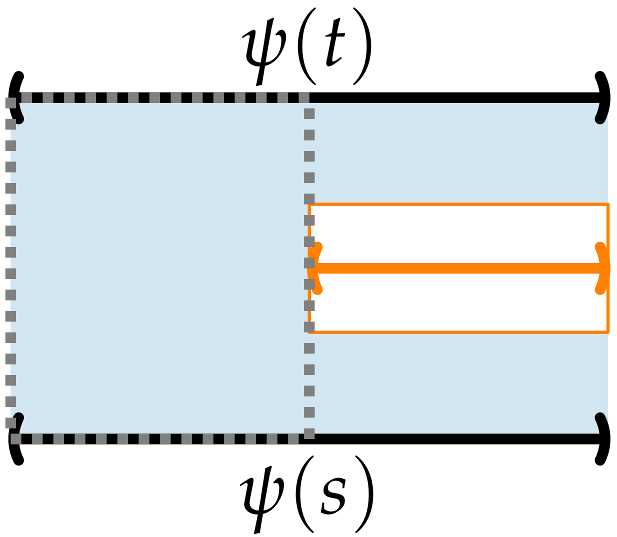


- Right **F**ixed – due to the orange bar
- Left **L**oose – due to the orange bar

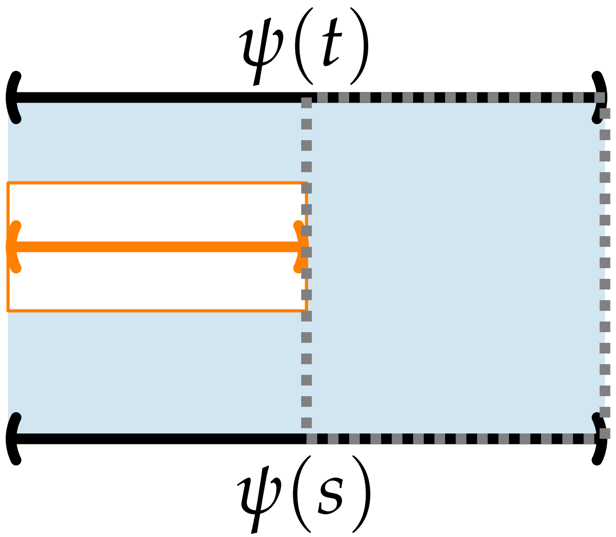
- Left **F**ixed – due to the orange bar
- Right **L**oose – due to the orange bar



Types of tiles

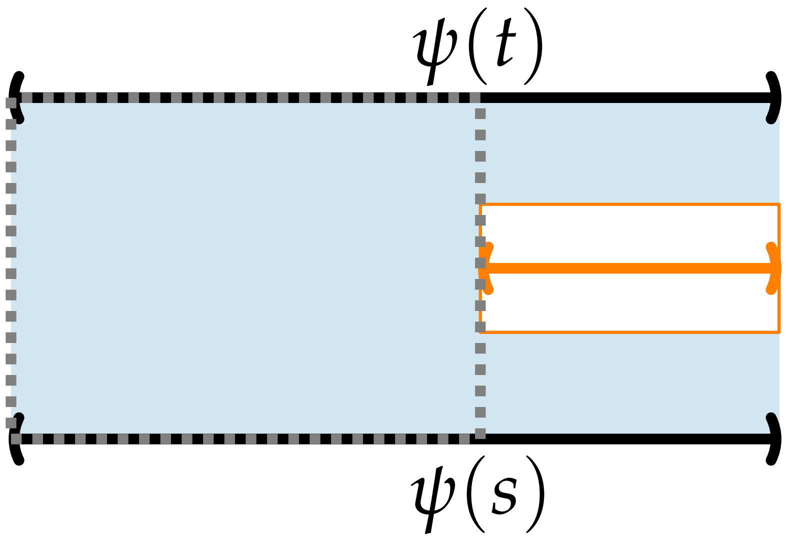


- Right **F**ixed – due to the orange bar
- Left **L**oose – due to the orange bar

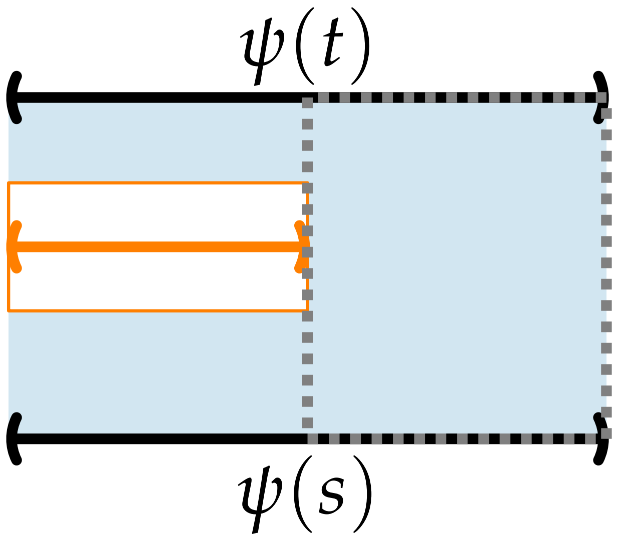


- Left **F**ixed – due to the orange bar
- Right **L**oose – due to the orange bar

Types of tiles

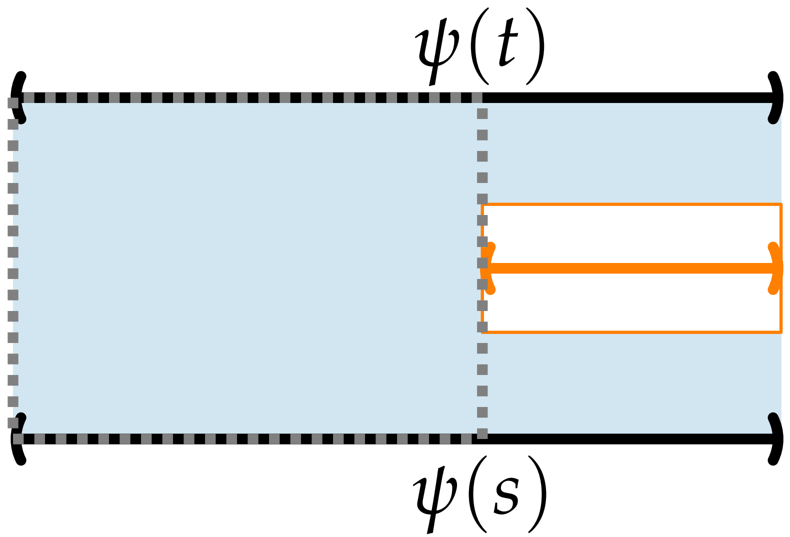


- Right **F**ixed – due to the orange bar
- Left **L**oose – due to the orange bar

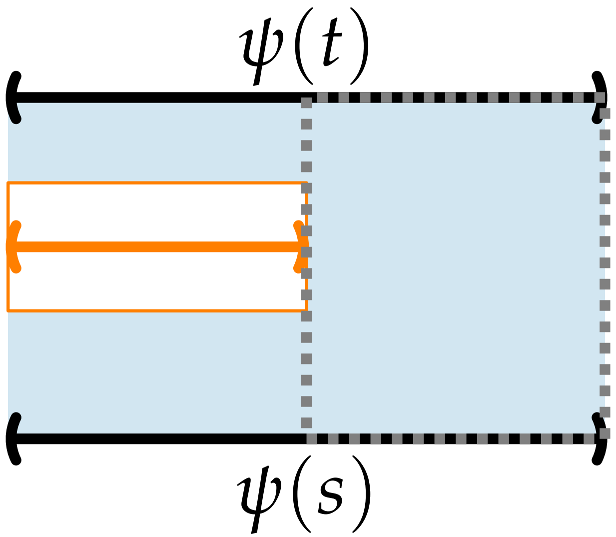


- Left **F**ixed – due to the orange bar
- Right **L**oose – due to the orange bar

Types of tiles



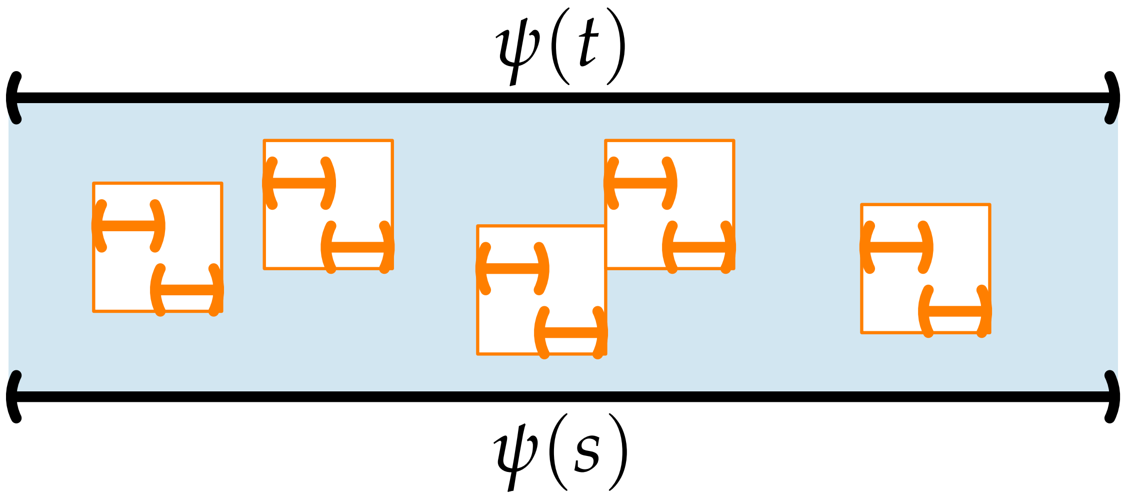
- Right **F**ixed – due to the orange bar
- Left **L**oose – due to the orange bar

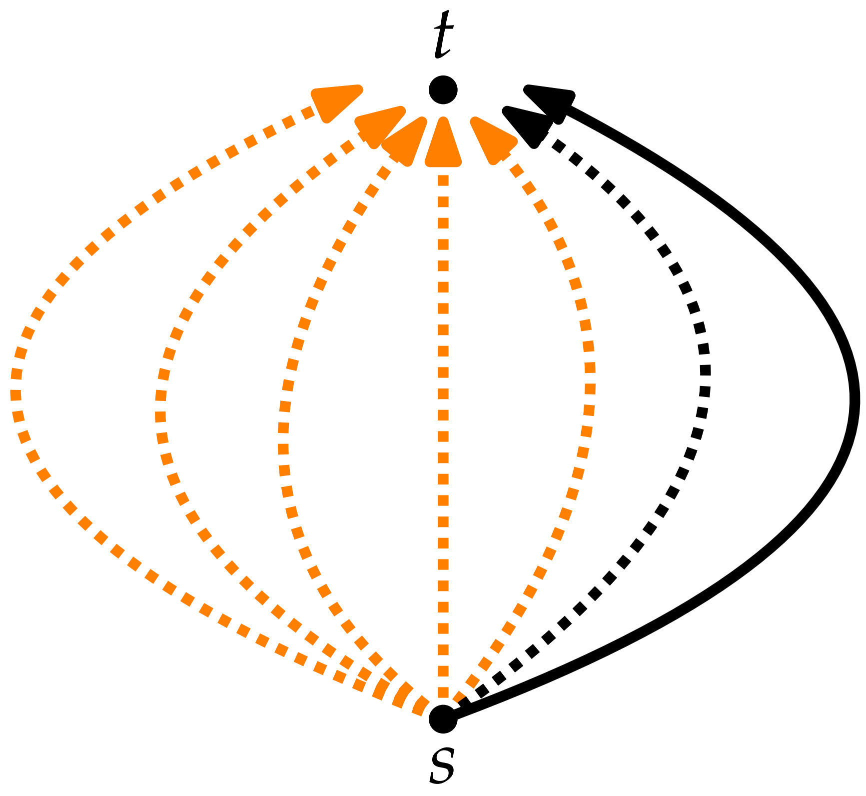


- Left **F**ixed – due to the orange bar
- Right **L**oose – due to the orange bar

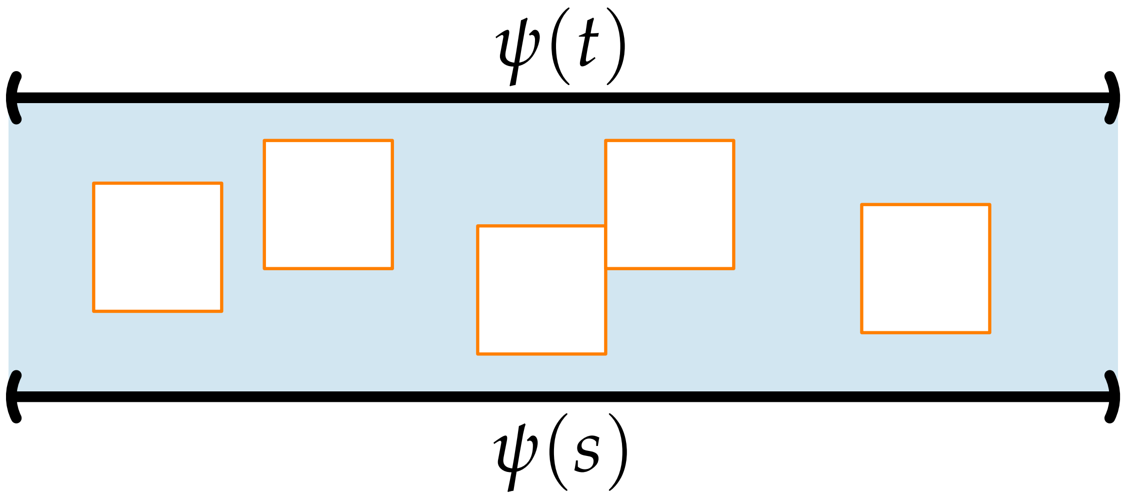
Four different types: **FF, FL, LF, LL**

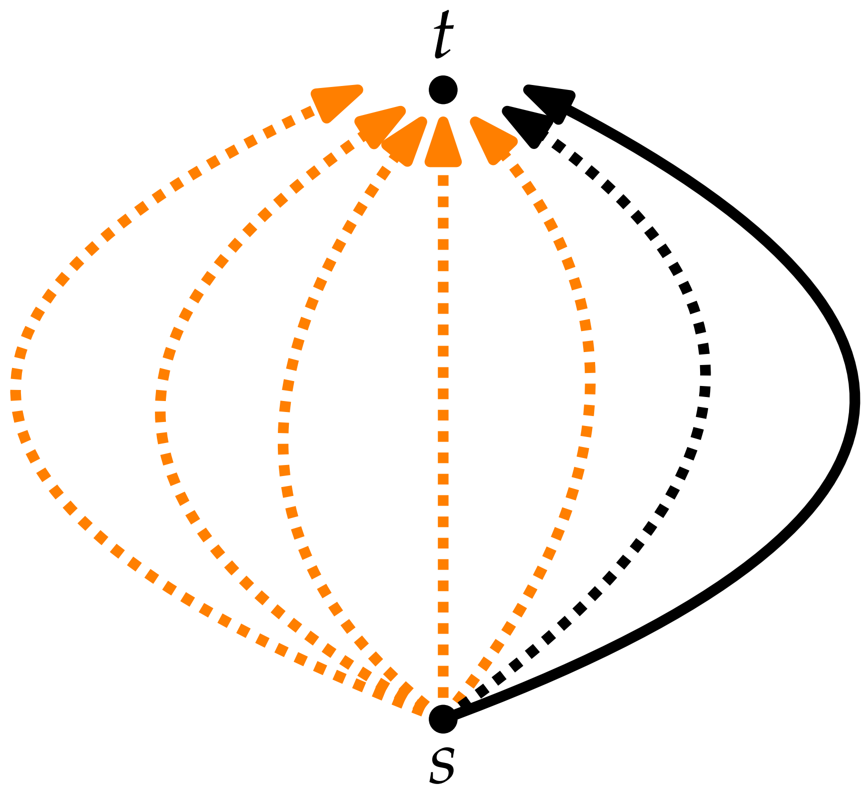
P nodes



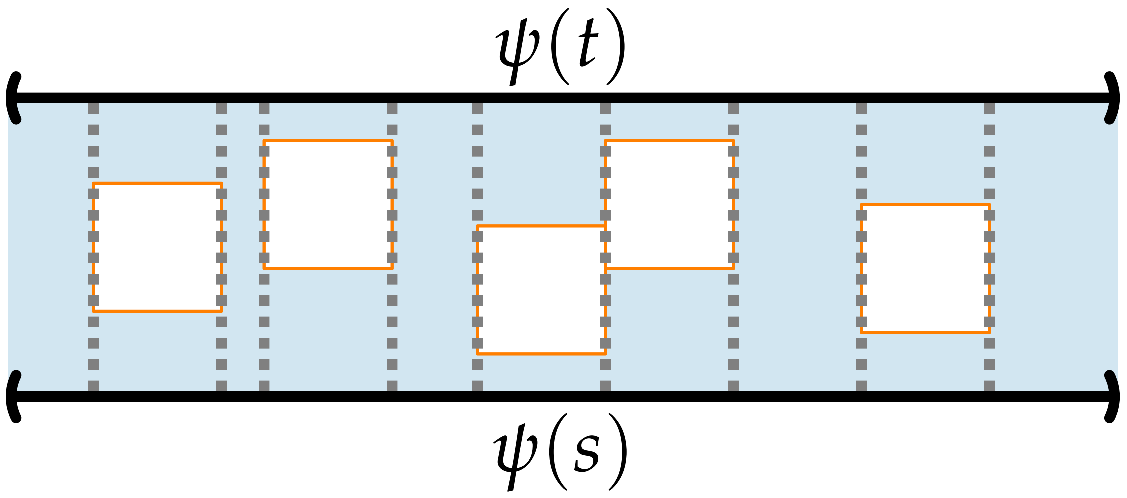


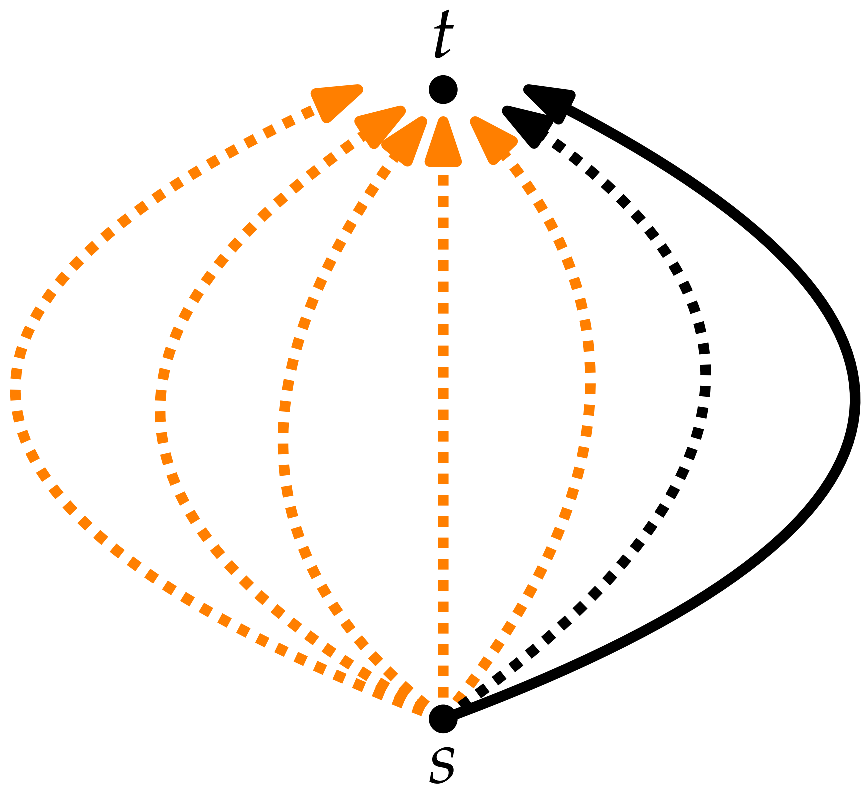
P nodes



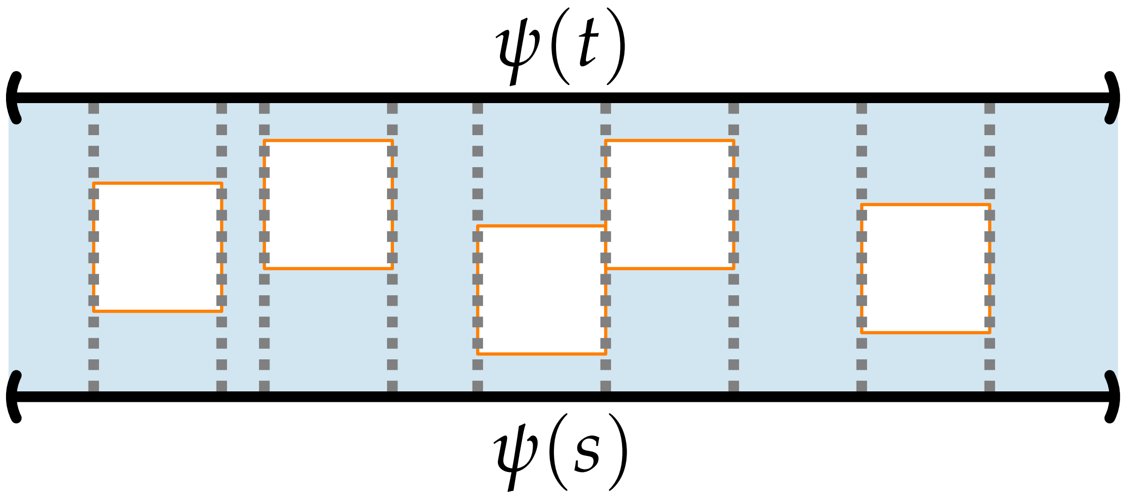


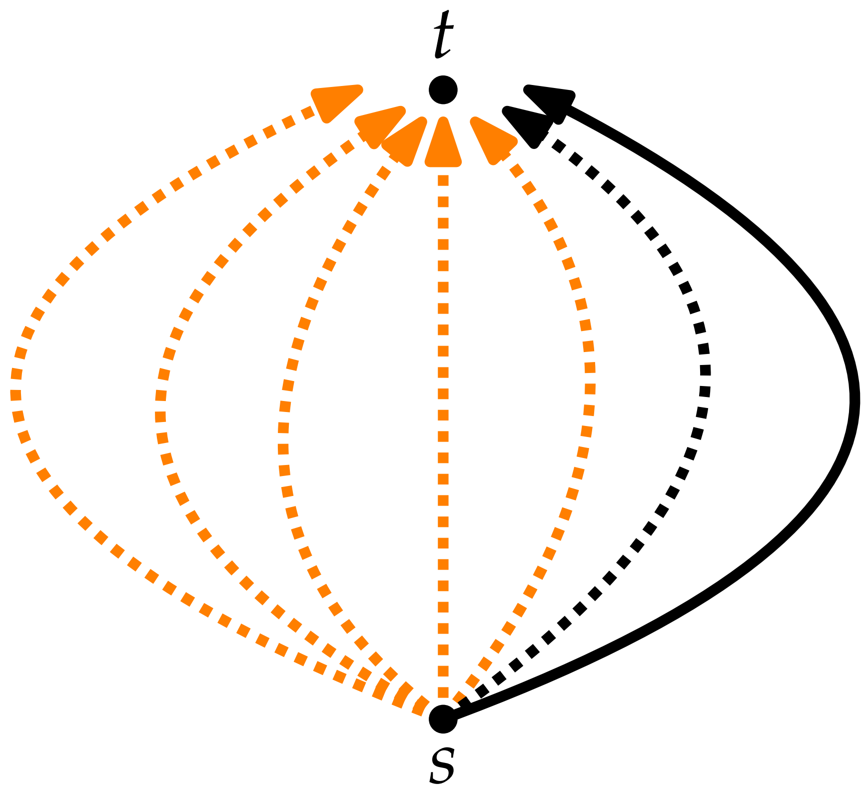
P nodes





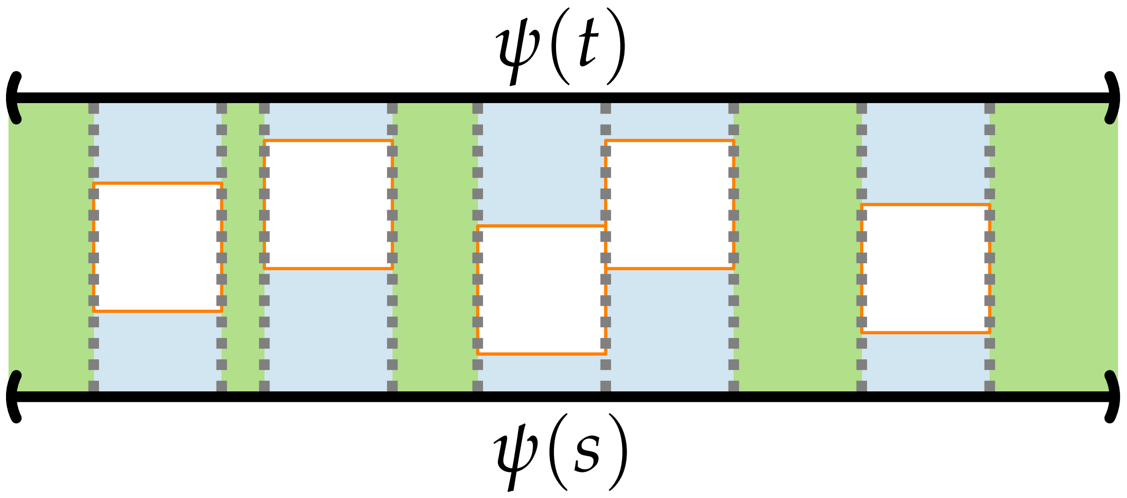
P nodes

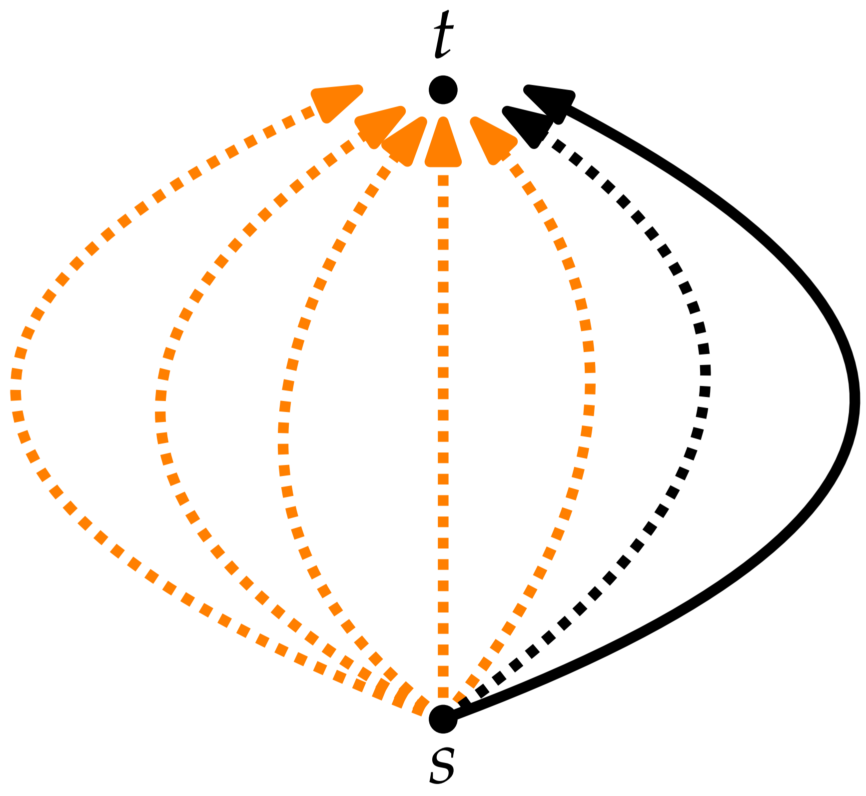




- Children of P node with prescribed bars occur in given left-to-right order
- But there might be some gaps. . .

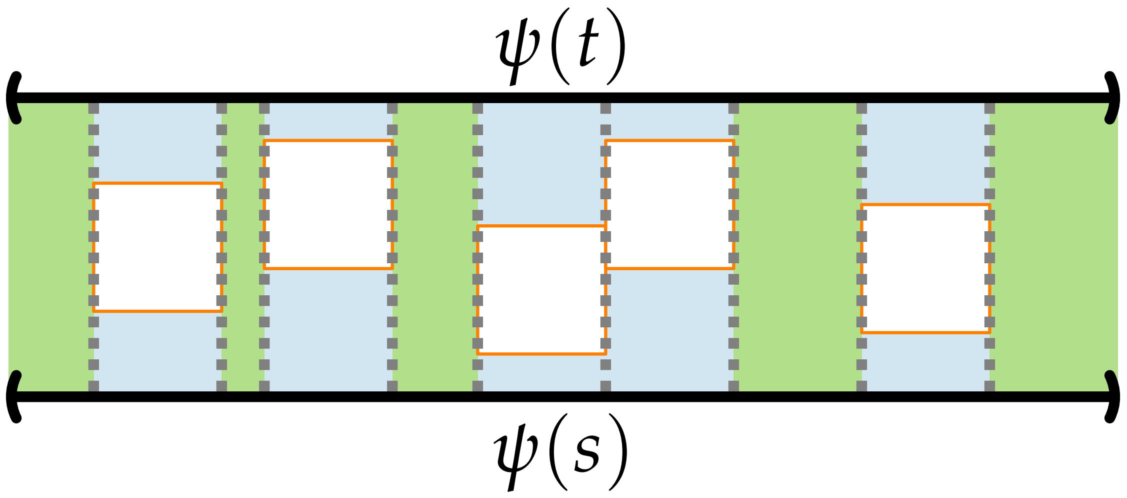
P nodes

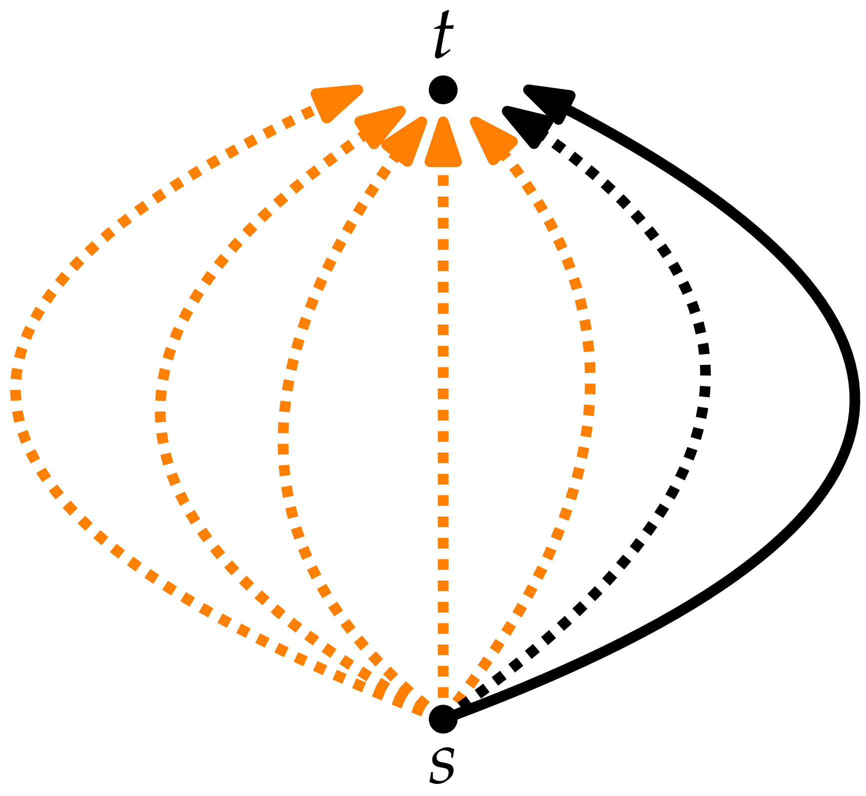




- Children of P node with prescribed bars occur in given left-to-right order
- But there might be some gaps...

P nodes

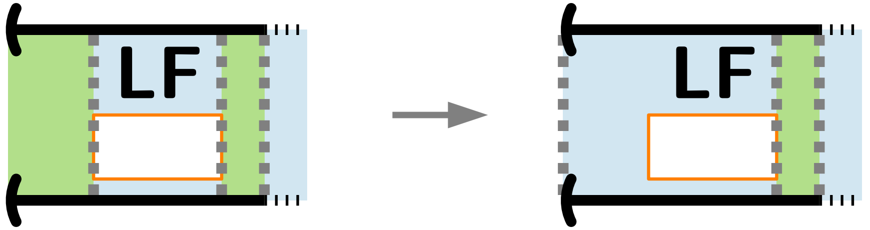




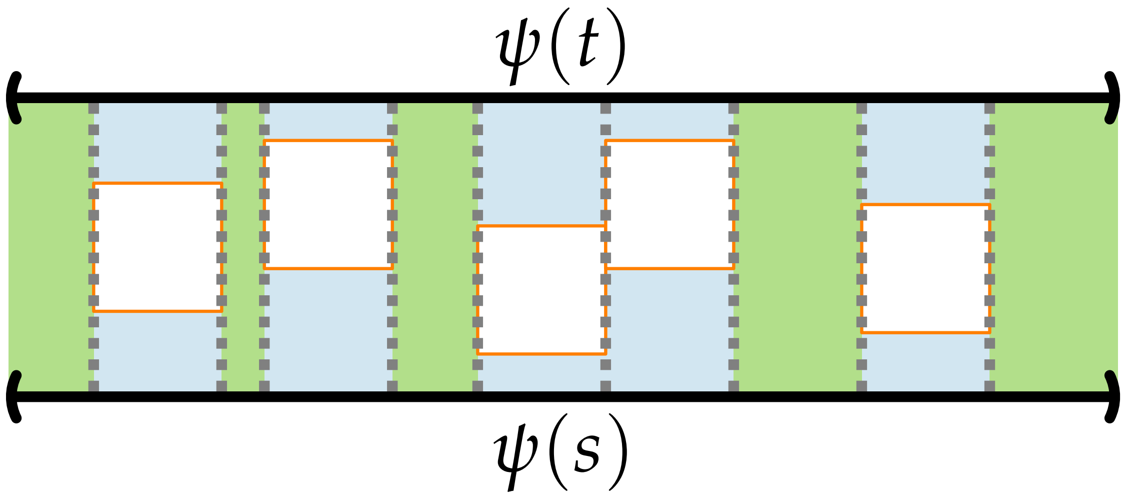
- Children of P node with prescribed bars occur in given left-to-right order
- But there might be some gaps. . .

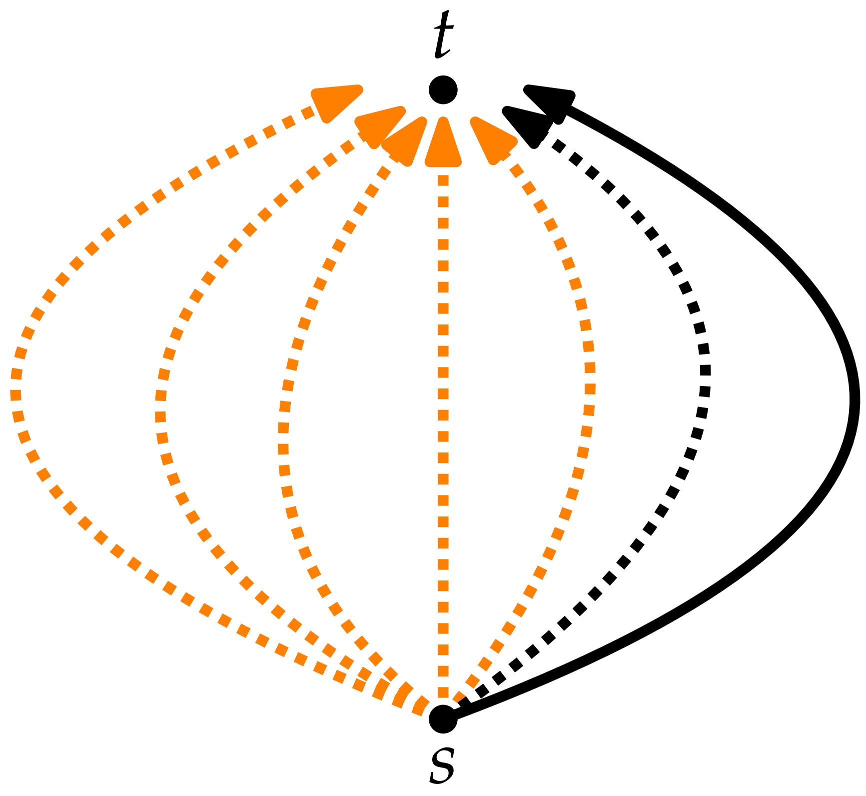
Idea.

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



P nodes

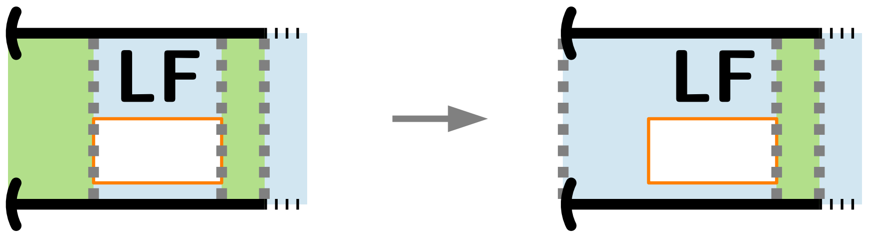




- Children of P node with prescribed bars occur in given left-to-right order
- But there might be some gaps. . .

Idea.

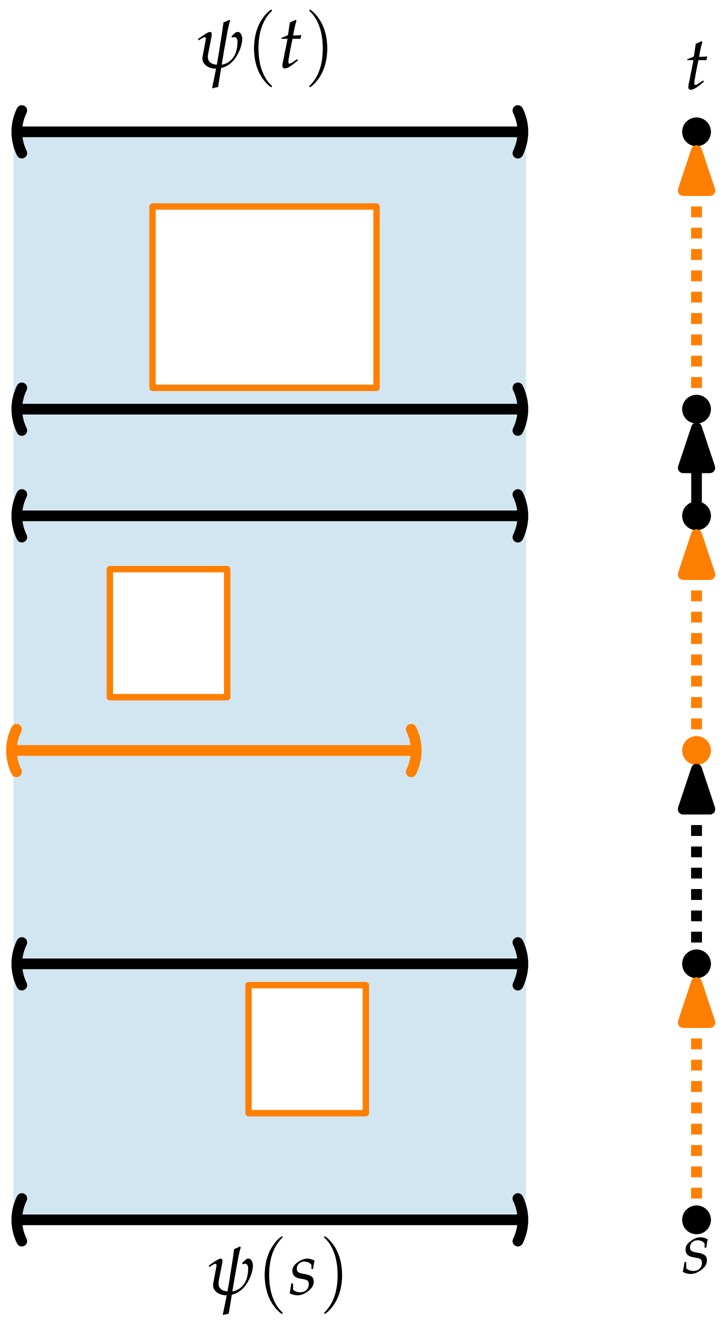
Greedily *fill* the gaps by preferring to "stretch" the children with prescribed bars.



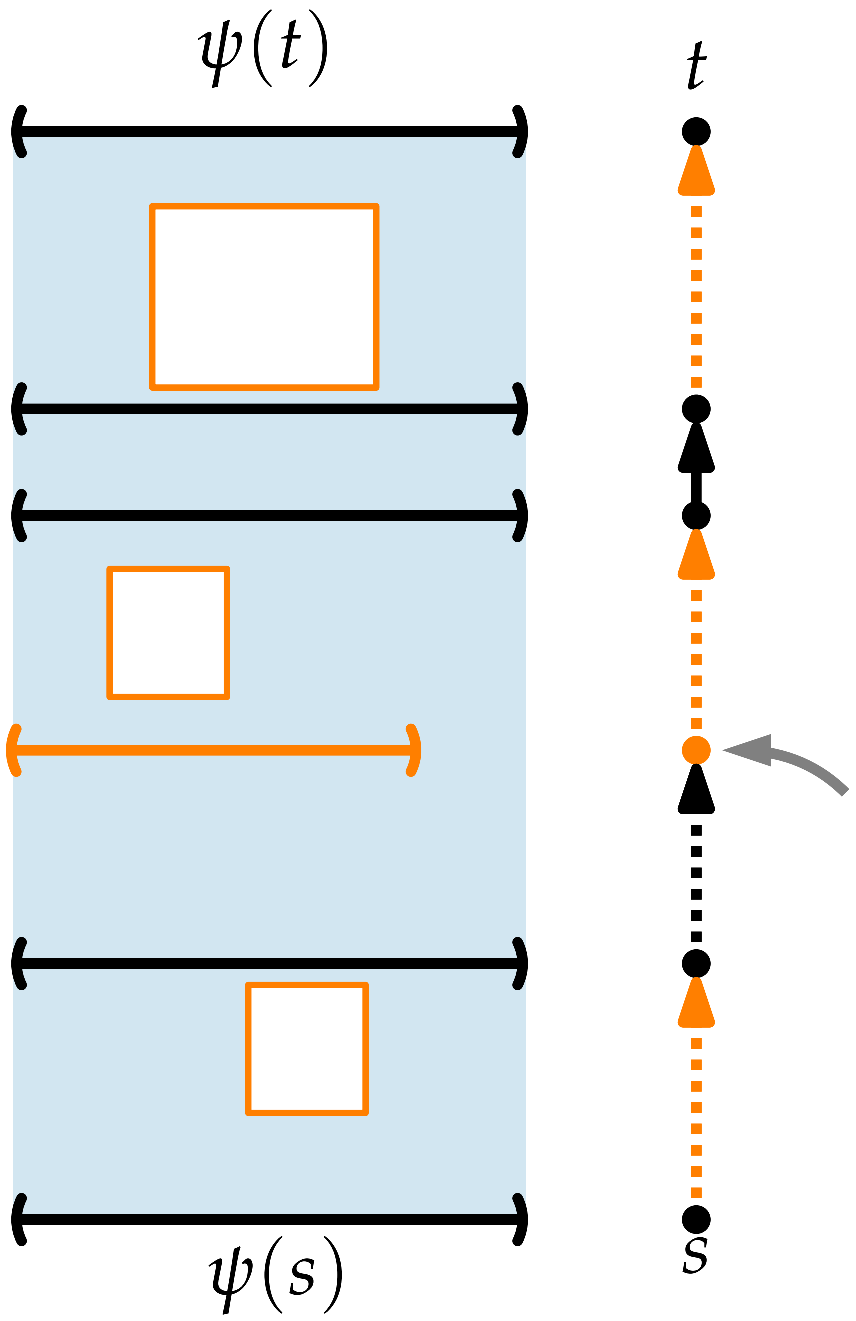
Outcome.

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

S nodes

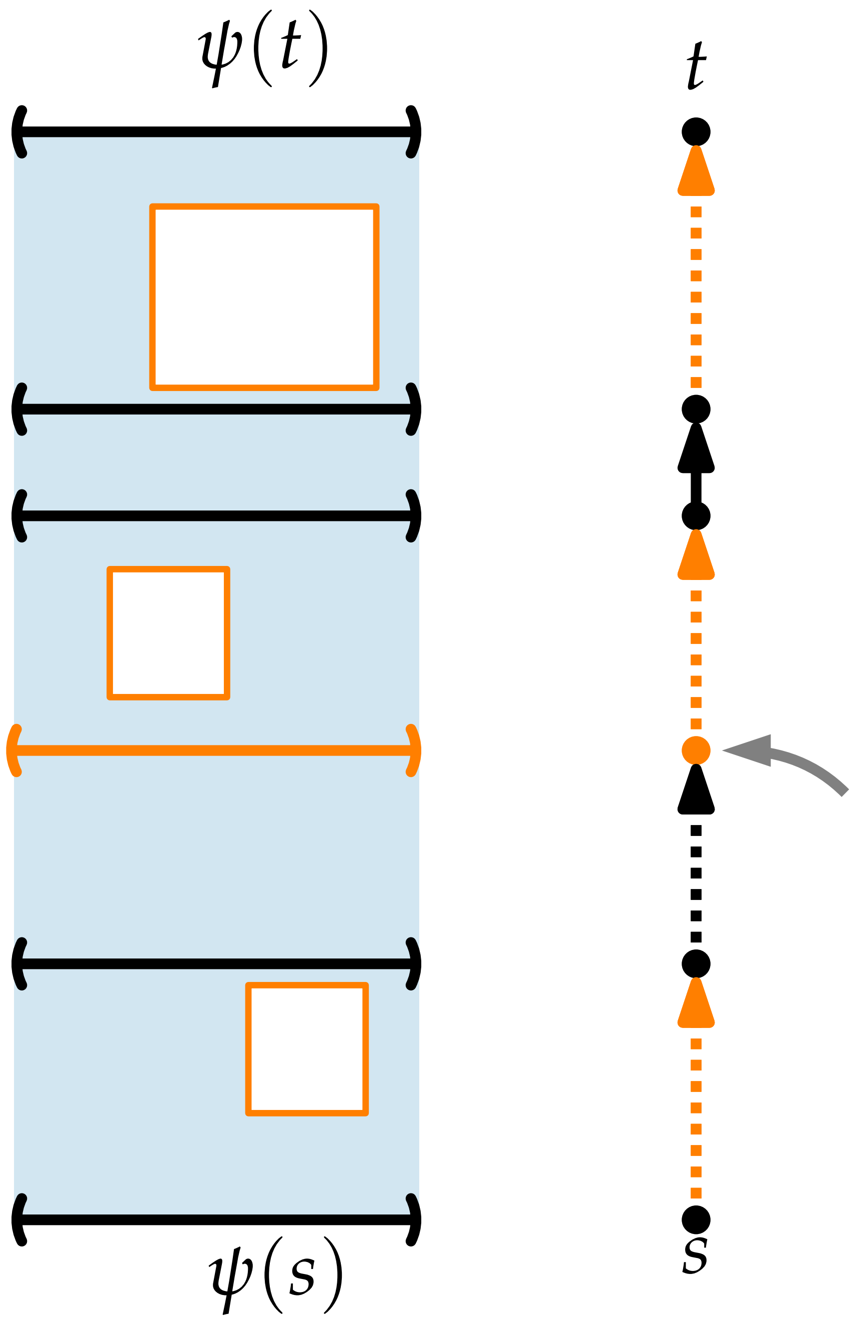


S nodes



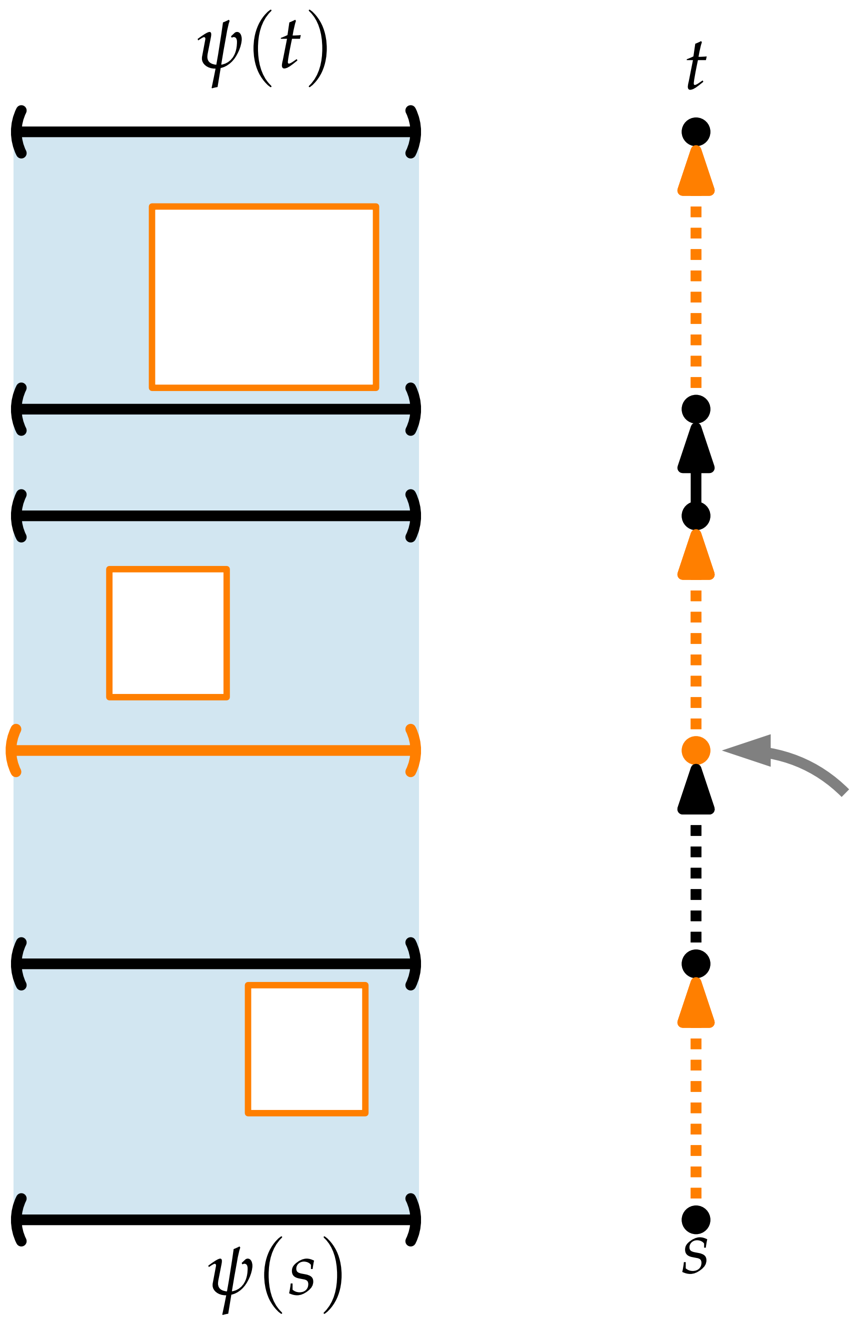
This fixed vertex means we can only make a Fixed-Fixed representation!

S nodes



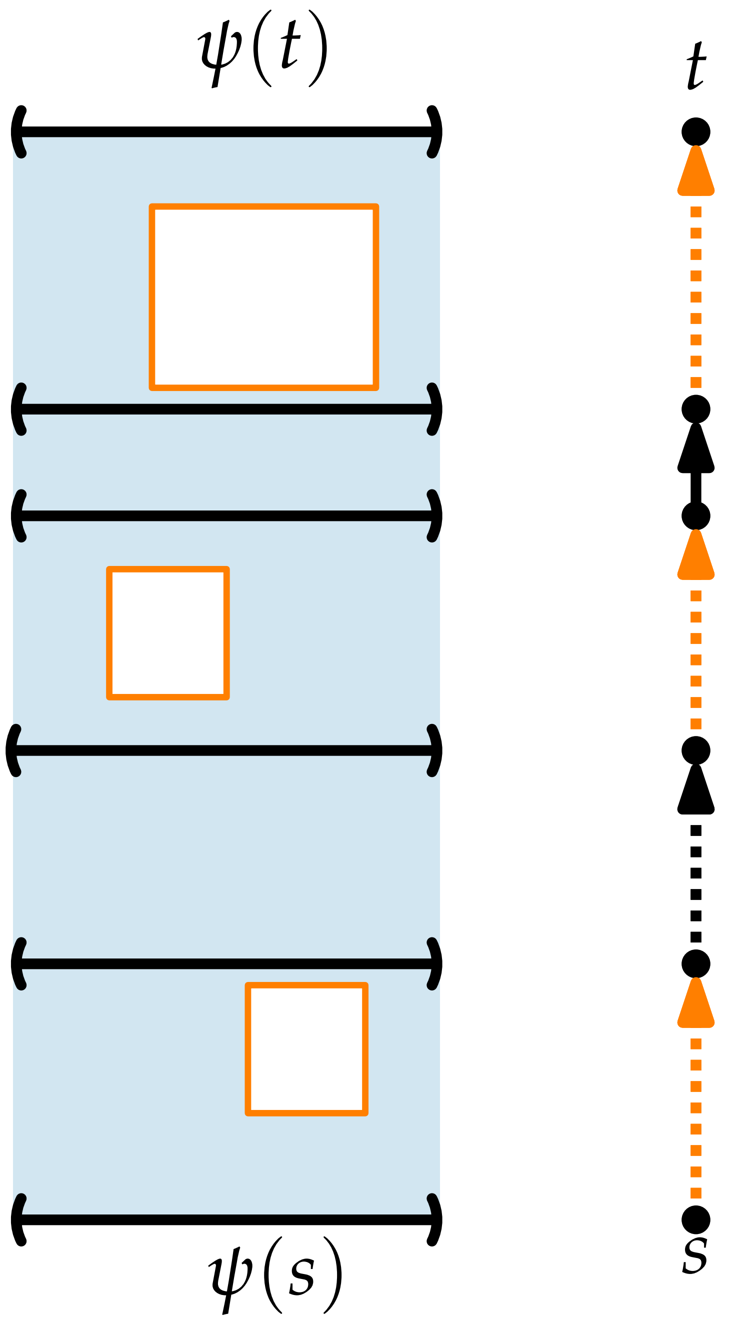
This fixed vertex means we can only make a Fixed-Fixed representation!

S nodes

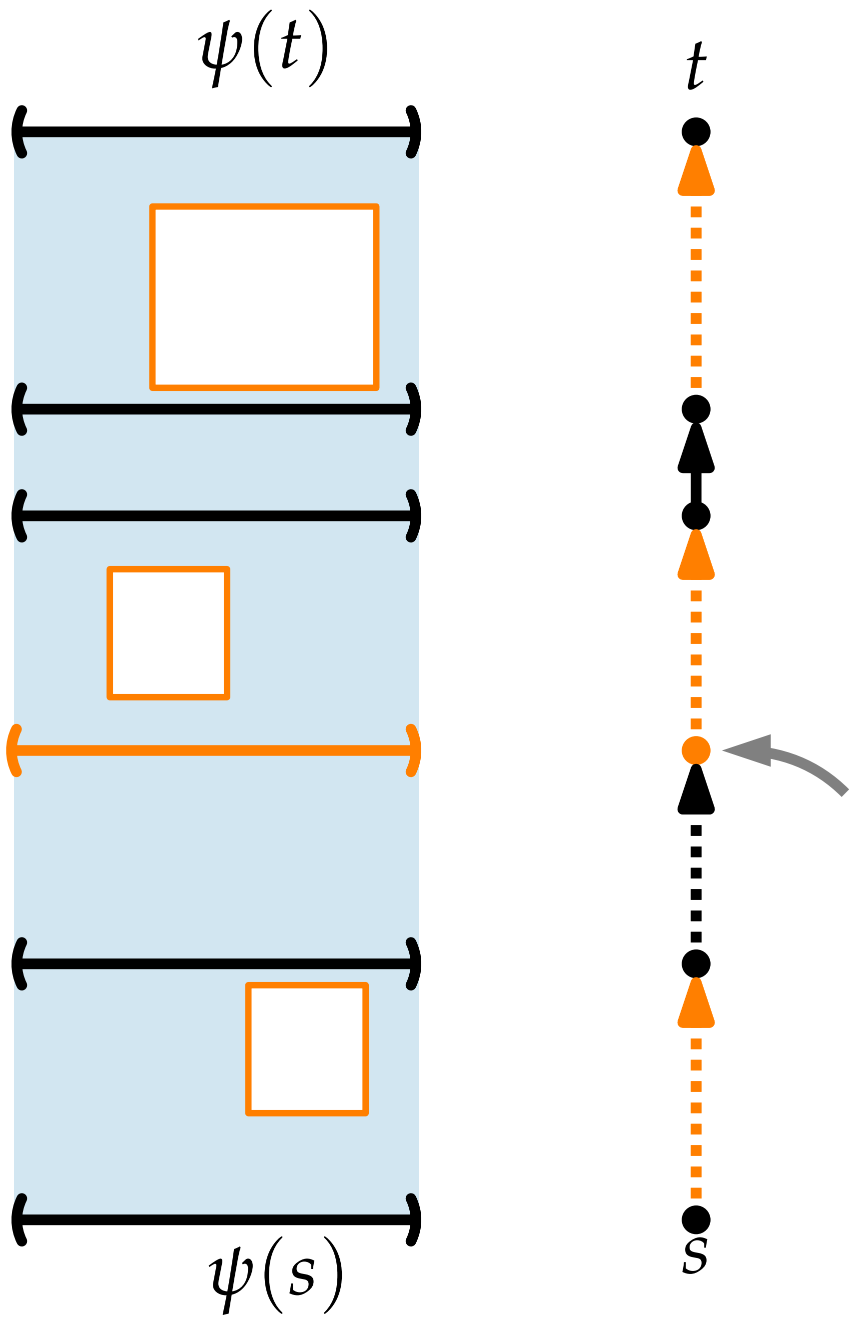


Here we have a chance to make all (LL, FL, LF, FF) types.

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



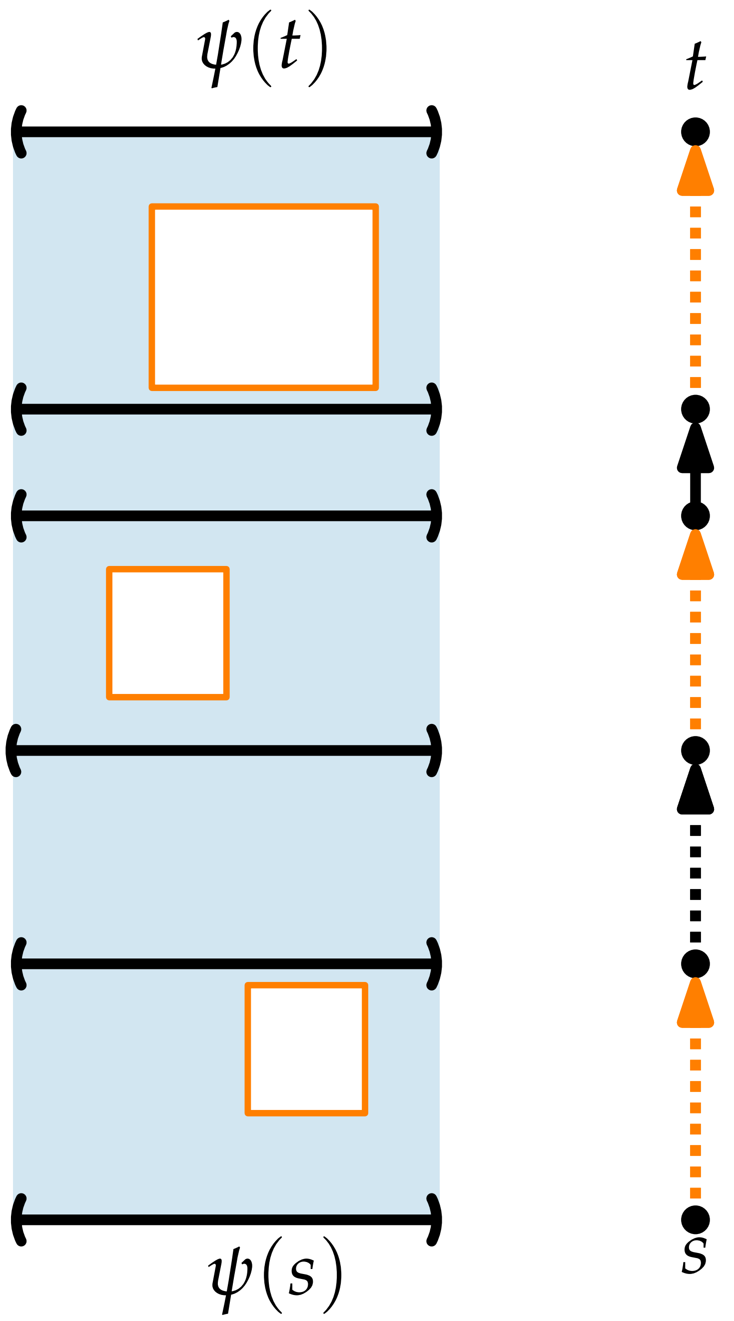
S nodes



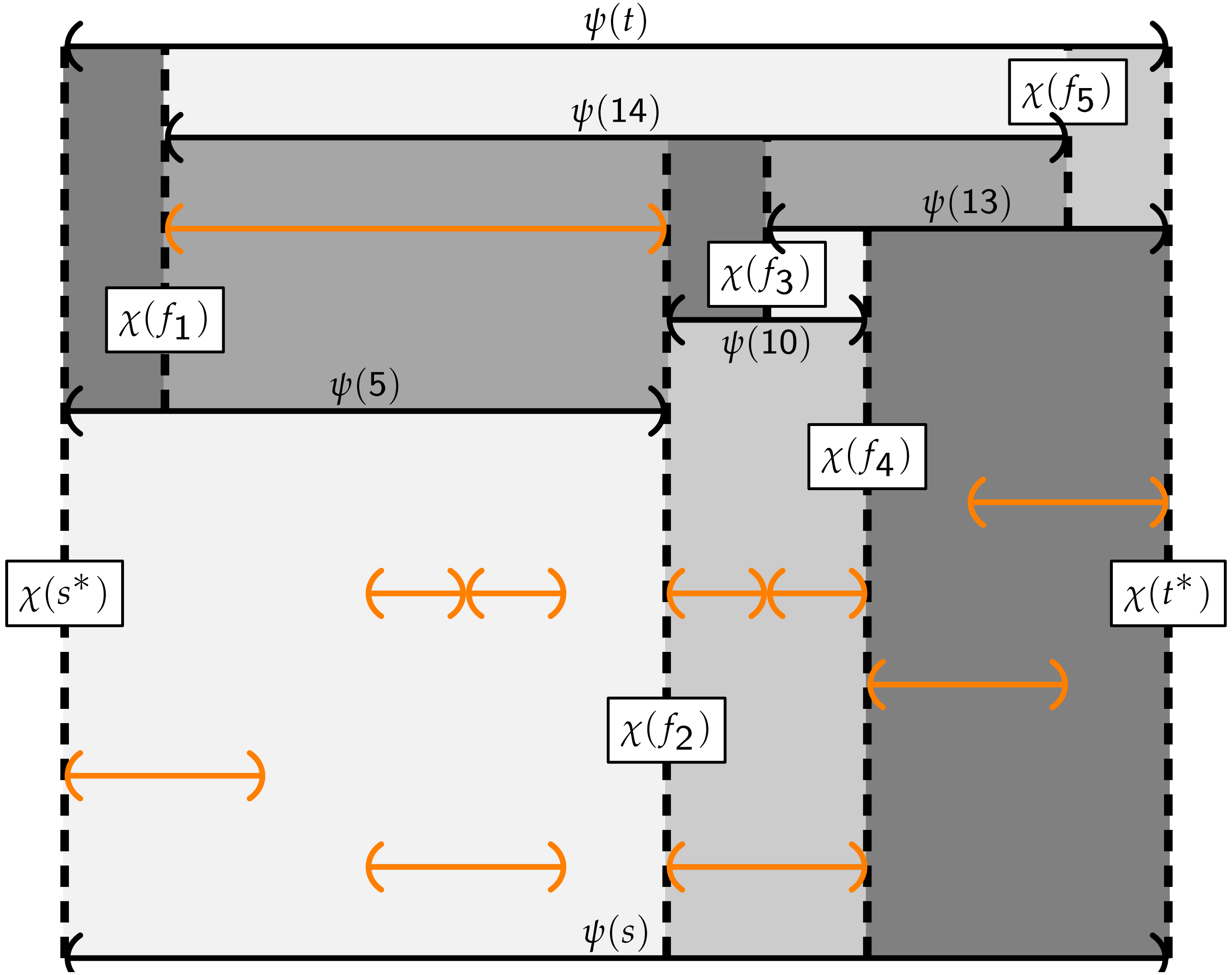
Here we have a chance to make all (LL, FL, LF, FF) types.

How does this work?

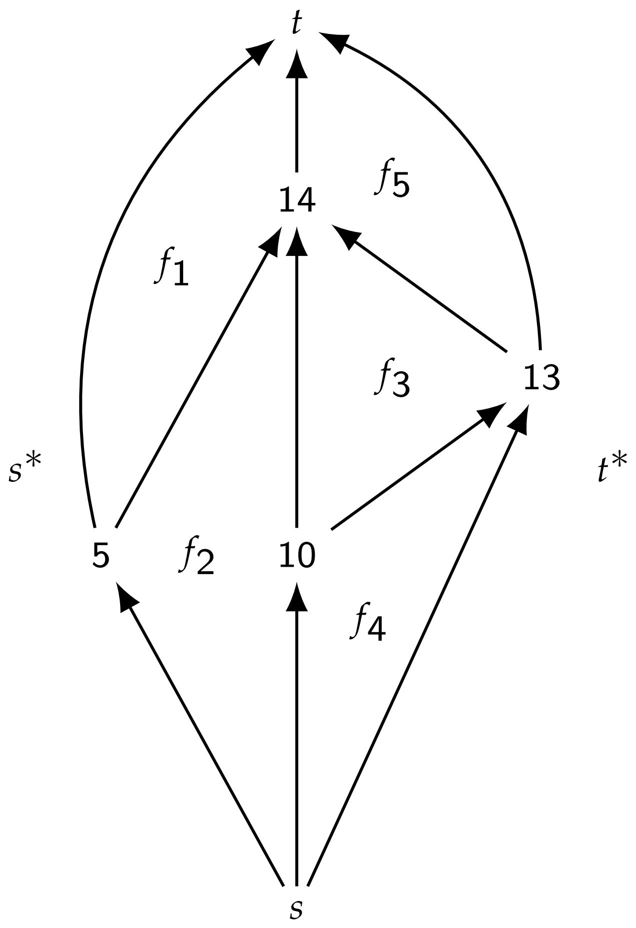
This fixed vertex means we can only make a Fixed-Fixed representation!



R nodes

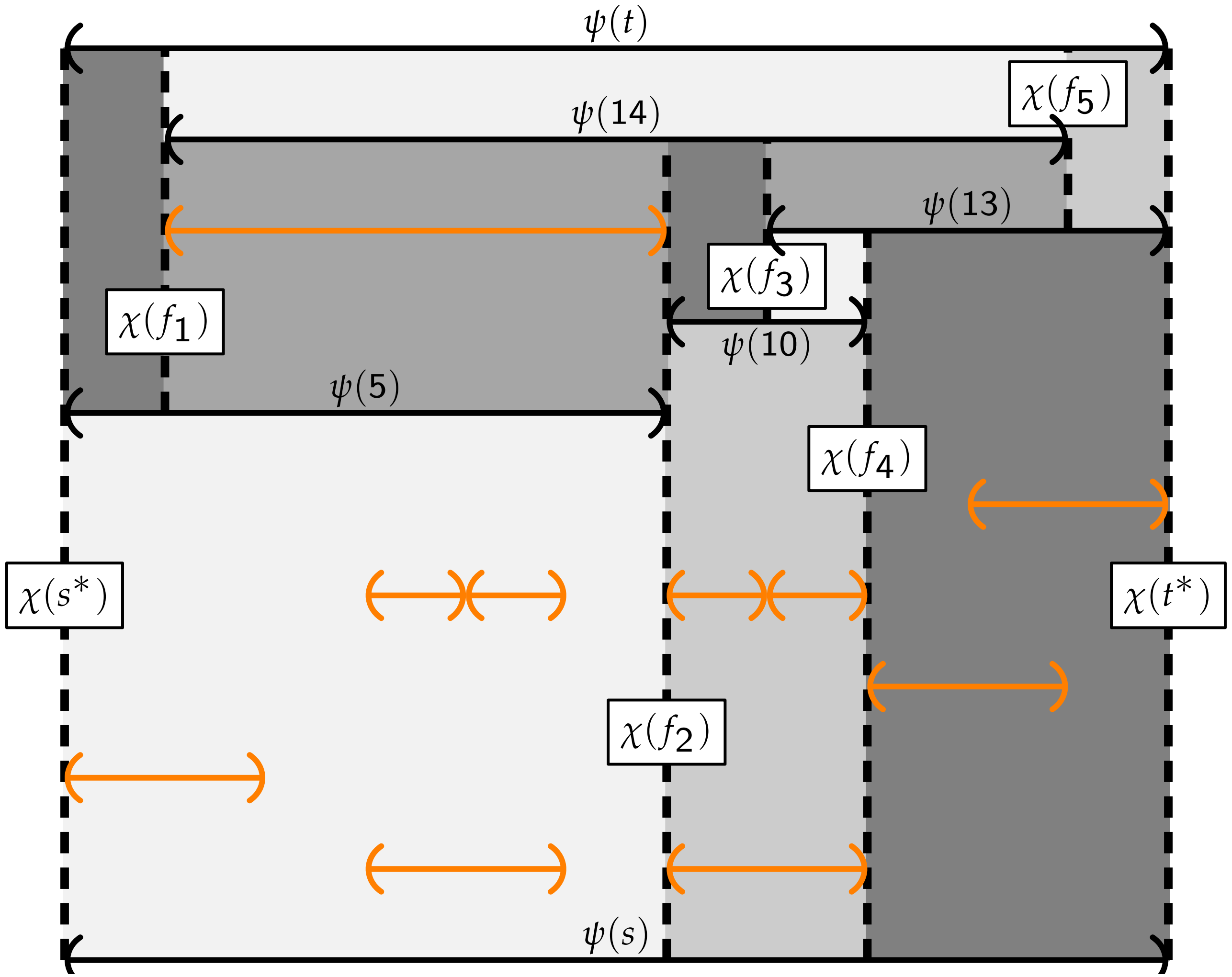


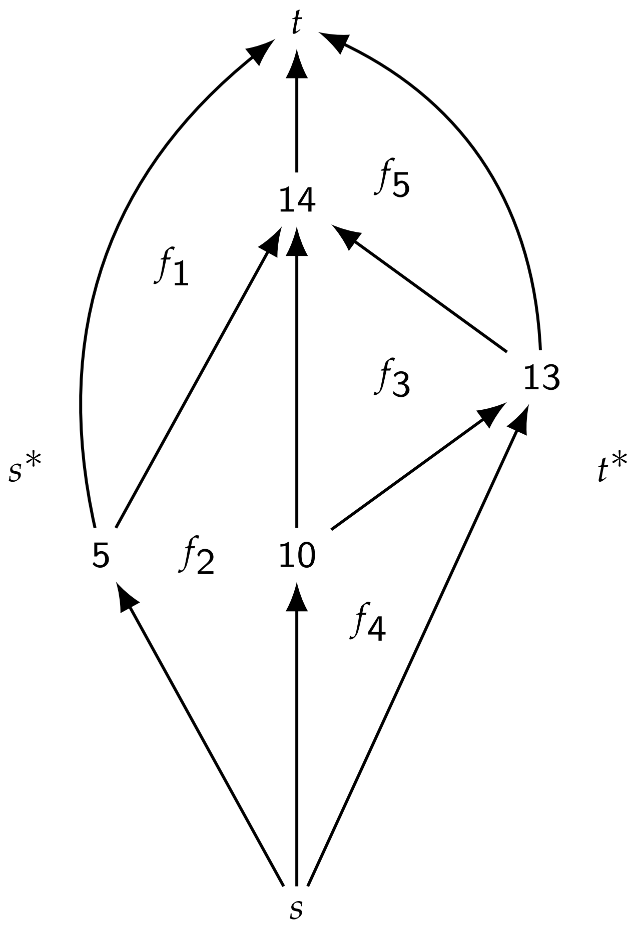
$\psi(t)$
$\chi(f_5)$
$\psi(14)$
$\psi(13)$
$\chi(f_3)$
$\chi(f_1)$
$\psi(10)$
$\psi(5)$
$\chi(f_4)$
$\chi(s^*)$
$\chi(t^*)$
$\chi(f_2)$
$\psi(s)$



t
14
f_5
f_1
f_3
13
s^*
t^*
5
f_2
10
f_4
s

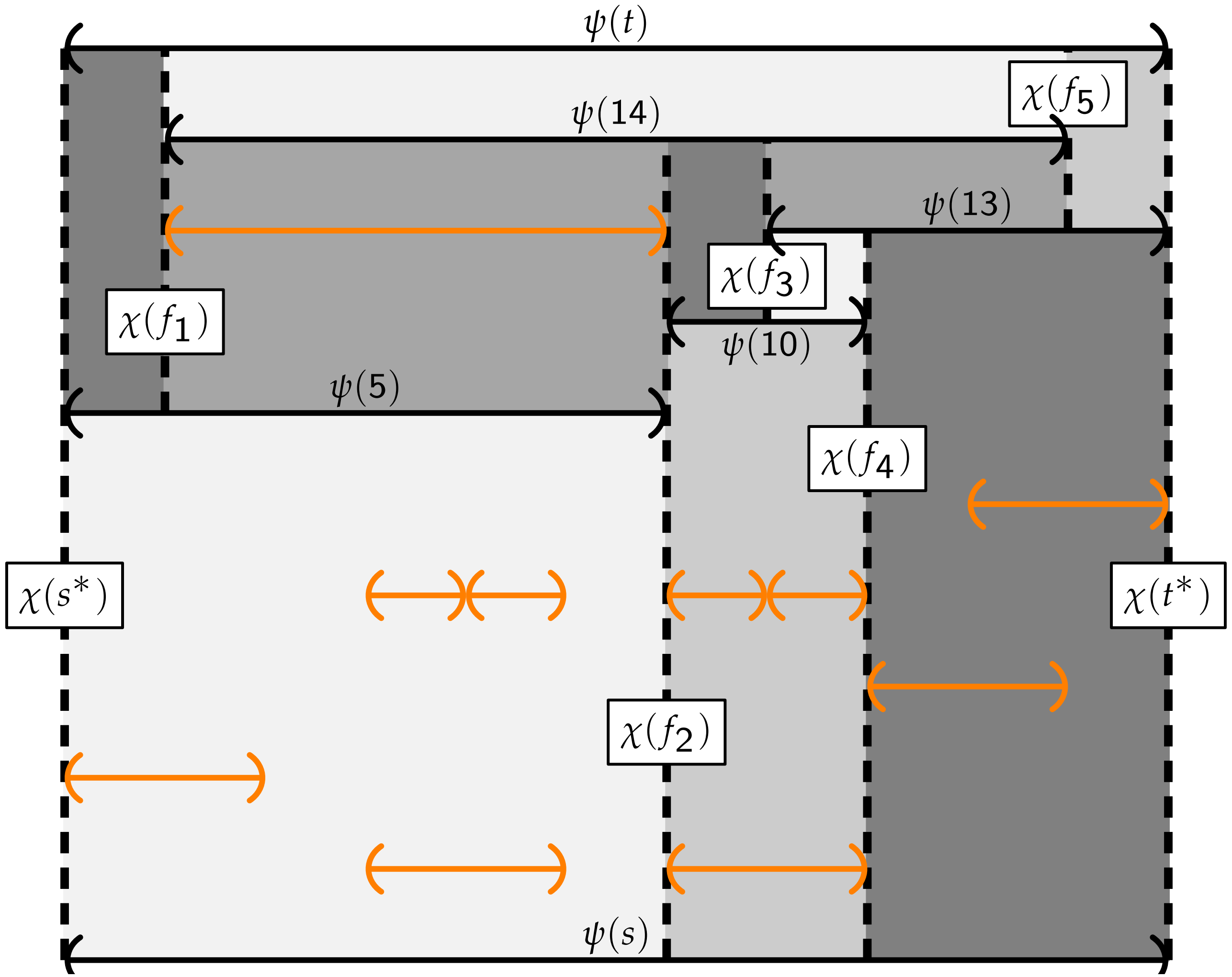
R nodes with 2-SAT formulation

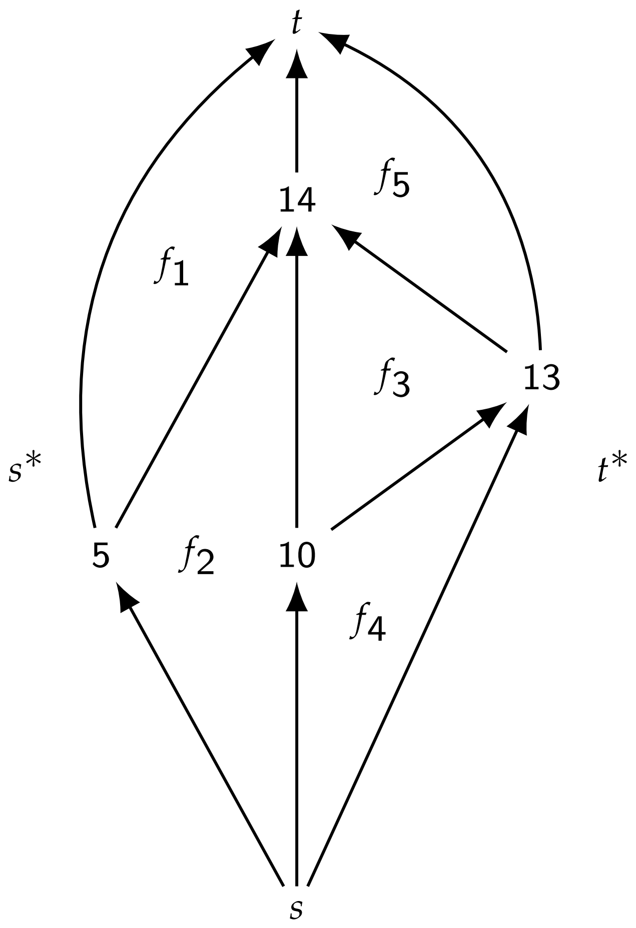




R nodes with 2-SAT formulation

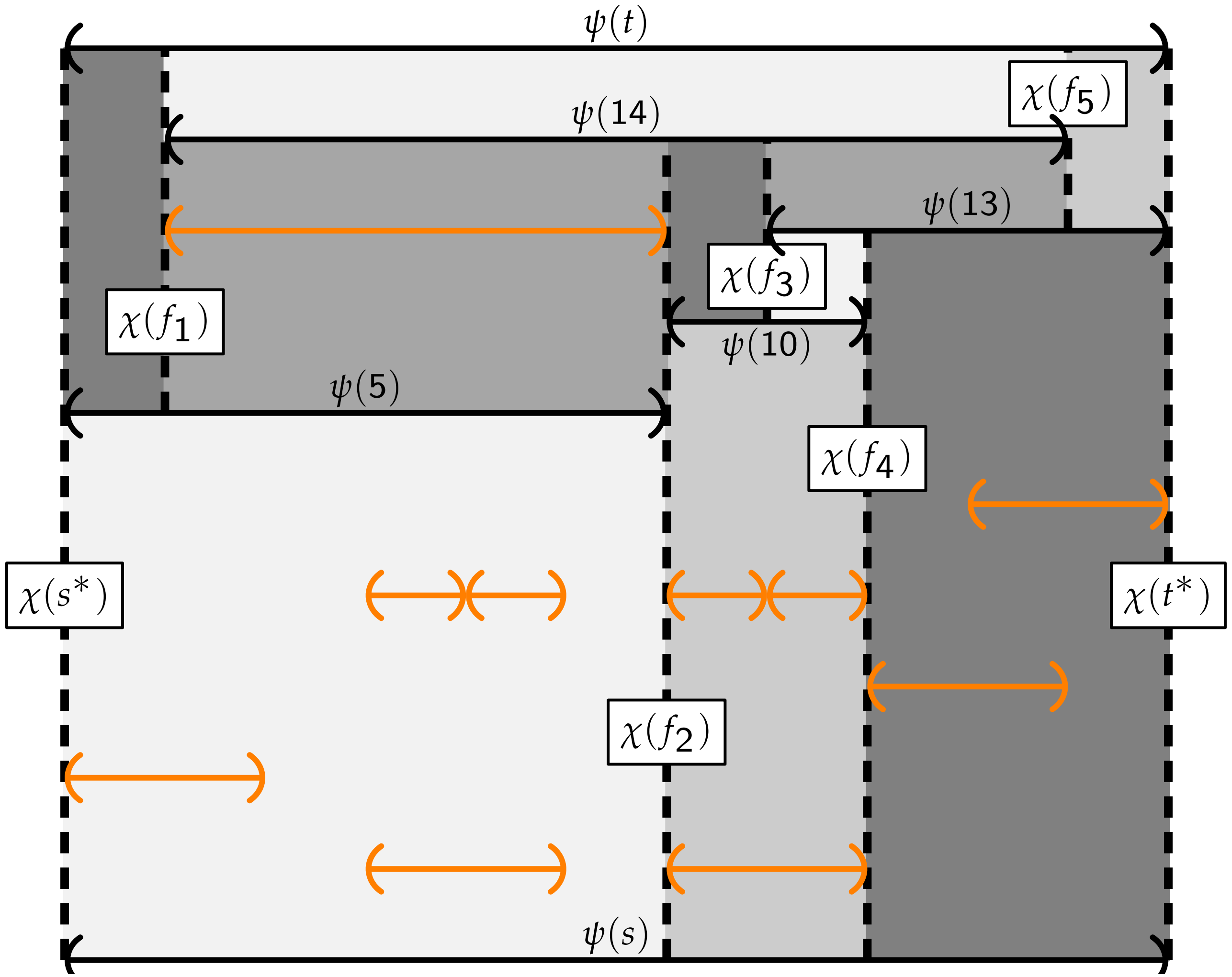
- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of {FF,FL,LF,LL}

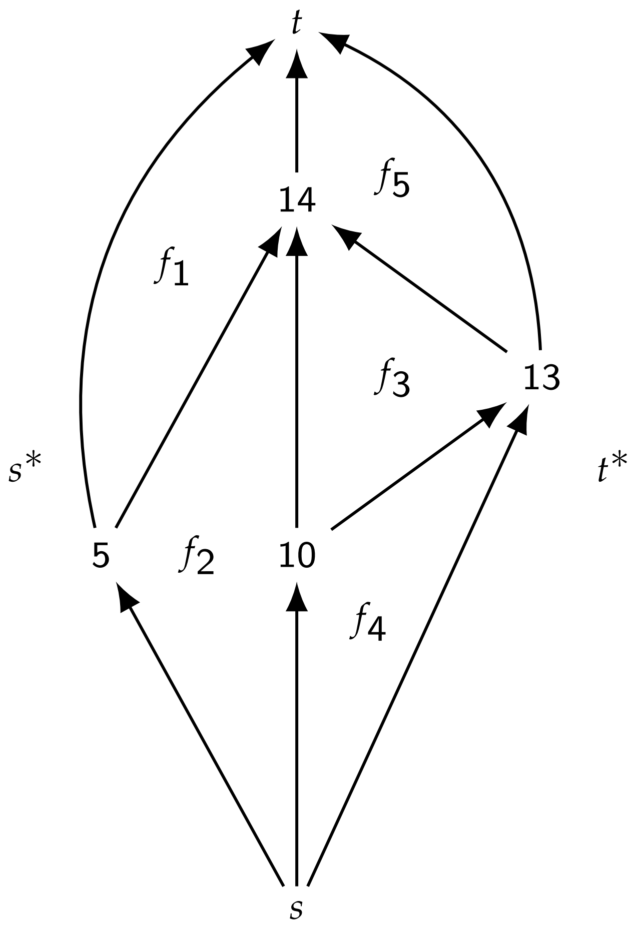




R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of {FF,FL,LF,LL}
- for each face
  - 2 variables encoding position of the splitting line
  - consistency clauses



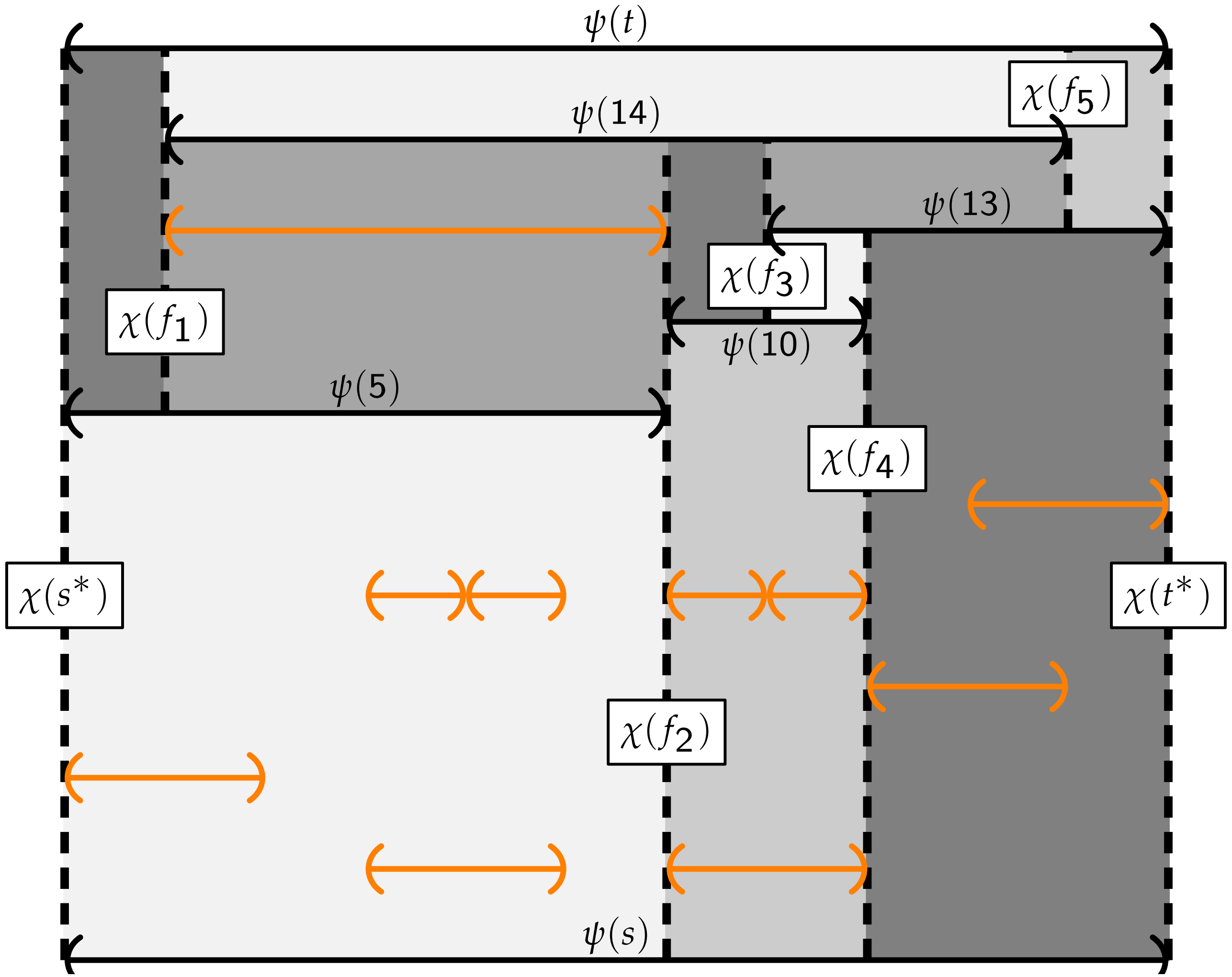


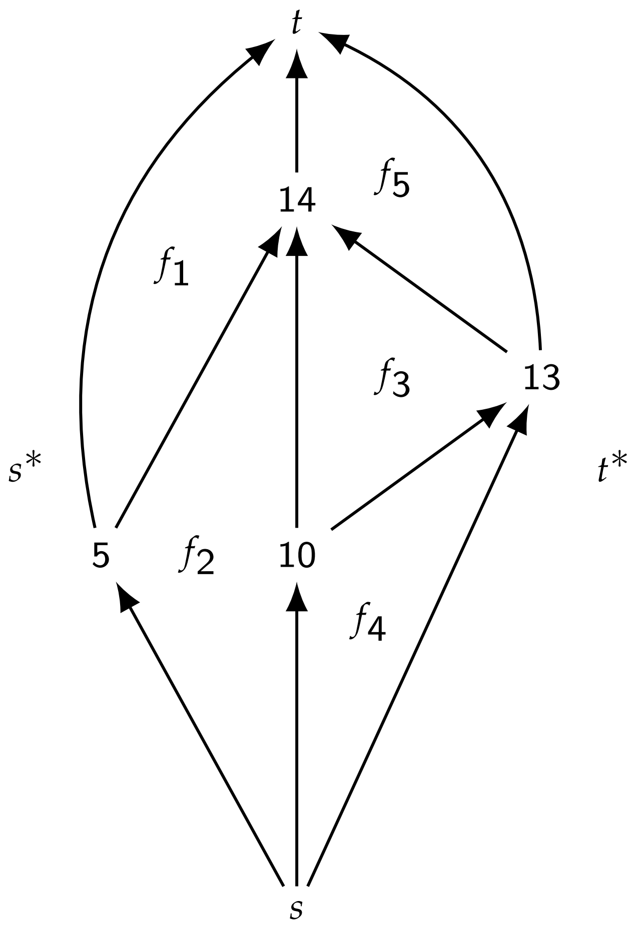
R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of {FF,FL,LF,LL}

- for each face
  - 2 variables encoding position of the splitting line
  - consistency clauses

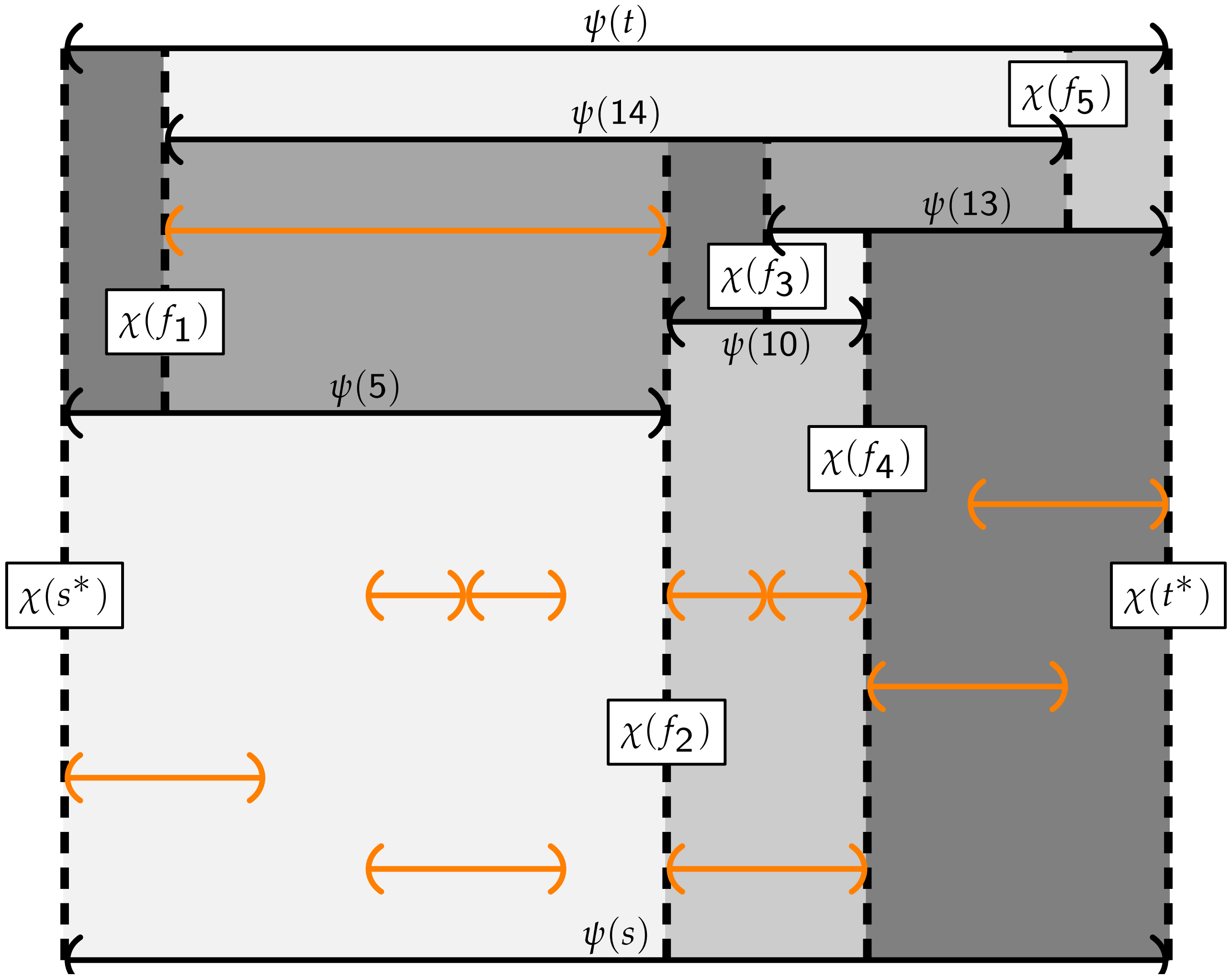
- ordering clauses
  - quadratically many

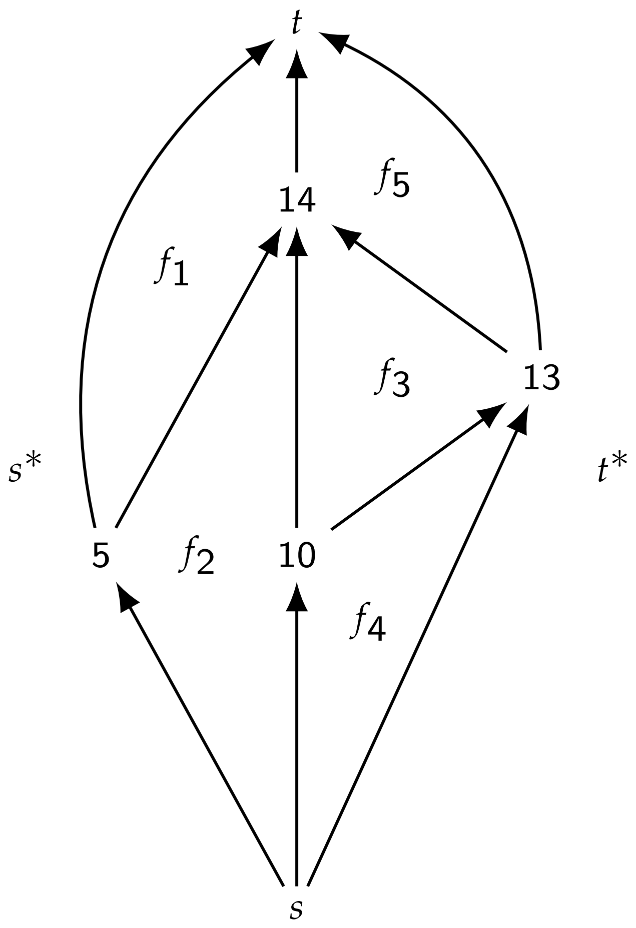




R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of {FF,FL,LF,LL}
- for each face
  - 2 variables encoding position of the splitting line
  - consistency clauses
  - ordering clauses
    - quadratically many
    - tricky part: use only $O(n \log^2 n)$ clauses





NP-hardness of RepExt in general case

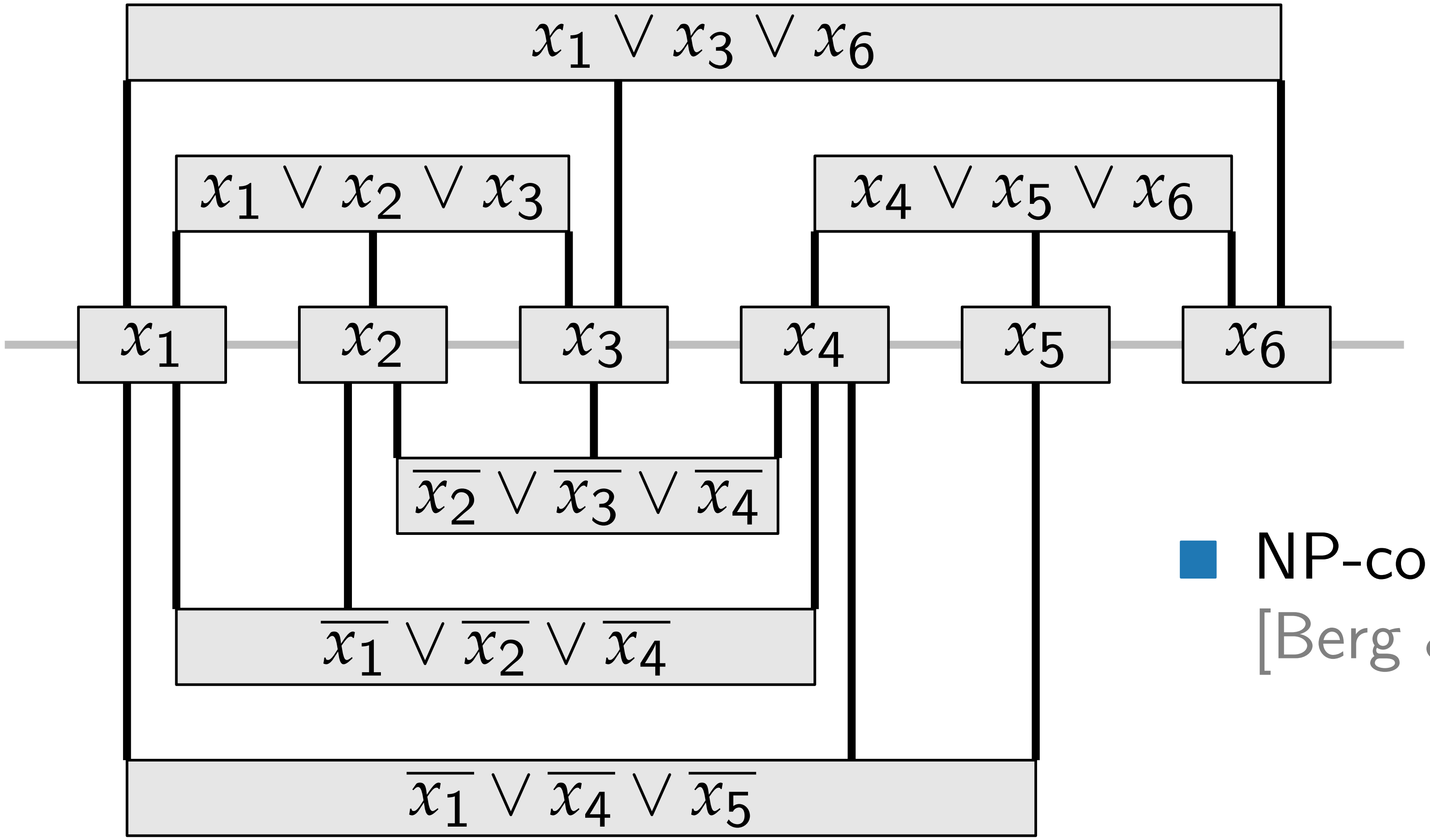
Theorem 2.
ε-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT

NP-hardness of RepExt in general case

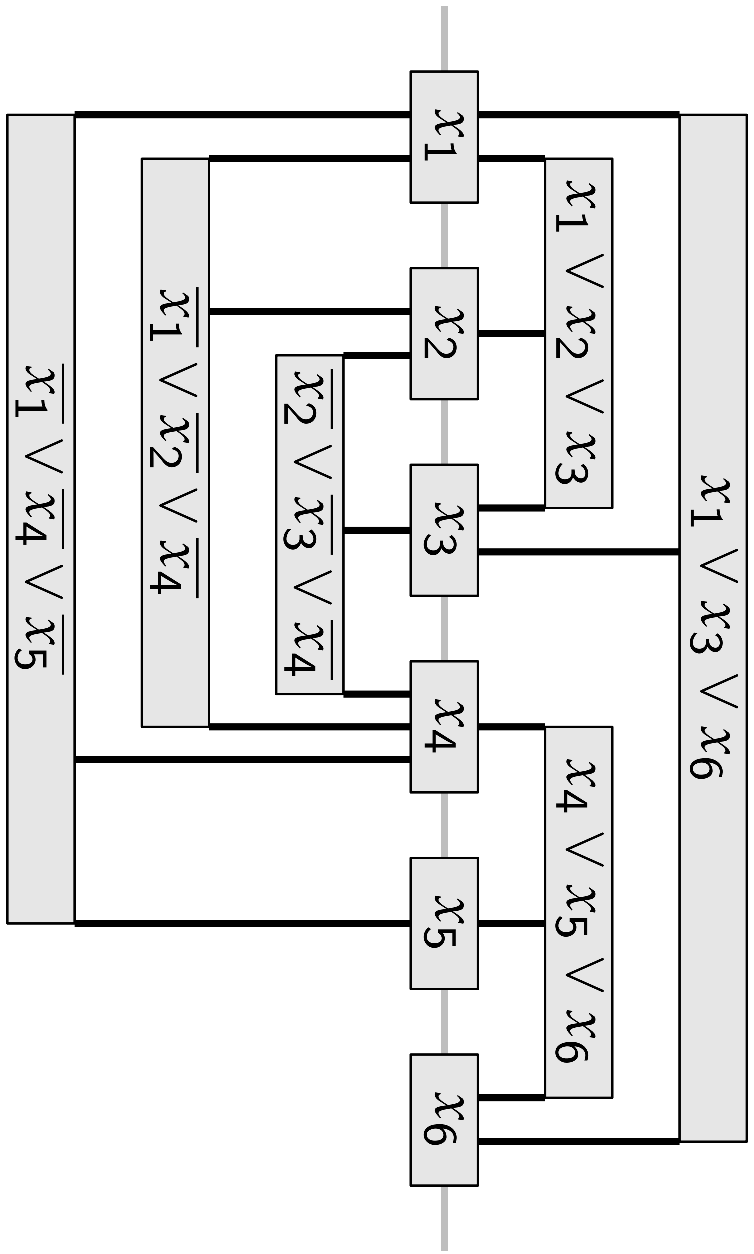
Theorem 2.
ε-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT



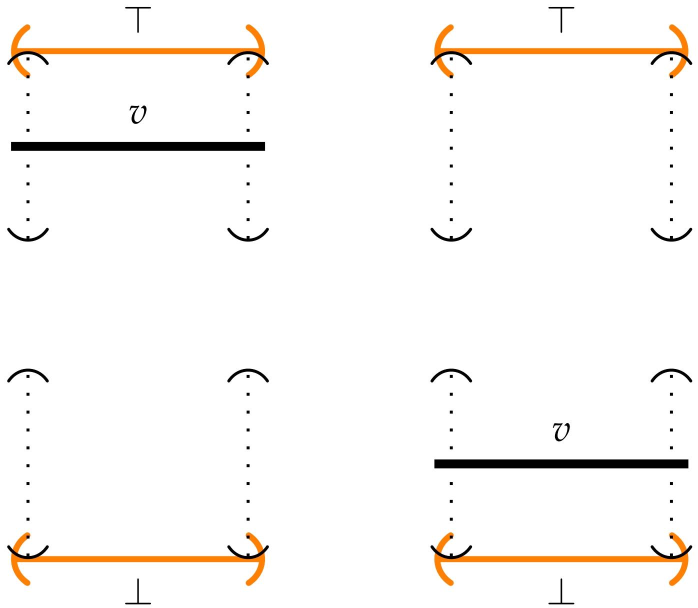
- NP-complete
  [Berg & Khosravi '10]

NP-hardness of RepExt in general case

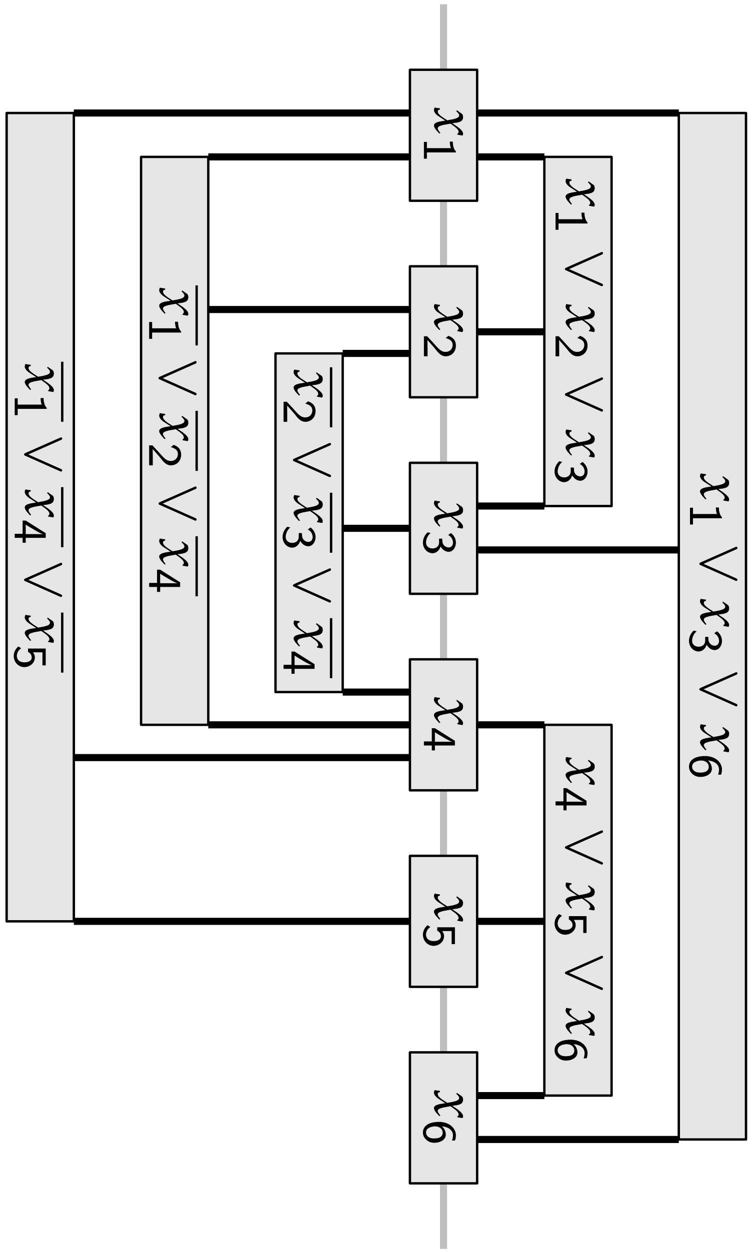


Wire Transmission

transmitting
true and false

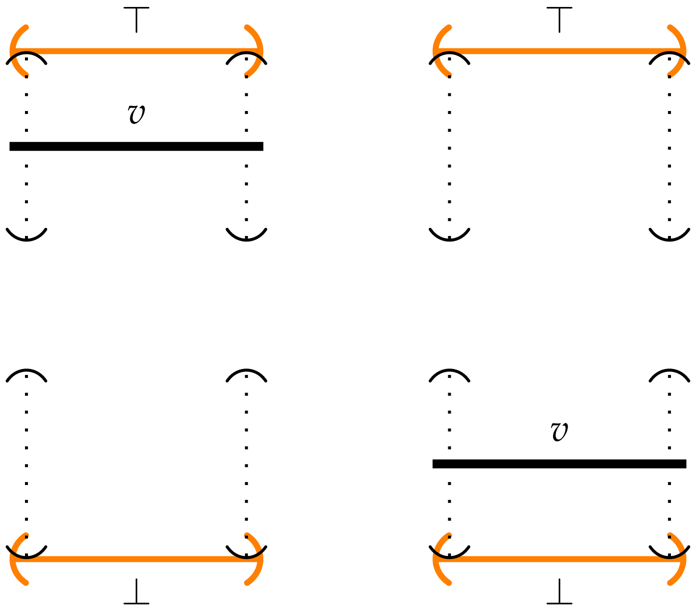


NP-hardness of RepExt in general case



Wire Transmission

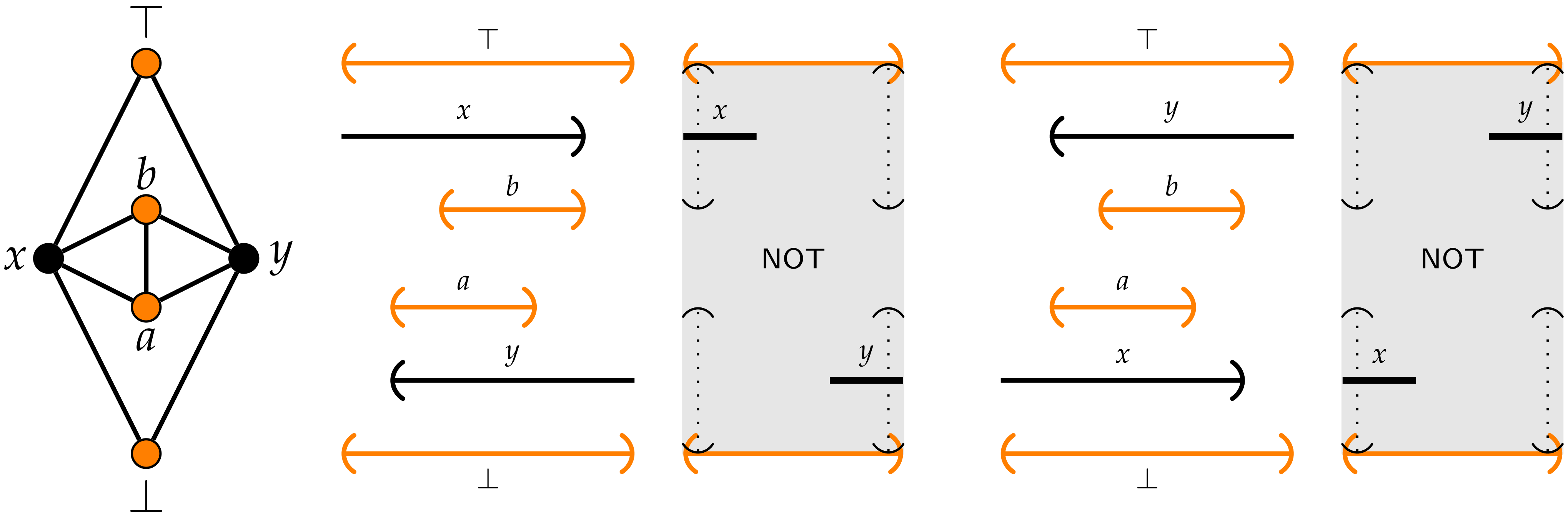
transmitting
true and false



Remark. The following details omit the copying gadgets used for multiple occurrences of the variables

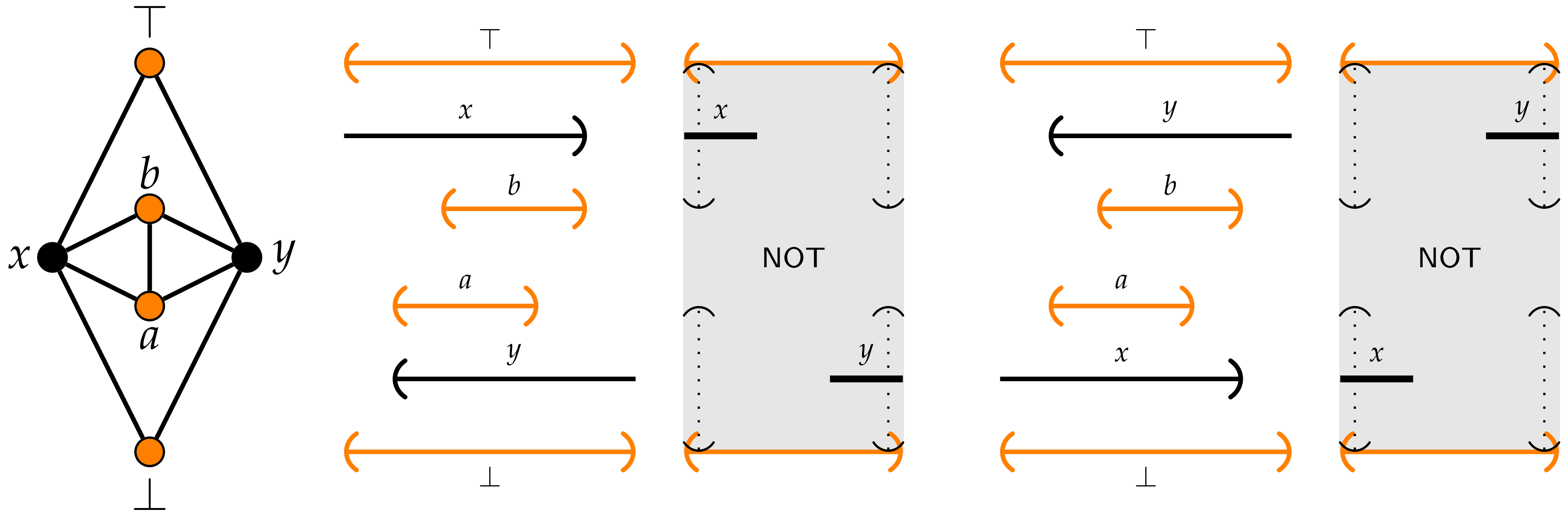
NP-hardness of RepExt in general case

NOT gate



NP-hardness of RepExt in general case

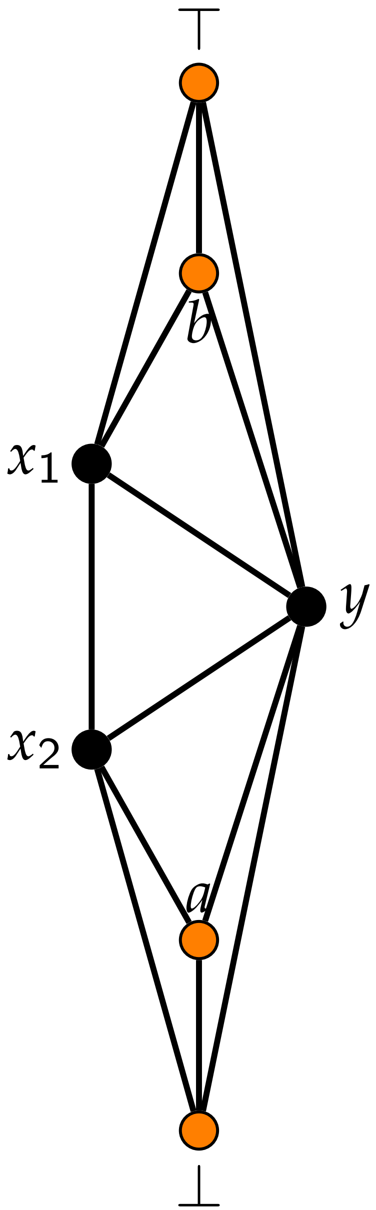
NOT gate

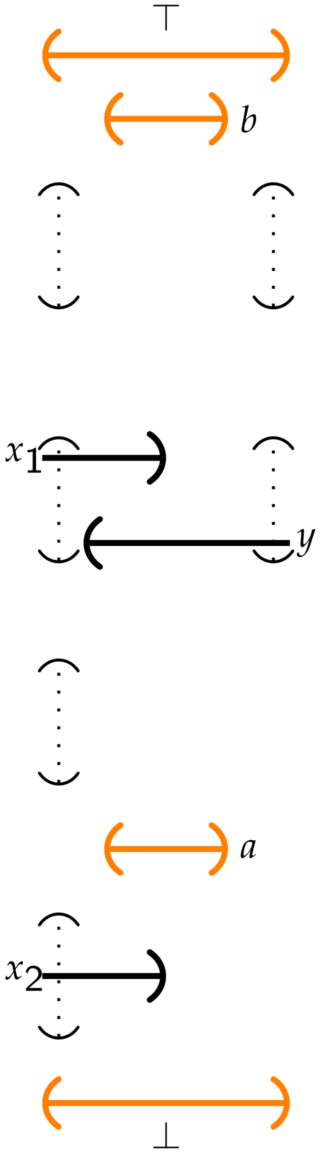


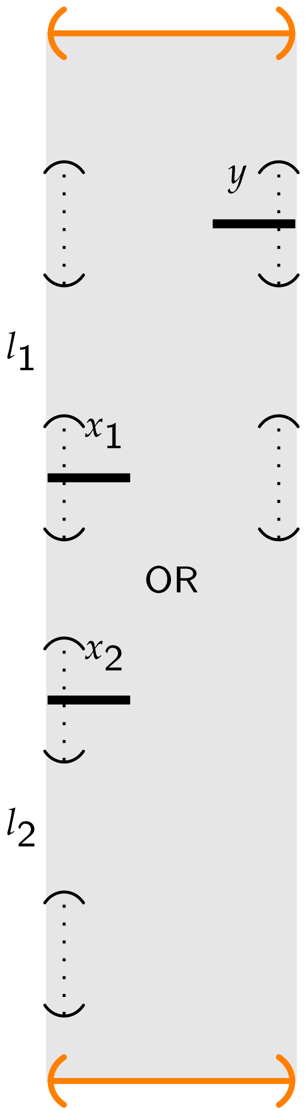
Note: the bars of x and y cannot occur between a and b since a and b are not supposed to be adjacent to either of $\bot$ and $\top$

NP-hardness of RepExt in general case

OR gate

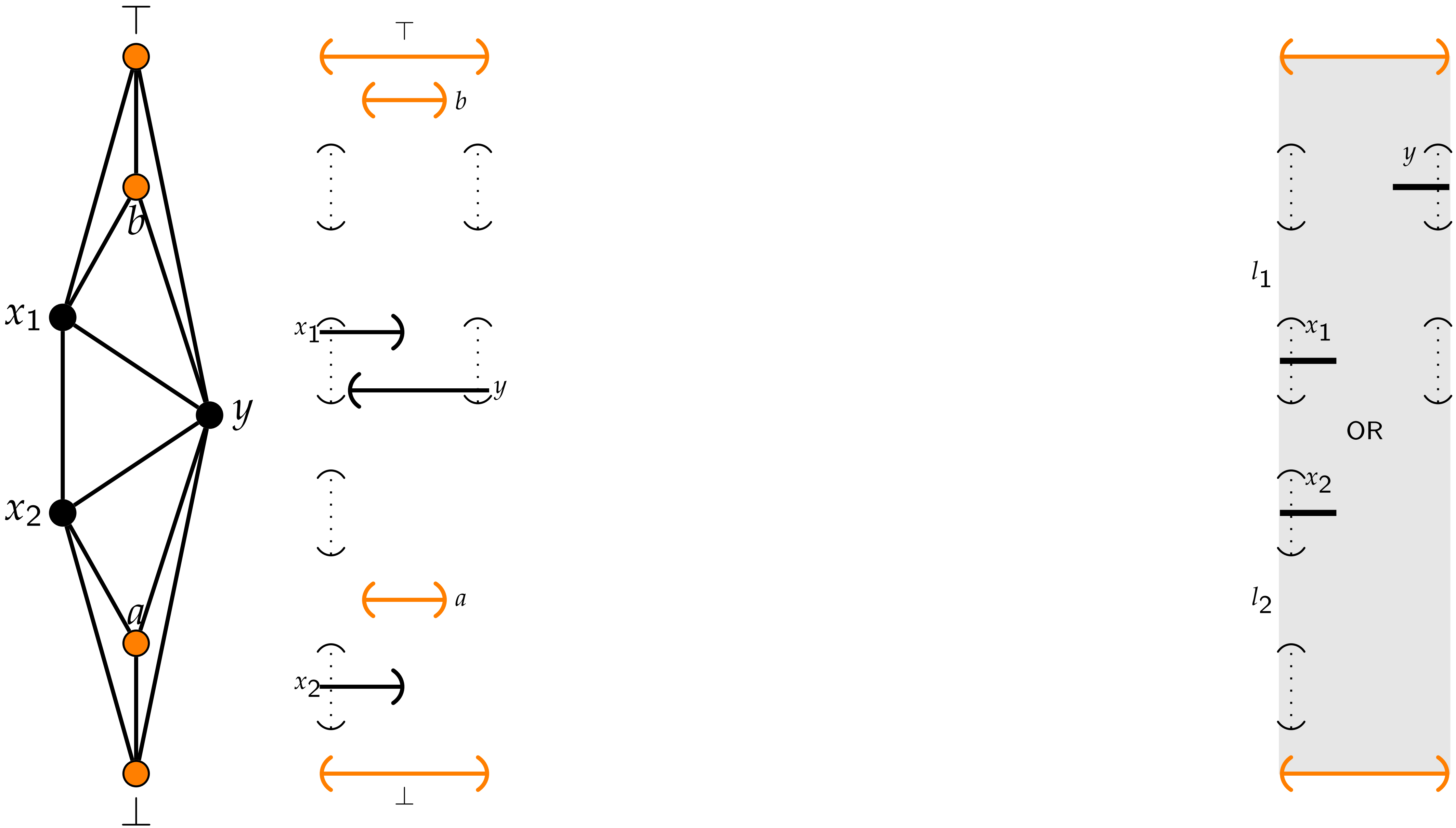






NP-hardness of RepExt in general case

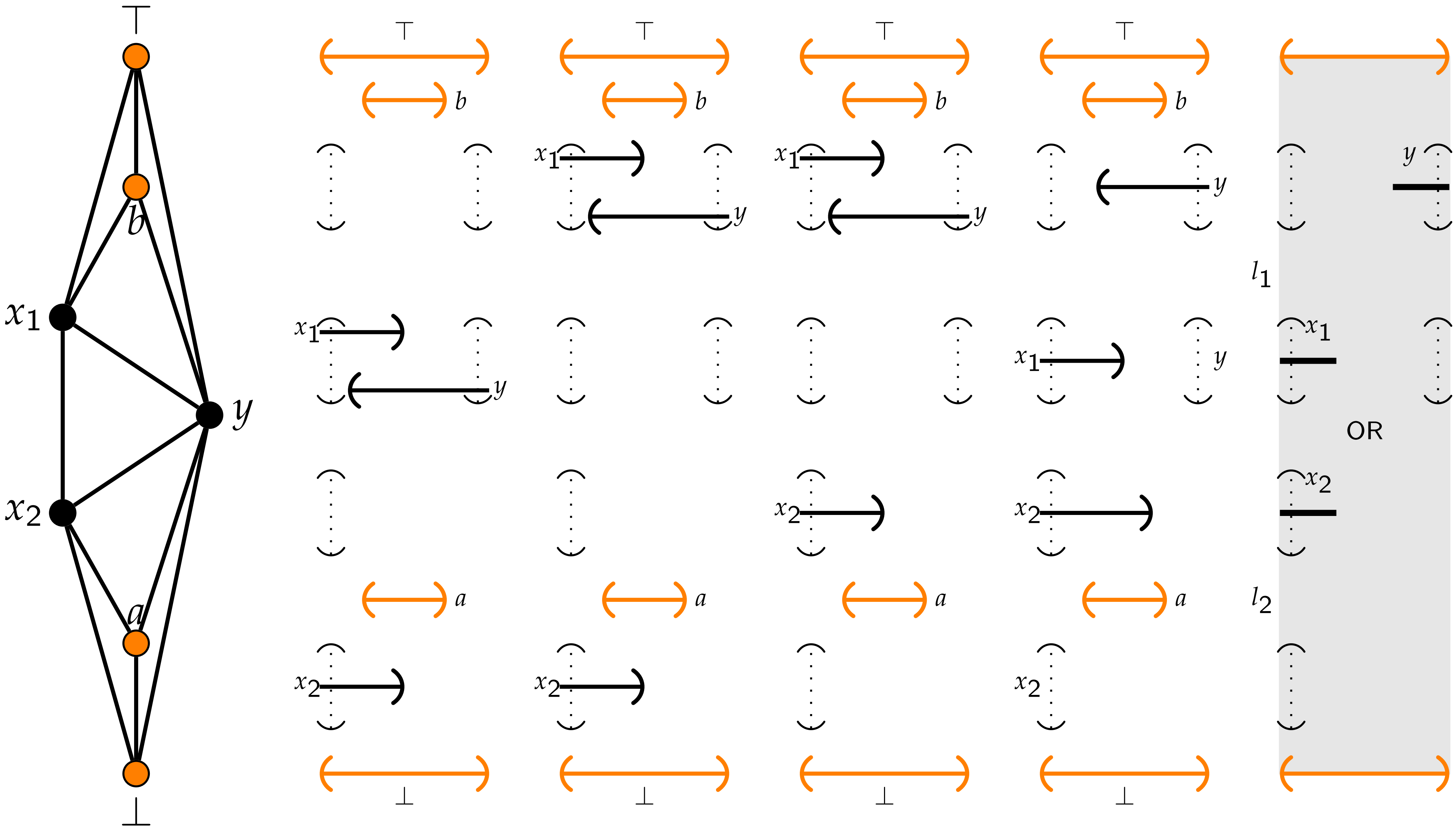
OR gate



sublte point: only need to guarantee that “false” values trasmit

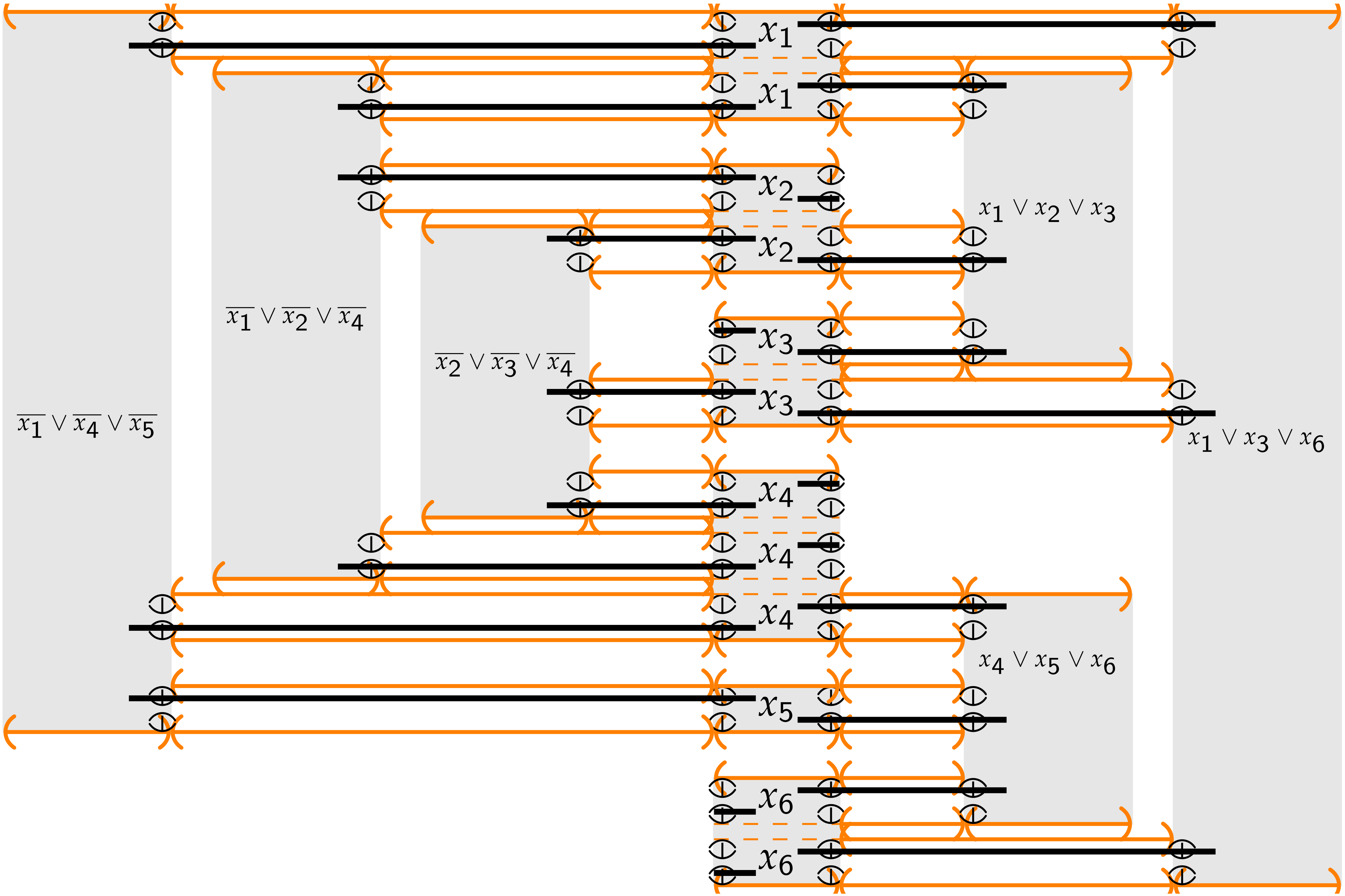
NP-hardness of RepExt in general case

OR gate



sublte point: only need to guarantee that "false" values trasmit

NP-hardness of RepExt in general case



NP-hardness on the Integer Grid (or fixed ε)

Theorem 3.
ε-Bar Visibility Representation Ext. is NP-complete 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 on the Integer Grid (or fixed ε)

Theorem 3.
ε-Bar Visibility Representation Ext. is NP-complete for (series-parallel) st-graphs when restricted to the **integer grid** (or if any fixed $\varepsilon > 0$ is specified).

3-Partition.
Input: A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.
Question: Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into m triples such that the total sum of each triple is exactly w?

- Strongly NP-complete [Garey & Johnson '79]

NP-hardness on the Integer Grid (or fixed ε)

3-Partition.
Input: A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.
Question: Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into m triples such that the total sum of each triple is exactly w?

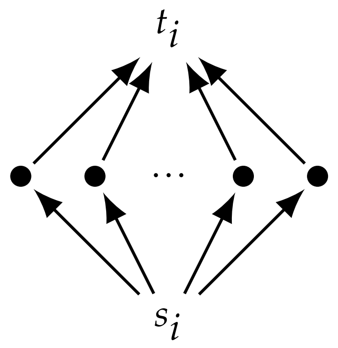
$a_i \rightarrow$

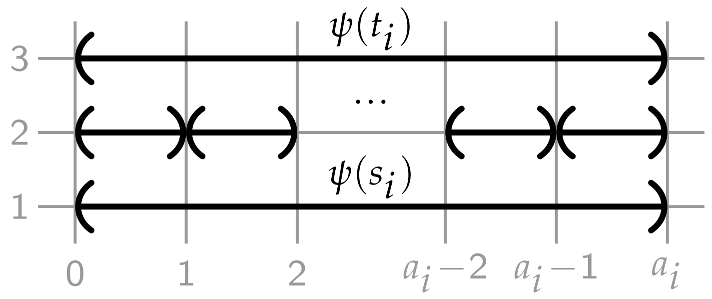
NP-hardness on the Integer Grid (or fixed ε)

3-Partition.

Input: A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.

Question: Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into m triples such that the total sum of each triple is exactly w?

$a_i \rightarrow$  H_i

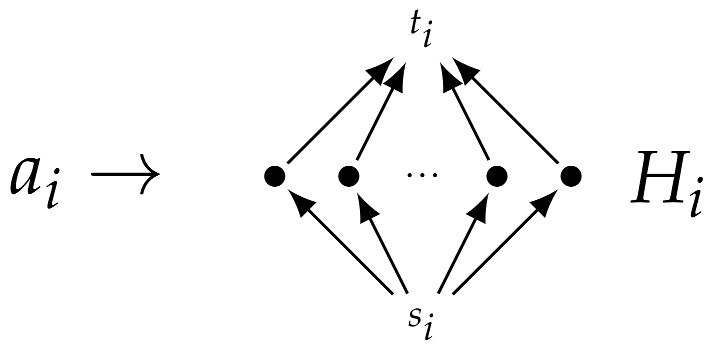


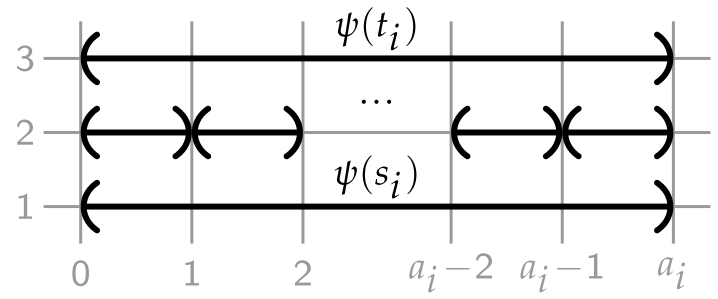
NP-hardness on the Integer Grid (or fixed ε)

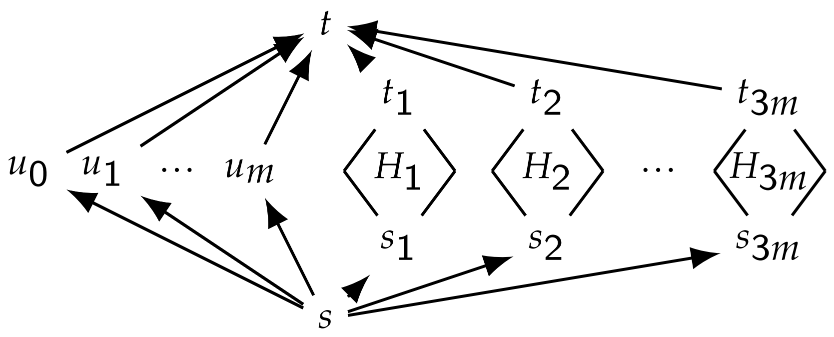
3-Partition.

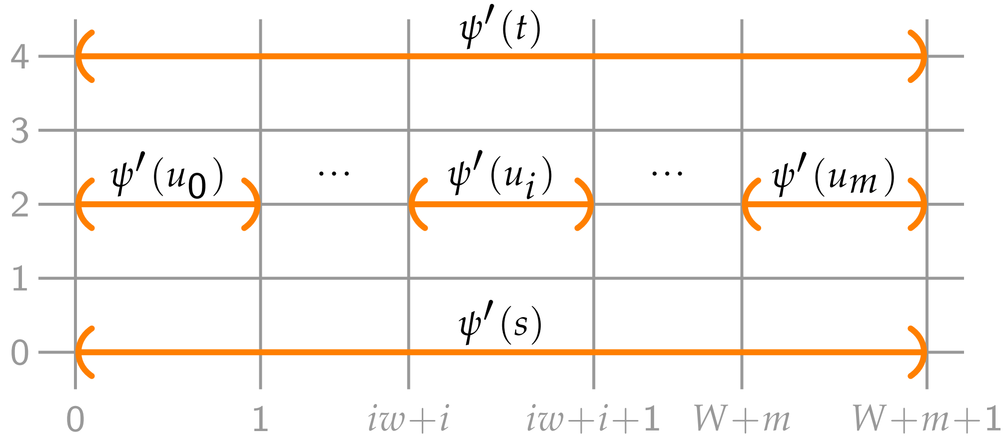
Input: A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.

Question: Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into m triples such that the total sum of each triple is exactly w?









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 can be solved in polynomial time on st-graphs? DAGs?
- Can **Strong** Bar Visibility Recognition / Representation Extension can be solved in polynomial time on st-graphs?

Literature

Main source:

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

Referenced papers:

- [Gutwenger, Mutzel '01] A Linear Time Implementation of SPQR-Trees
- [Wismath '85] Characterizing bar line-of-sight graphs
- [Tamassia, Tollis '86] Algorithms for visibility representations of planar graphs
- [Andreae '92] Some results on visibility graphs
- [Chaplick, Dorbec, Kratchovíl, Montassier, Stacho '14] Contact representations of planar graphs: Extending a partial representation is hard