2-Reachability is Almost as Easy as Transitive Closure

ETH zürich

TI Mittagsseminar, November 8, 2016, daniel.graf@inf.ethz.ch

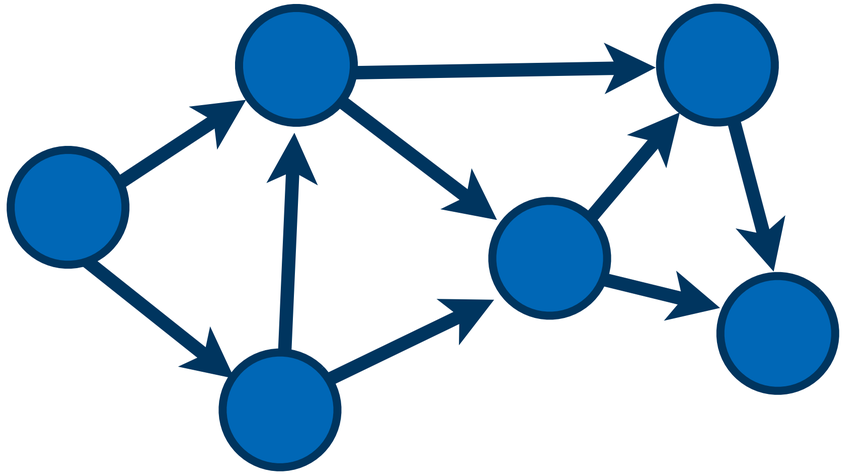
2-Reachability is Almost as Easy as Transitive Closure

ETH zürich

TI Mittagsseminar, November 8, 2016, daniel.graf@inf.ethz.ch

Connectivity in Directed Graphs

ETH zürich

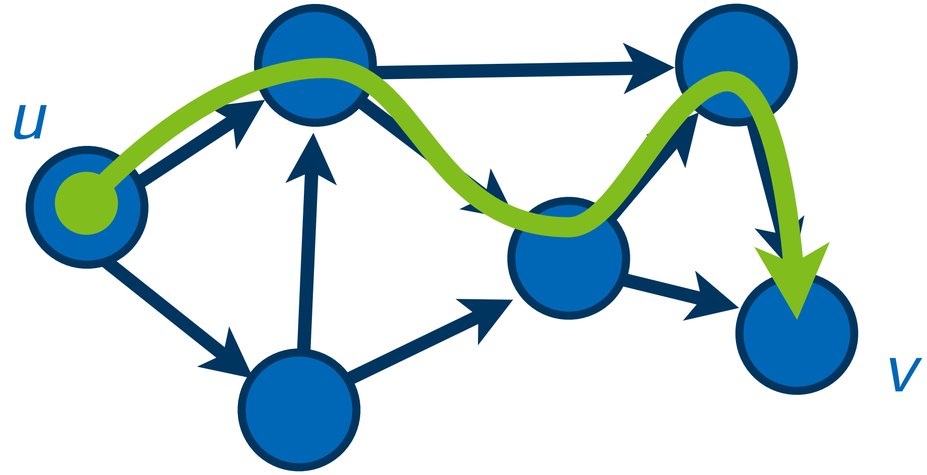


Connectivity in Directed Graphs

ETH zürich

Reachability: can we get from u to v?

Algorithms 101: DFS, toposort, SCC



Connectivity in Directed Graphs

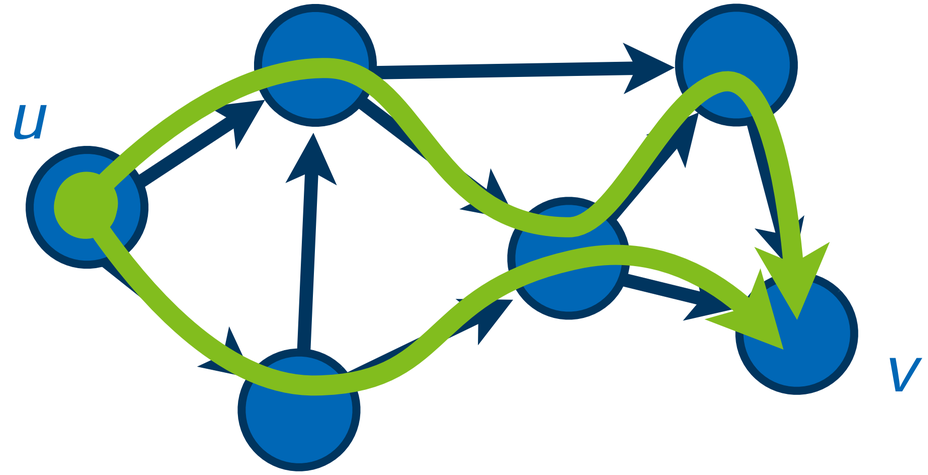
ETH zürich

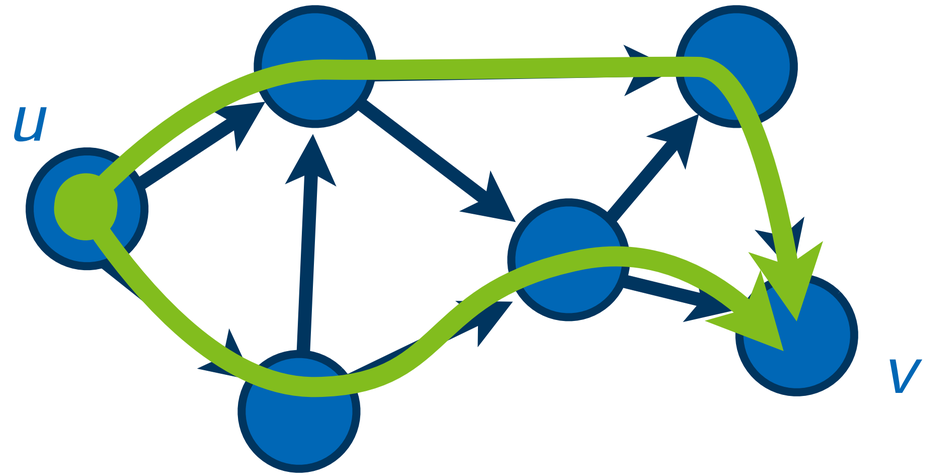
Reachability: can we get from u to v?

Algorithms 101: DFS, toposort, SCC

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?





Connectivity in Directed Graphs

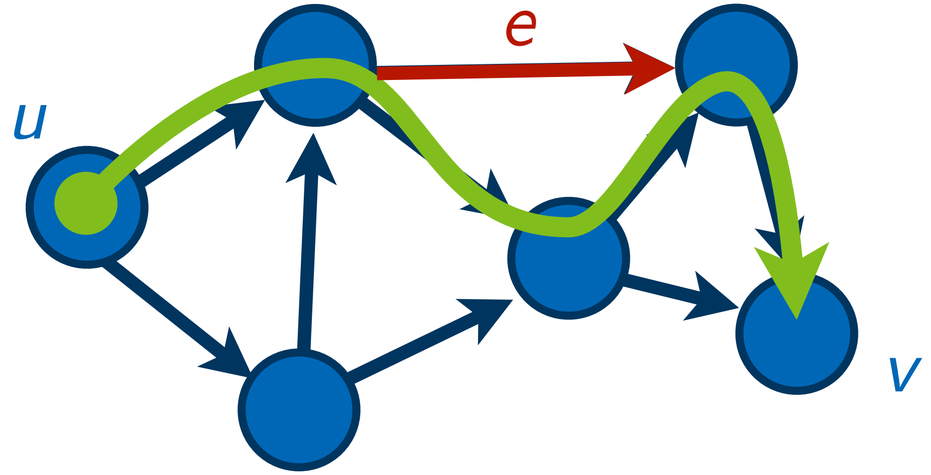
ETH zürich

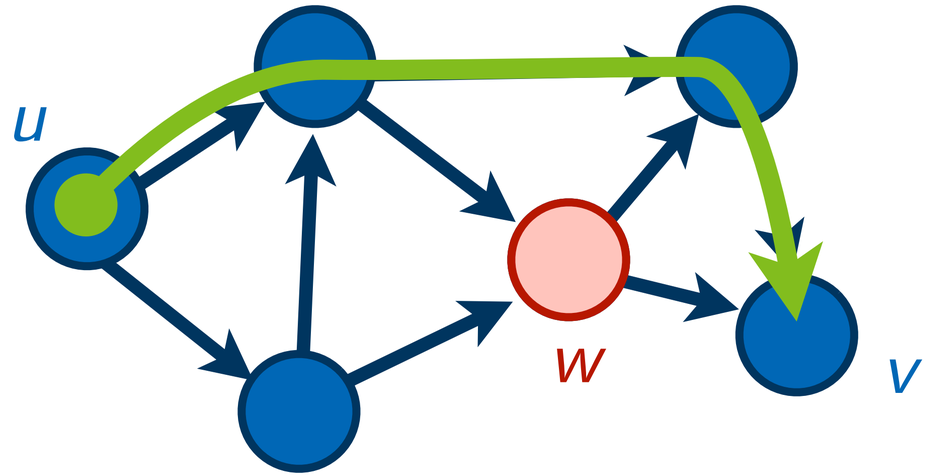
Reachability: can we get from u to v?

Algorithms 101: DFS, toposort, SCC

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?
- u-v-path avoiding e?
- u-v-path avoiding w?





Connectivity in Directed Graphs

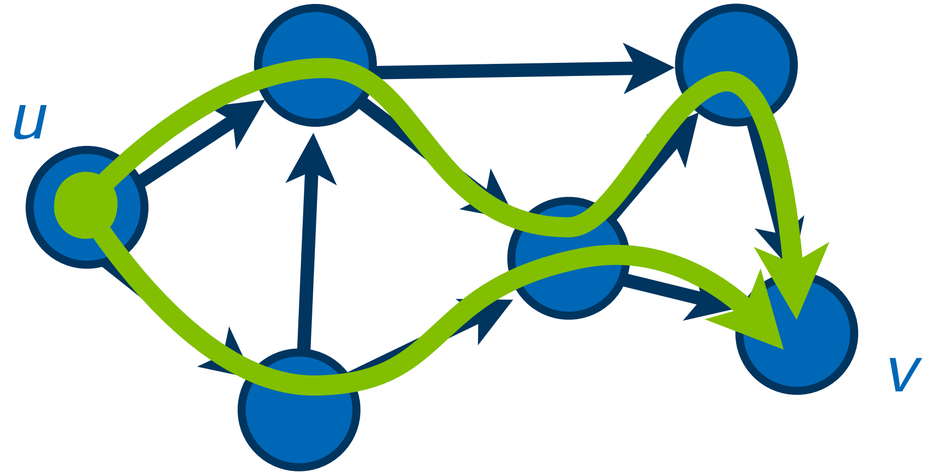
ETH zürich

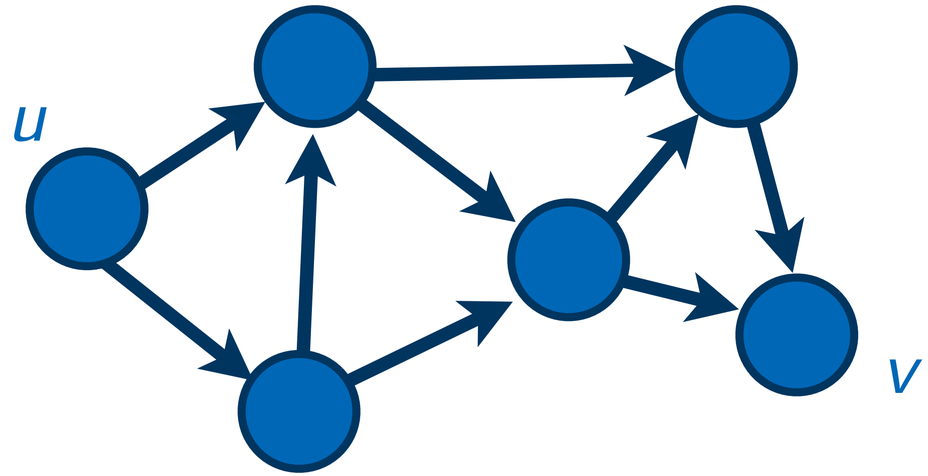
Reachability: can we get from u to v?

Algorithms 101: DFS, toposort, SCC

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?
- u-v-path avoiding e?
- u-v-path avoiding w?



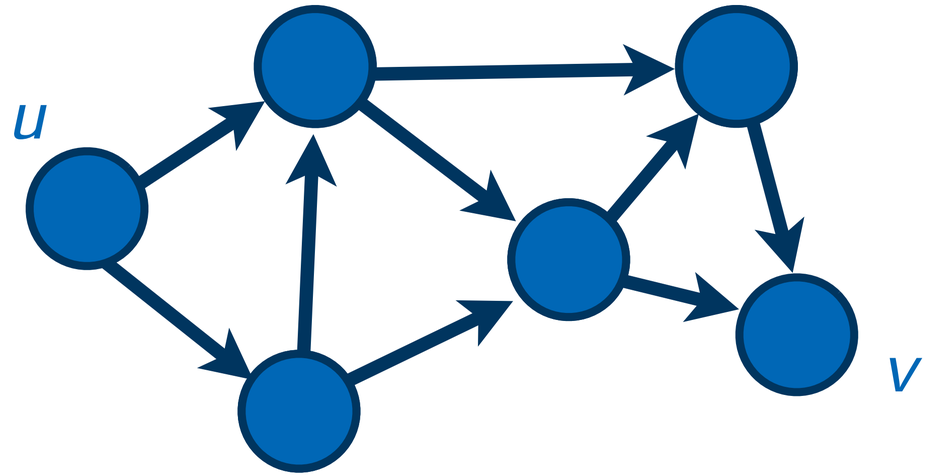


Our Problem: All-Pairs 2-Reachability

ETH zürich

Given: directed graph $G = (V, E)$

with n vertices, m edges



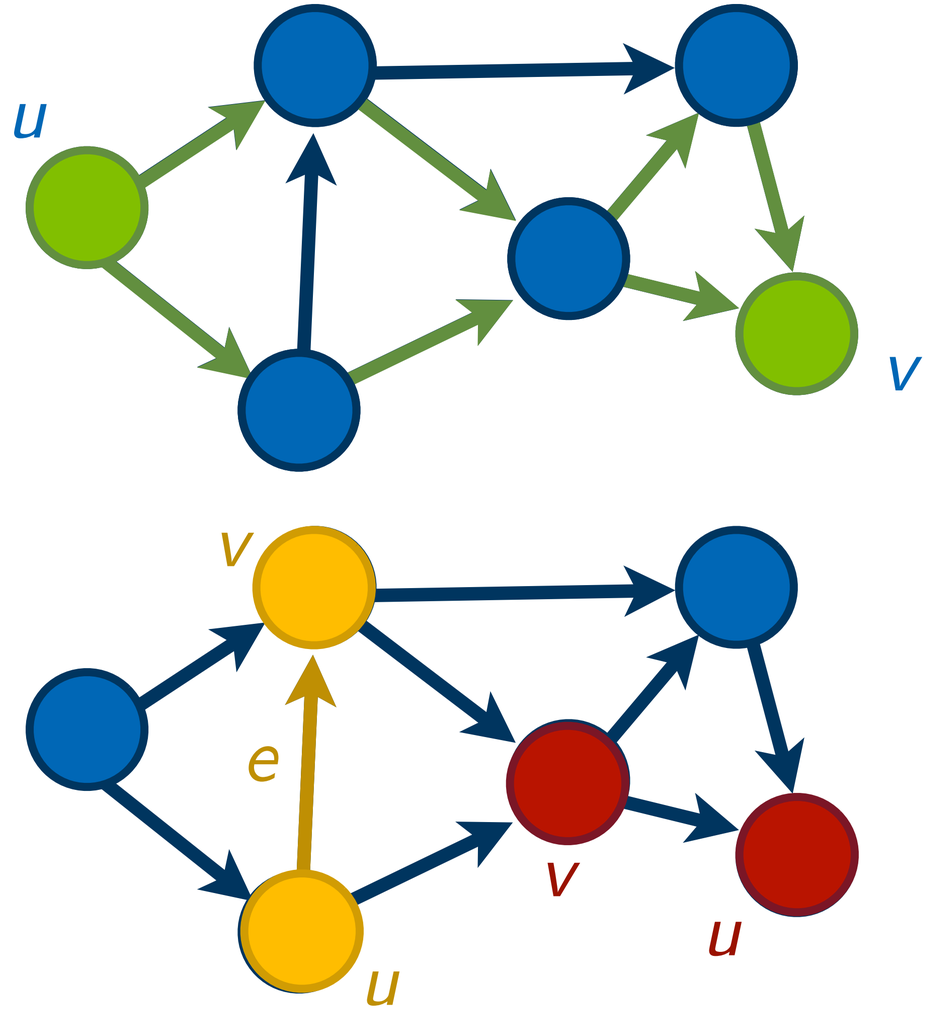
Our Problem: All-Pairs 2-Reachability

ETH zürich

Given: directed graph $G = (V, E)$
with n vertices, m edges

For all $u, v \in V$, decide if

- many edge-disjoint u-v-paths
- some edge e on every u-v-path
- no u-v-path at all



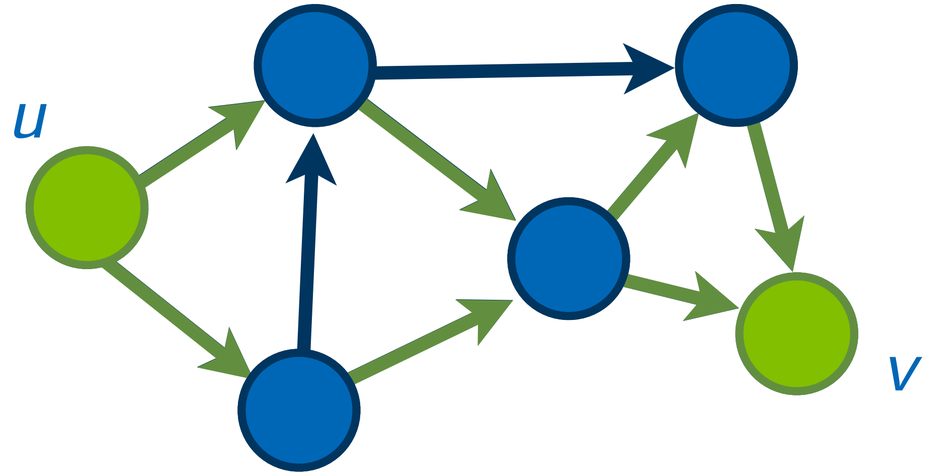
Our Problem: All-Pairs 2-Reachability

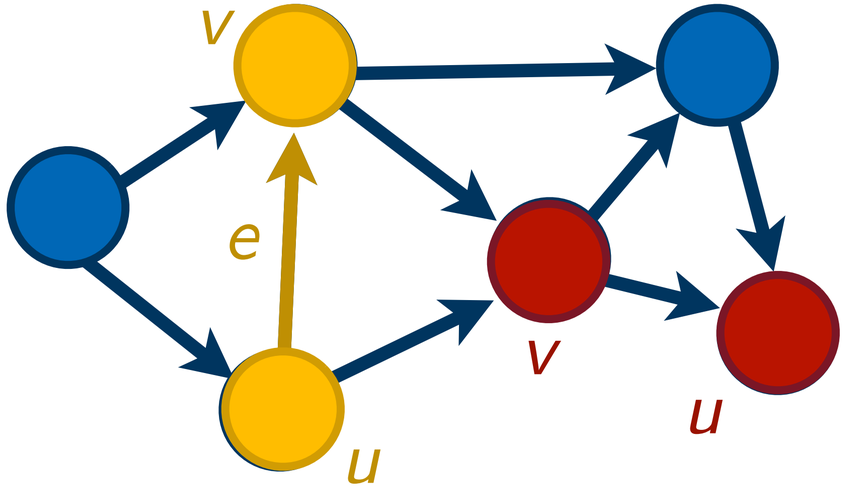
ETH zürich

Given: directed graph $G = (V, E)$
with n vertices, m edges

For all $u, v \in V$, decide if

- many edge-disjoint u-v-paths
- some edge e on every u-v-path
- no u-v-path at all



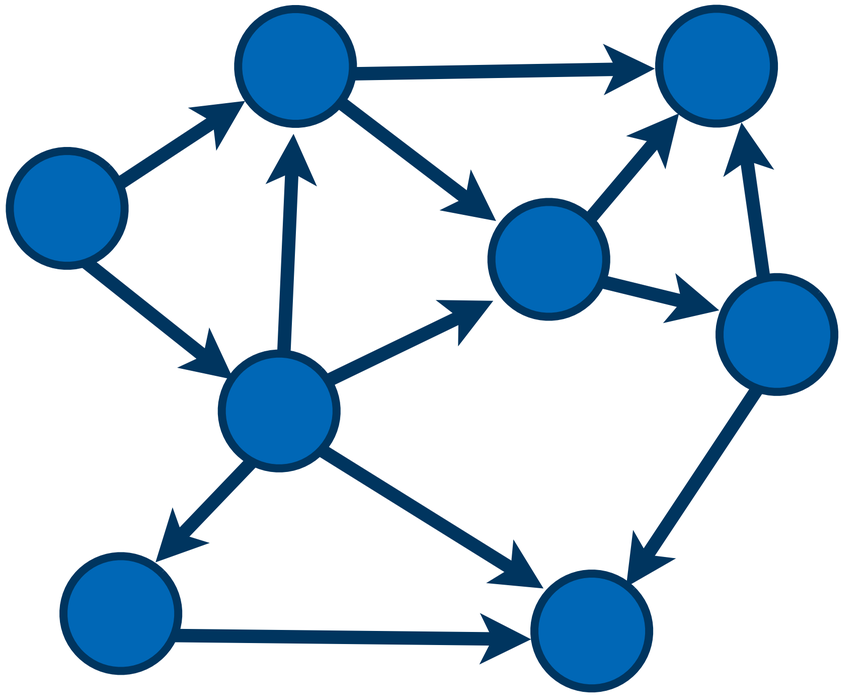


Goal: prepare for constant query time?

2-Reachability / Naive Solution

ETH zürich

Pseudocode

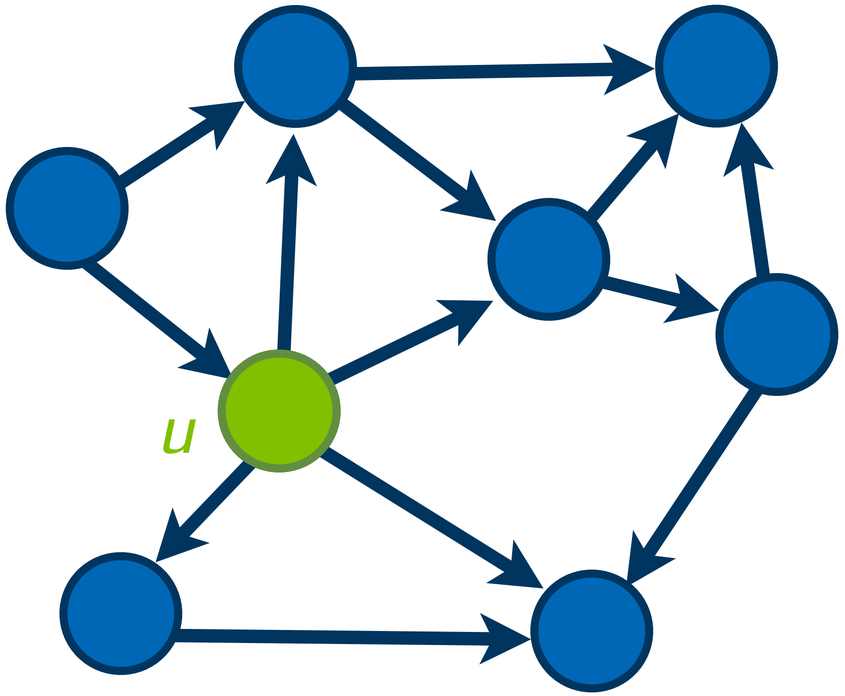


2-Reachability / Naive Solution

ETH zürich

Pseudocode

for every vertex u



2-Reachability / Naive Solution

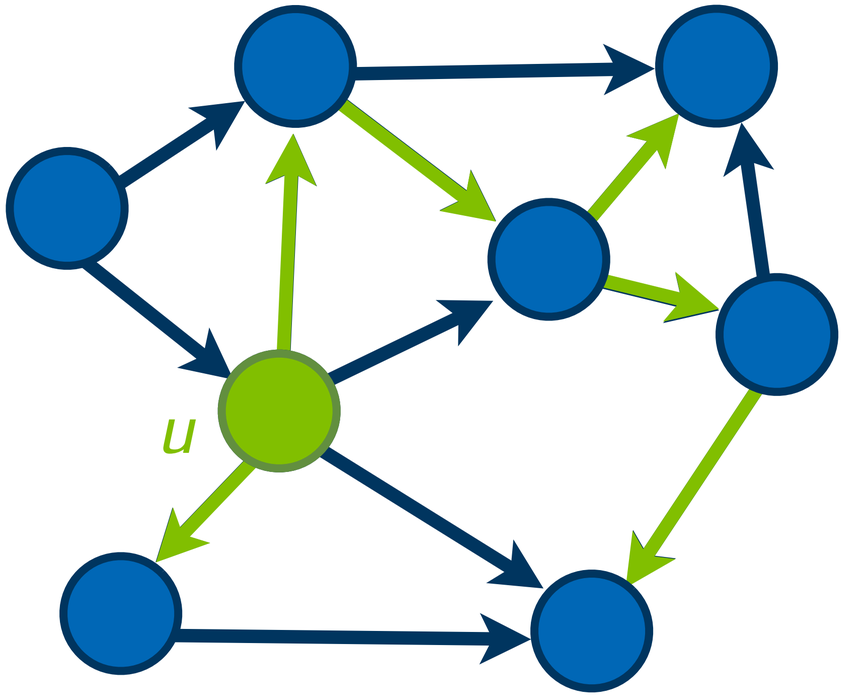
ETH zürich

Pseudocode

for every vertex u

    DFS from u

    remember reachability

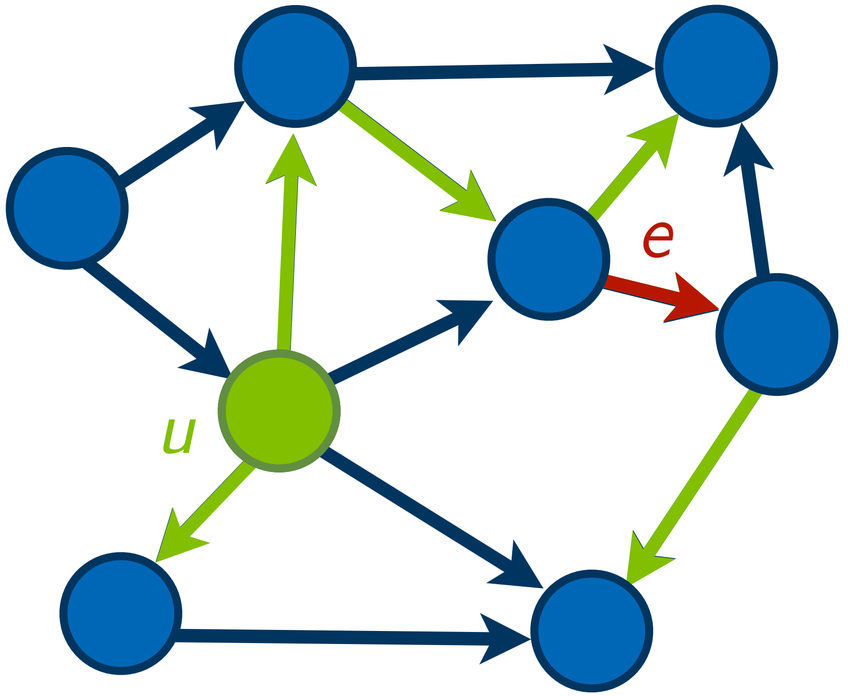


2-Reachability / Naive Solution

ETH zürich

Pseudocode

```
for every vertex u
    DFS from u
    remember reachability
    for every edge e
        
```

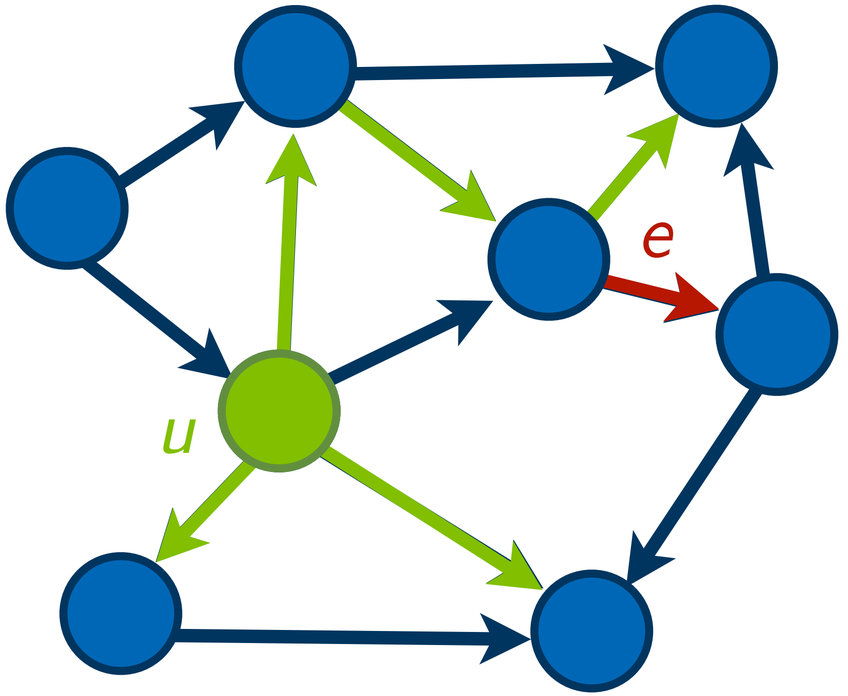


2-Reachability / Naive Solution

ETH zürich

Pseudocode

```
for every vertex u
    DFS from u
    remember reachability
    for every edge e
        DFS from u in G\e
        compare reachability
```

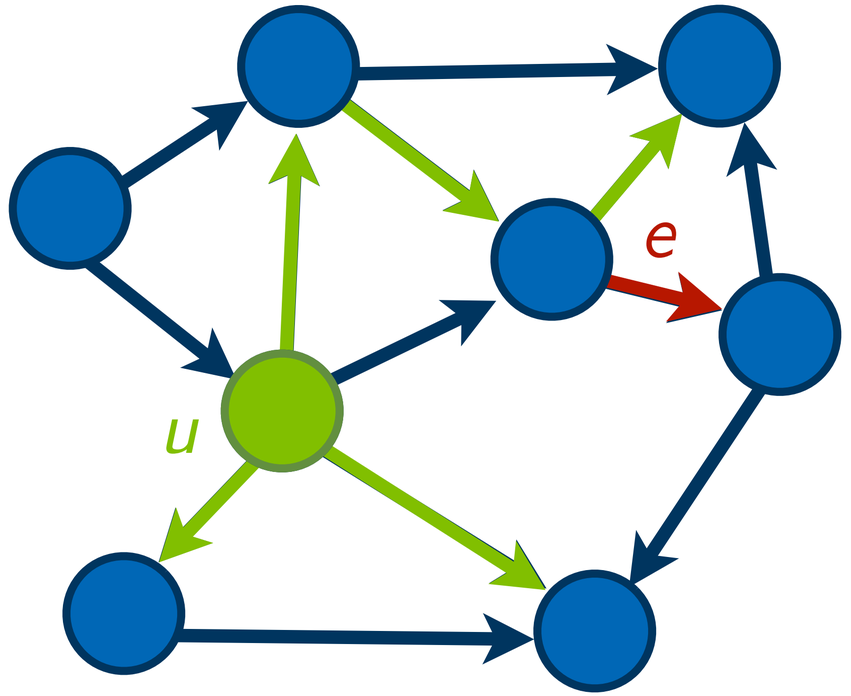


2-Reachability / Naive Solution

ETH zürich

Pseudocode

```
for every vertex u
    DFS from u
    remember reachability
    for every edge e
        DFS from u in G\e
        compare reachability
```



Time: $\mathcal{O}(n \cdot m \cdot m) \subseteq \mathcal{O}(n^5)$

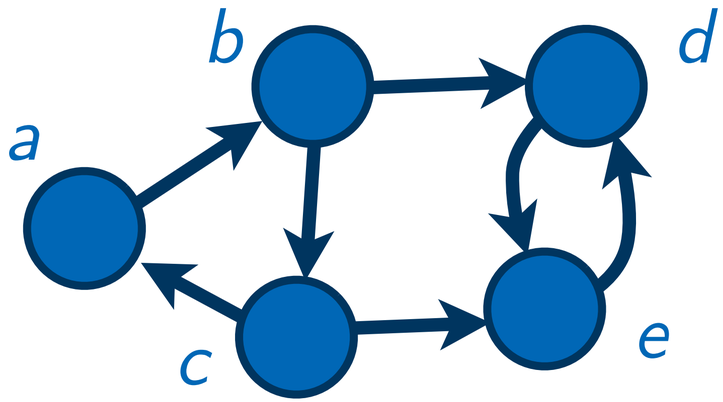
1-Reachability / Transitive Closure

ETH zürich

1-Reachability / Transitive Closure

ETH zürich

Given: directed graph $G = (V, E)$
with n vertices, m edges



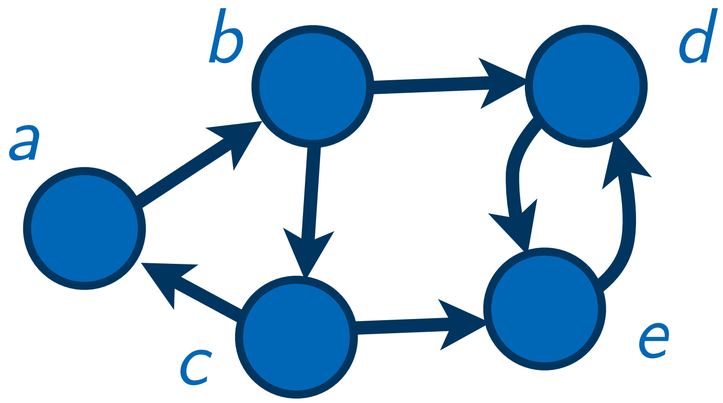
For all $u, v \in V$, decide if

- at least one u-v-path
- no u-v-path at all

1-Reachability / Transitive Closure

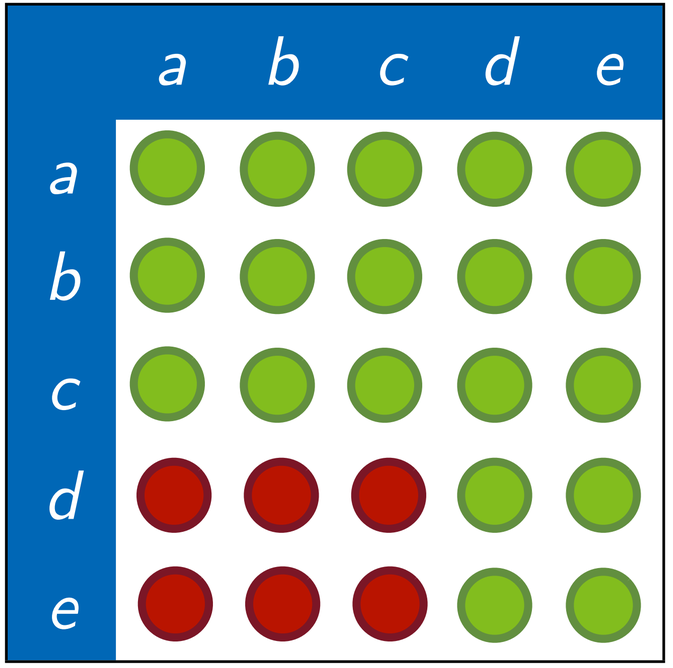
ETH zürich

Given: directed graph $G = (V, E)$
with n vertices, m edges



For all $u, v \in V$, decide if

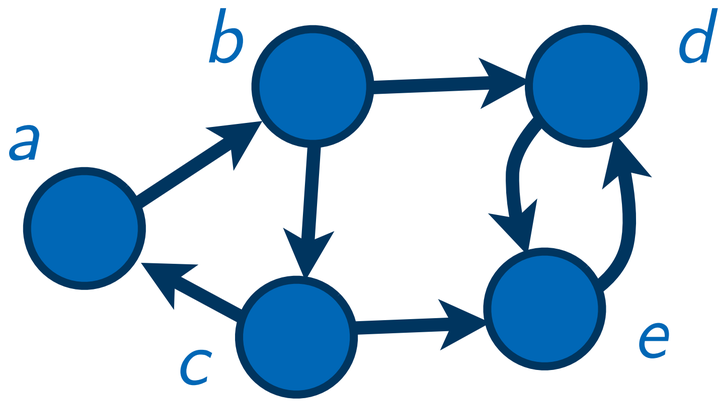
- at least one u-v-path
- no u-v-path at all



1-Reachability / Transitive Closure

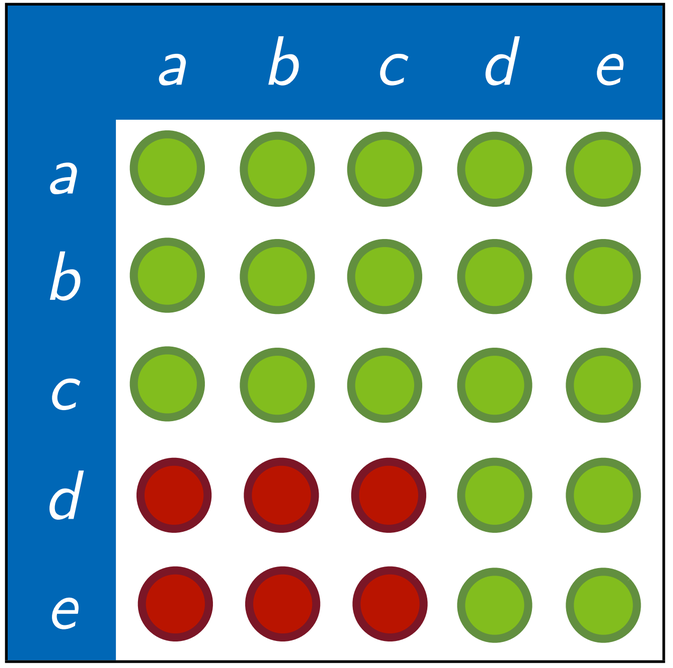
ETH zürich

Given: directed graph $G = (V, E)$
with n vertices, m edges



For all $u, v \in V$, decide if

- at least one u-v-path
- no u-v-path at all



Naive: DFS from every vertex
$\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

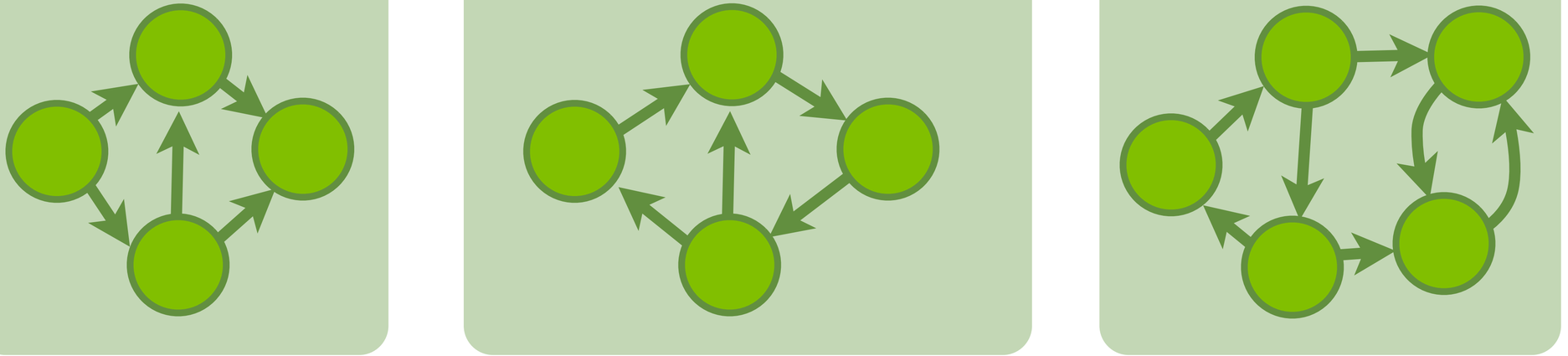
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm:

acyclic + strongly connected + combine



Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

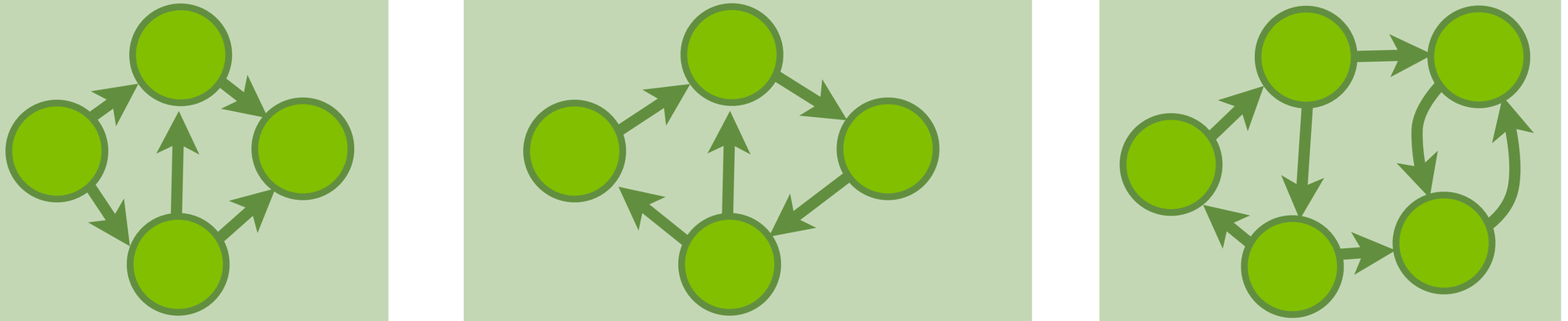
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm:

acyclic + strongly connected + combine



tricky trivial easy

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

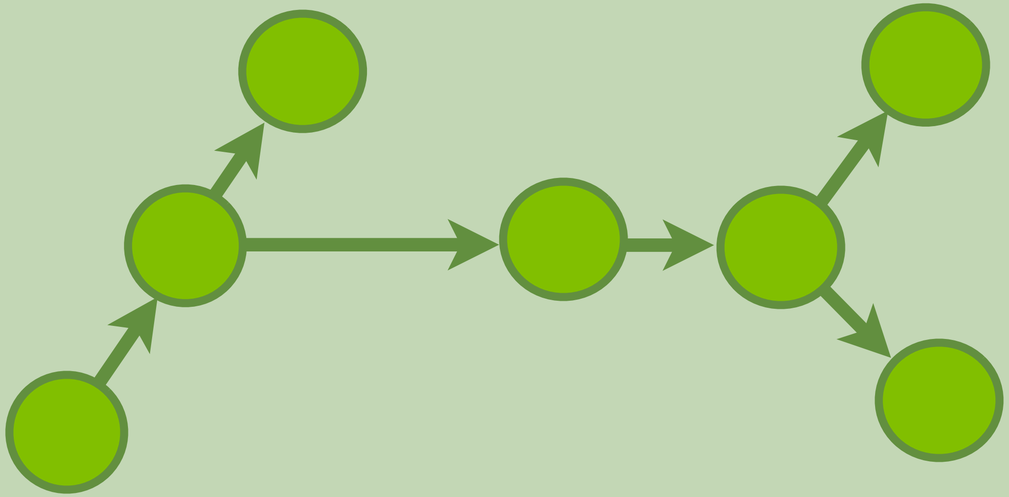
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

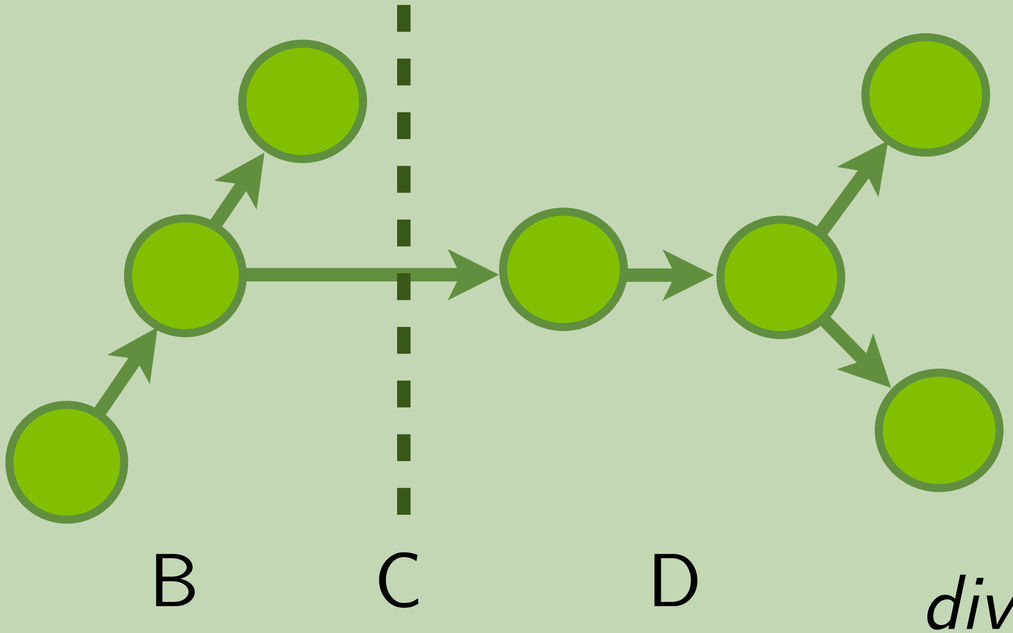
ETH zürich

Theorem TC as fast as Boolean matrix multiplication $\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



$$A = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}$$

divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

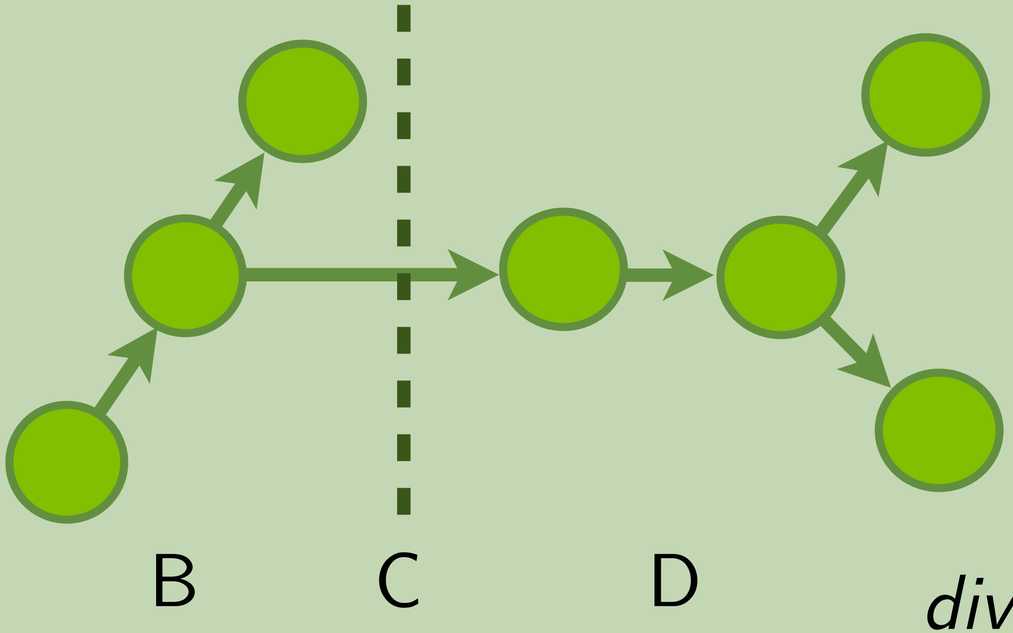
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



$$A^* = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}$$

divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

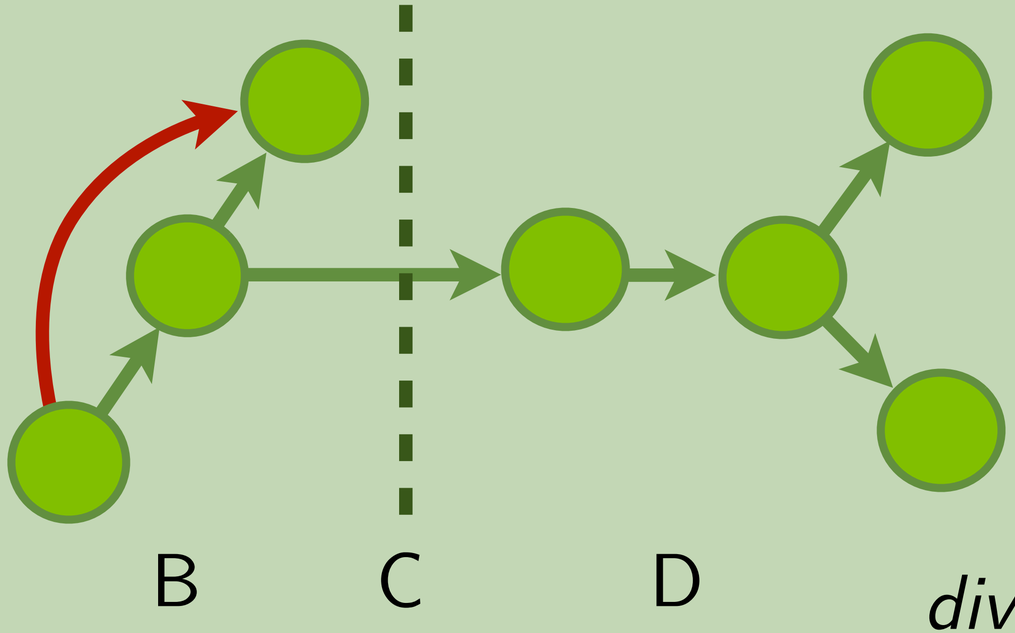
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



$$A^* = \begin{pmatrix} B^* & C \\ 0 & D \end{pmatrix}$$

divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

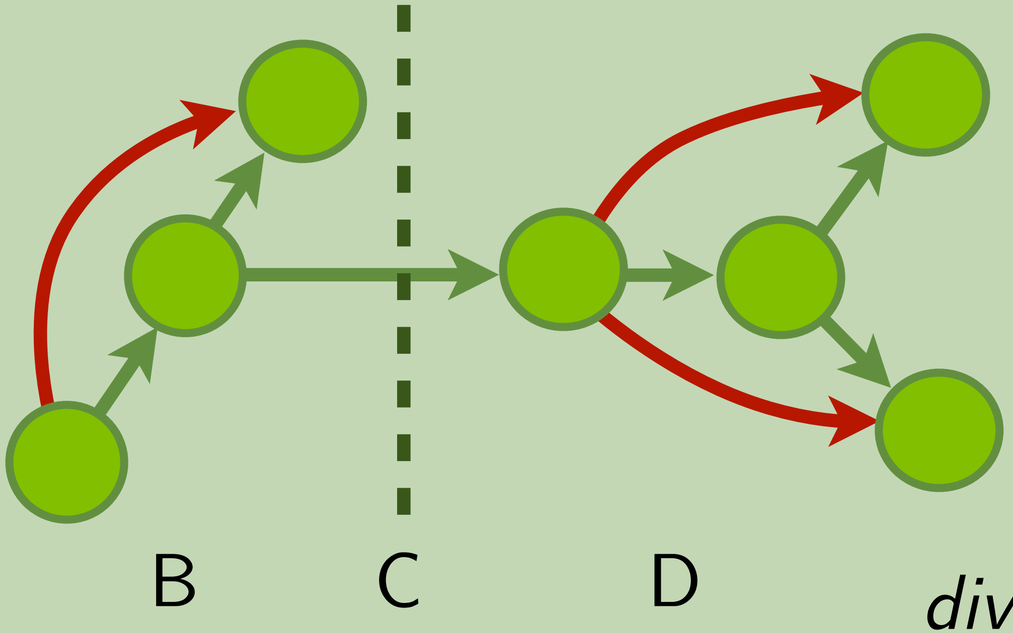
ETH zürich

Theorem TC as fast as Boolean matrix multiplication $\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



$$A^* = \begin{pmatrix} B^* & C \\ 0 & D^* \end{pmatrix}$$

divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

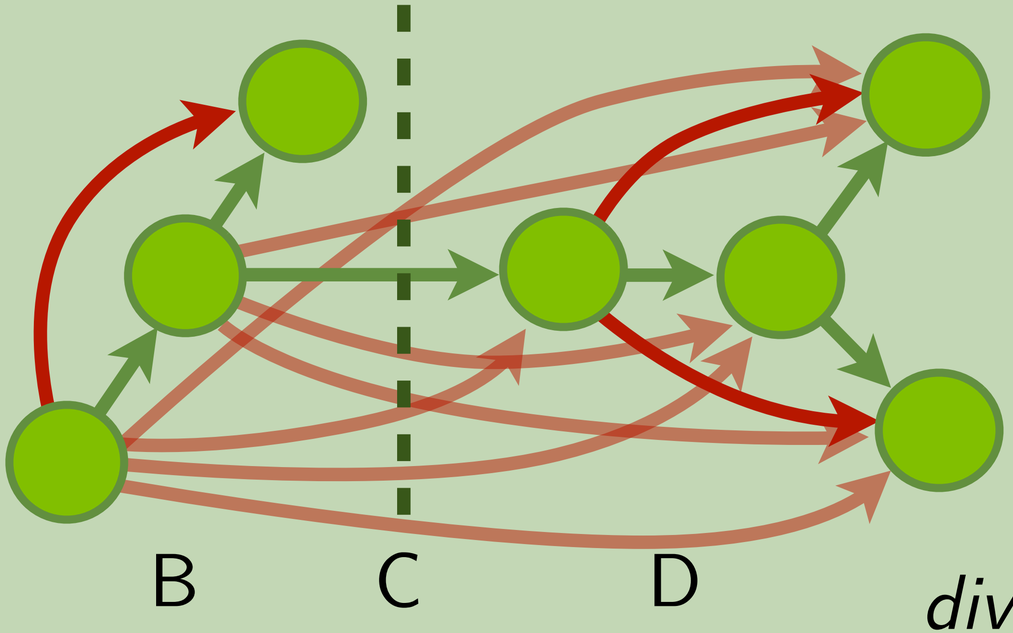
ETH zürich

Theorem TC as fast as Boolean matrix multiplication $\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



$$A^* = \begin{pmatrix} B^* & B^*CD^* \\ 0 & D^* \end{pmatrix}$$

divide & conquer along toposort order

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

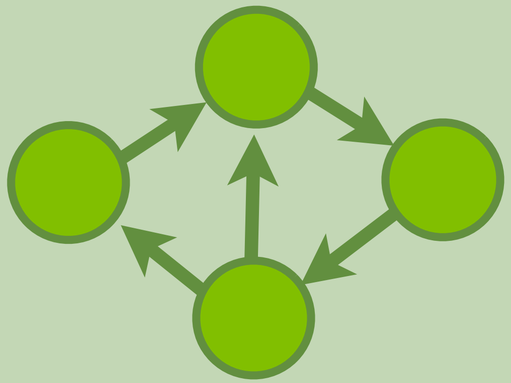
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^2)$



$A^* = \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$

clique

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

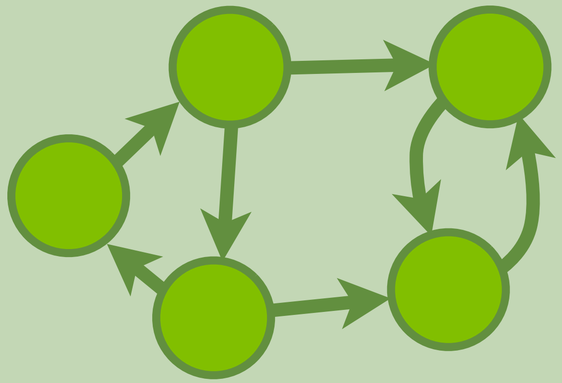
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



DAG of SCCs

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

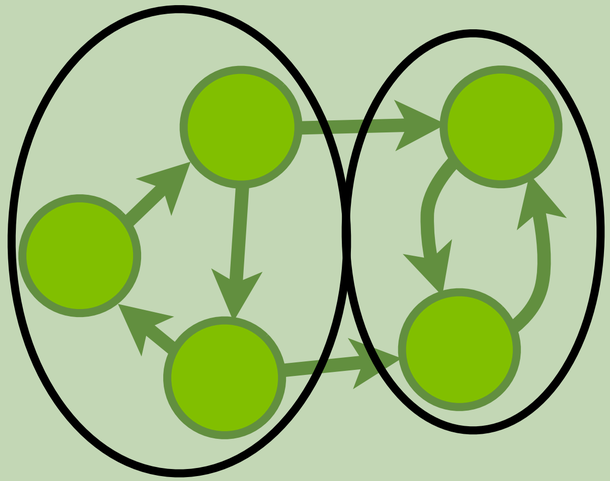
ETH zürich

Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$



DAG of SCCs

Fast Transitive Closure

Fischer, Meyer 1971
Munro, 1971

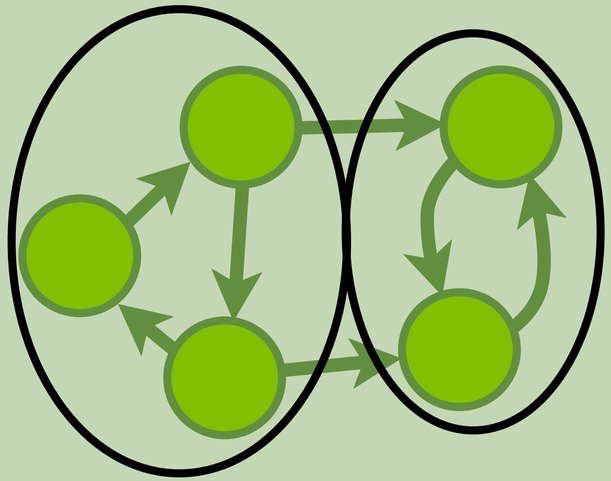
ETH zürich

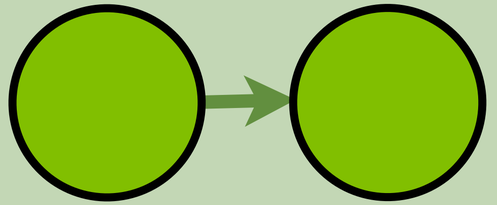
Theorem	TC as fast as Boolean matrix multiplication	$\mathcal{O}(n^\omega)$

$\omega < 2.373$

Algorithm: acyclic + strongly connected + combine

$\mathcal{O}(n^\omega)$





DAG of SCCs

No hope for $o(n^\omega)$

Fischer, Meyer 1971

ETH zürich

Theorem A·B reduces to transitive closure

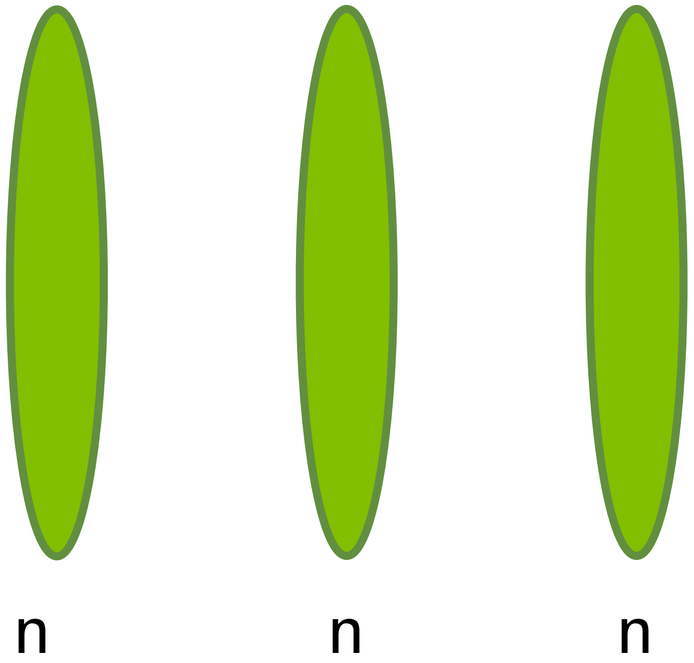
No hope for $o(n^\omega)$

Fischer, Meyer 1971

ETH zürich

Theorem A·B reduces to transitive closure

Proof:



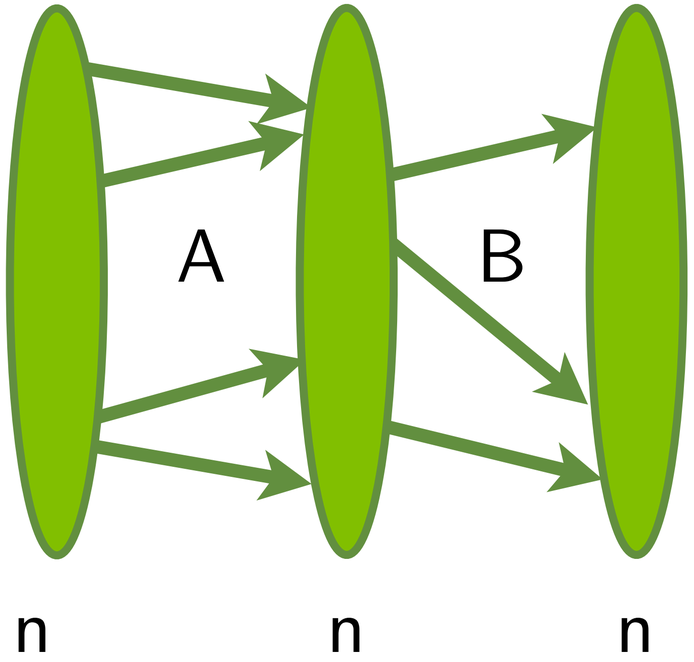
No hope for $o(n^\omega)$

Fischer, Meyer 1971

ETH zürich

Theorem A·B reduces to transitive closure

Proof:



$$\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}$$

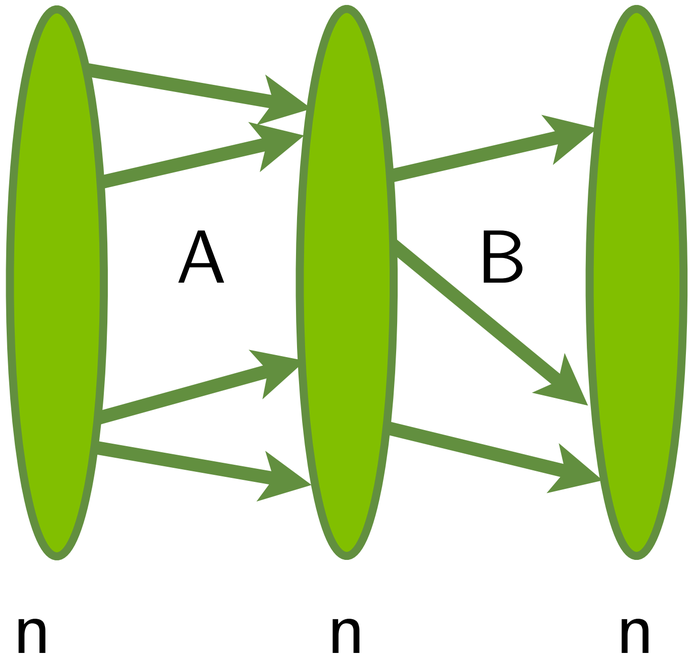
No hope for $o(n^\omega)$

Fischer, Meyer 1971

ETH zürich

Theorem A·B reduces to transitive closure

Proof:

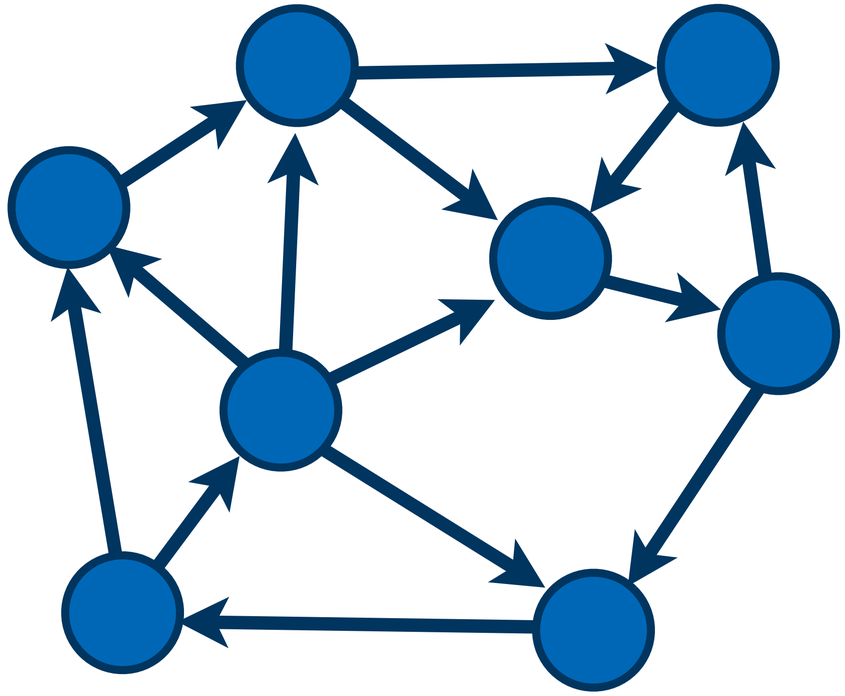


$$\begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}^* = \begin{pmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}$$

Improving 2-Reachability?

ETH zürich

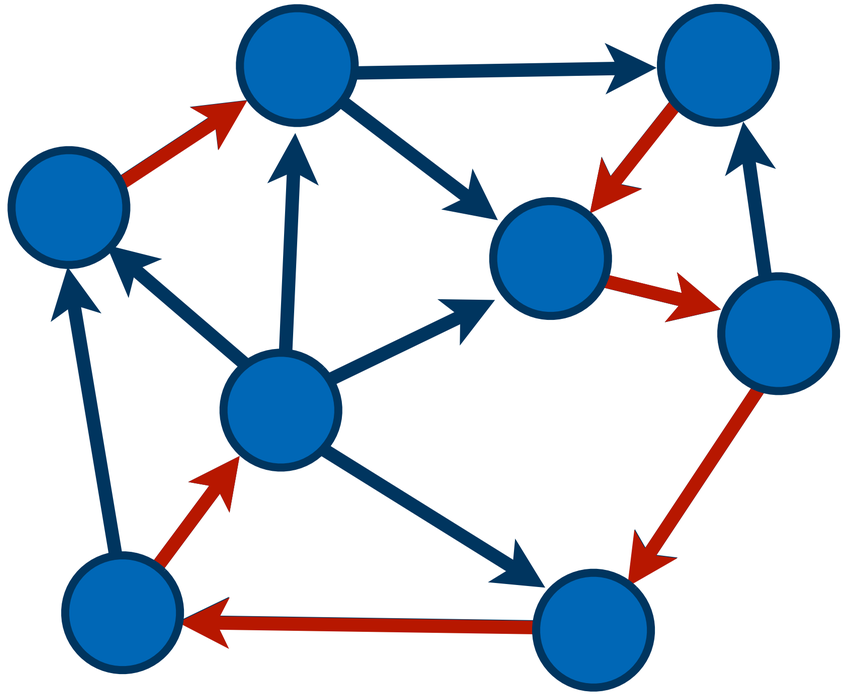
Strong bridge: edge whose removal increases number of SCCs



Improving 2-Reachability?

ETH zürich

Strong bridge: edge whose removal increases number of SCCs



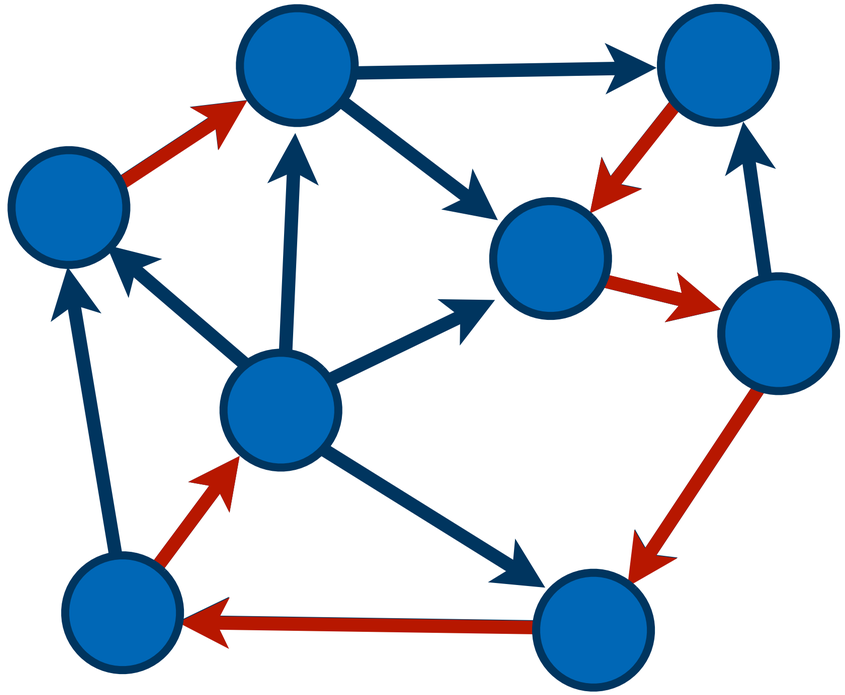
Improving 2-Reachability?

ETH zürich

Strong bridge: edge whose removal increases number of SCCs

Pseudocode for single SCC

```
for every vertex u
    DFS from u
    for every strong bridge e
        DFS from u in G\e
        compare reachability
```



$$\mathcal{O}(n \cdot n \cdot m) \subseteq \mathcal{O}(n^4)$$

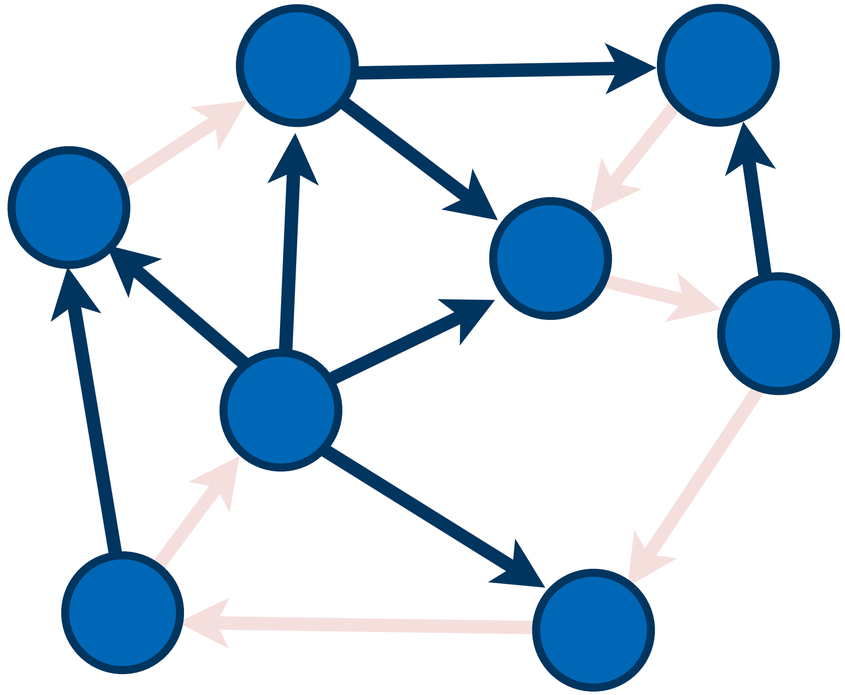
Improving 2-Reachability?

ETH zürich

Strong bridge: edge whose removal increases number of SCCs

Pseudocode for single SCC*

for every vertex u
- DFS from u
- ***remove all strong bridges***
- DFS from u in G‘
- compare reachability



*no witnesses

$$\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$$

Improving 2-Reachability?

ETH zürich

Strong bridge: edge whose removal increases number of SCCs

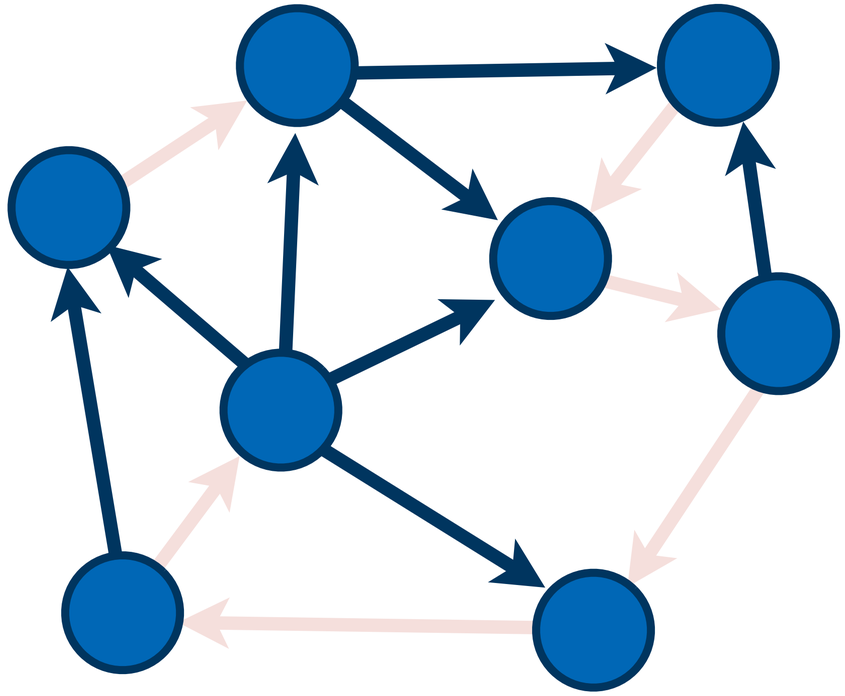
Pseudocode for single SCC*

compute transitive closure of G

remove all strong bridges

compute transitive closure of G‘

compare reachability



*no witnesses

$$\mathcal{O}(n^{\omega}) \subseteq \mathcal{O}(n^{2.373})$$

Improving 2-Reachability?

Strong bridge: edge whose removal increases number of SCCs

Pseudocode for single SCC*

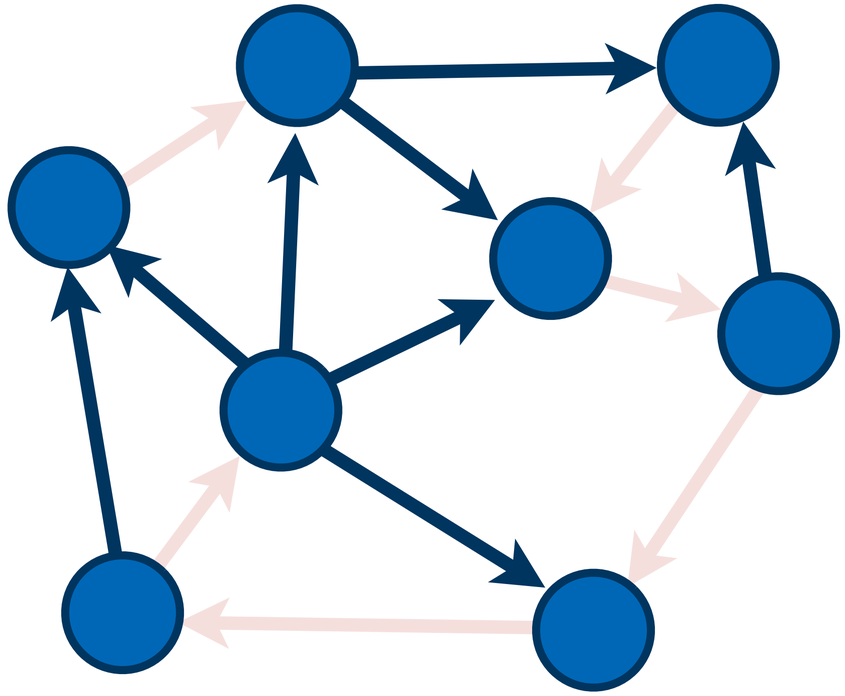
compute transitive closure of G

remove all strong bridges

compute transitive closure of G‘

compare reachability

WRONG



*no witnesses

$$\mathcal{O}(n^{\omega}) \subseteq \mathcal{O}(n^{2.373})$$

Improving 2-Reachability?

Strong bridge: edge whose removal increases number of SCCs

Pseudocode for single SCC*

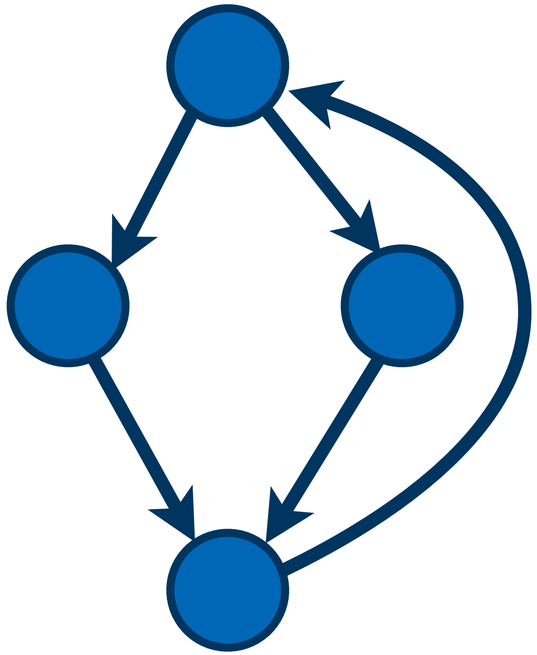
compute transitive closure of G

remove all strong bridges

compute transitive closure of G‘

compare reachability

WRONG



*no witnesses

$$\mathcal{O}(n^{\omega}) \subseteq \mathcal{O}(n^{2.373})$$

Improving 2-Reachability?

ETH zürich

Strong bridge: edge whose removal increases number of SCCs

Pseudocode for single SCC*

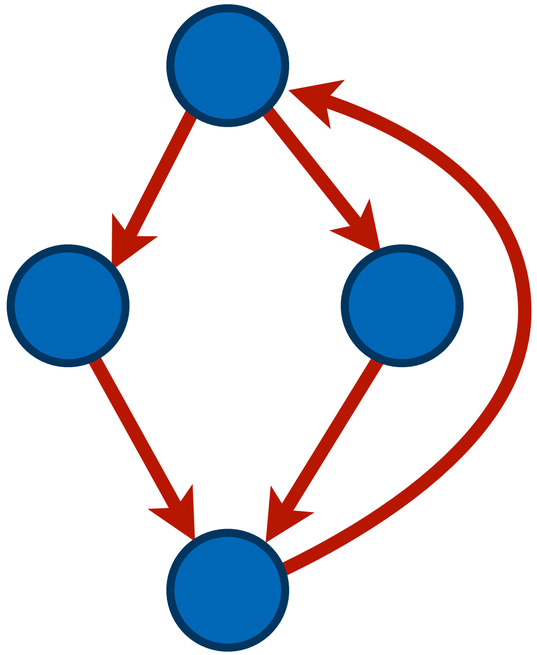
compute transitive closure of G

remove all strong bridges

compute transitive closure of G‘

compare reachability

WRONG



*no witnesses

$$\mathcal{O}(n^{\omega}) \subseteq \mathcal{O}(n^{2.373})$$

Improving 2-Reachability!

ETH zürich

Previously fastest: single source in $\mathcal{O}(m)$, all pairs in $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Improving 2-Reachability!

ETH zürich

[Alstrup, Harel, Lauridsen, Thorup 1990] ← non-trivial combinatorics, domination trees

Previously fastest: single source in $\mathcal{O}(m)$, all pairs in $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Improving 2-Reachability!

ETH zürich

[Alstrup, Harel, Lauridsen, Thorup 1990] ← non-trivial combinatorics, domination trees

Previously fastest: single source in $\mathcal{O}(m)$, all pairs in $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Main Result	All-pairs 2-reachability with witnesses in	$\mathcal{O}(n^\omega \log n)$

Improving 2-Reachability!

ETH zürich

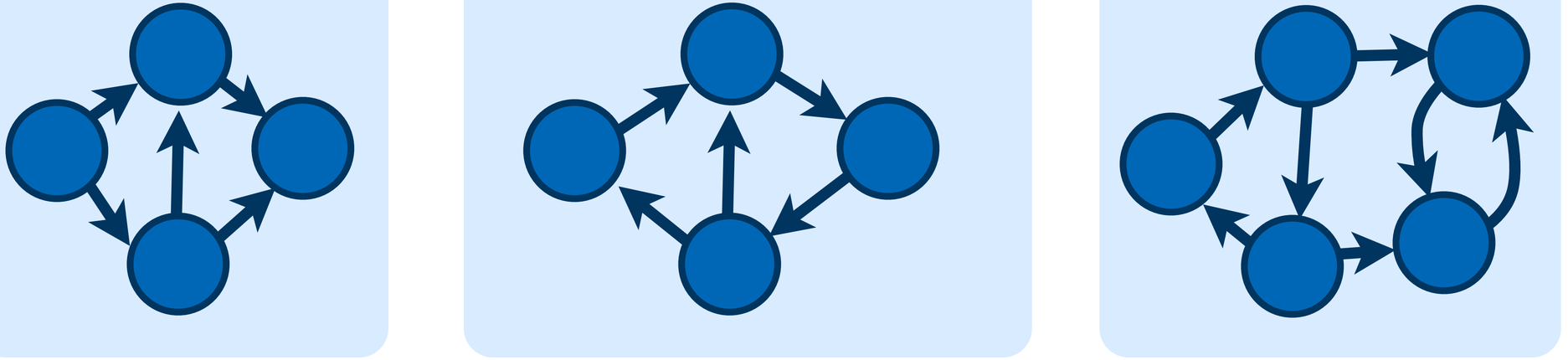
[Alstrup, Harel, Lauridsen, Thorup 1990] ← non-trivial combinatorics, domination trees

Previously fastest: single source in $\mathcal{O}(m)$, all pairs in $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Main Result All-pairs 2-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

Algorithm:

acyclic + strongly connected + combine



Improving 2-Reachability!

ETH zürich

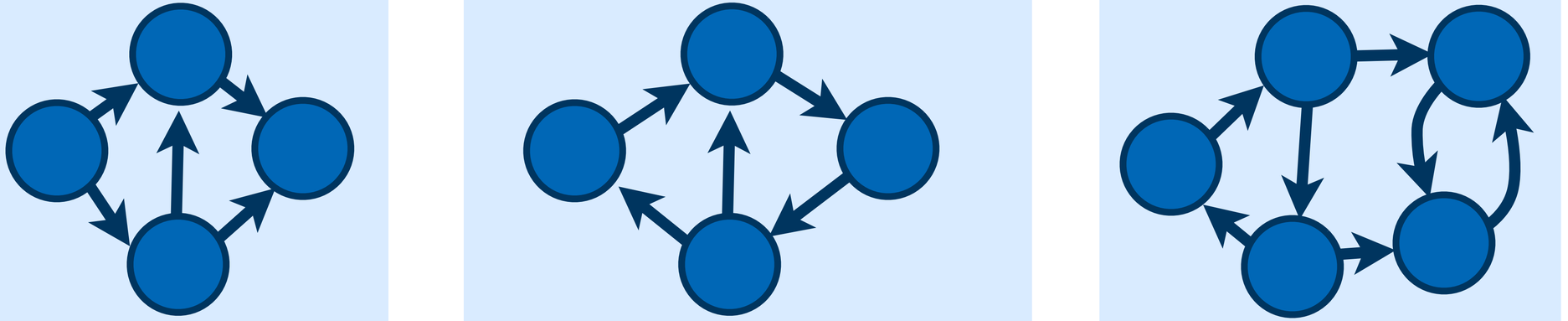
[Alstrup, Harel, Lauridsen, Thorup 1990] ← non-trivial combinatorics, domination trees

Previously fastest: single source in $\mathcal{O}(m)$, all pairs in $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

Main Result All-pairs 2-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

Algorithm:

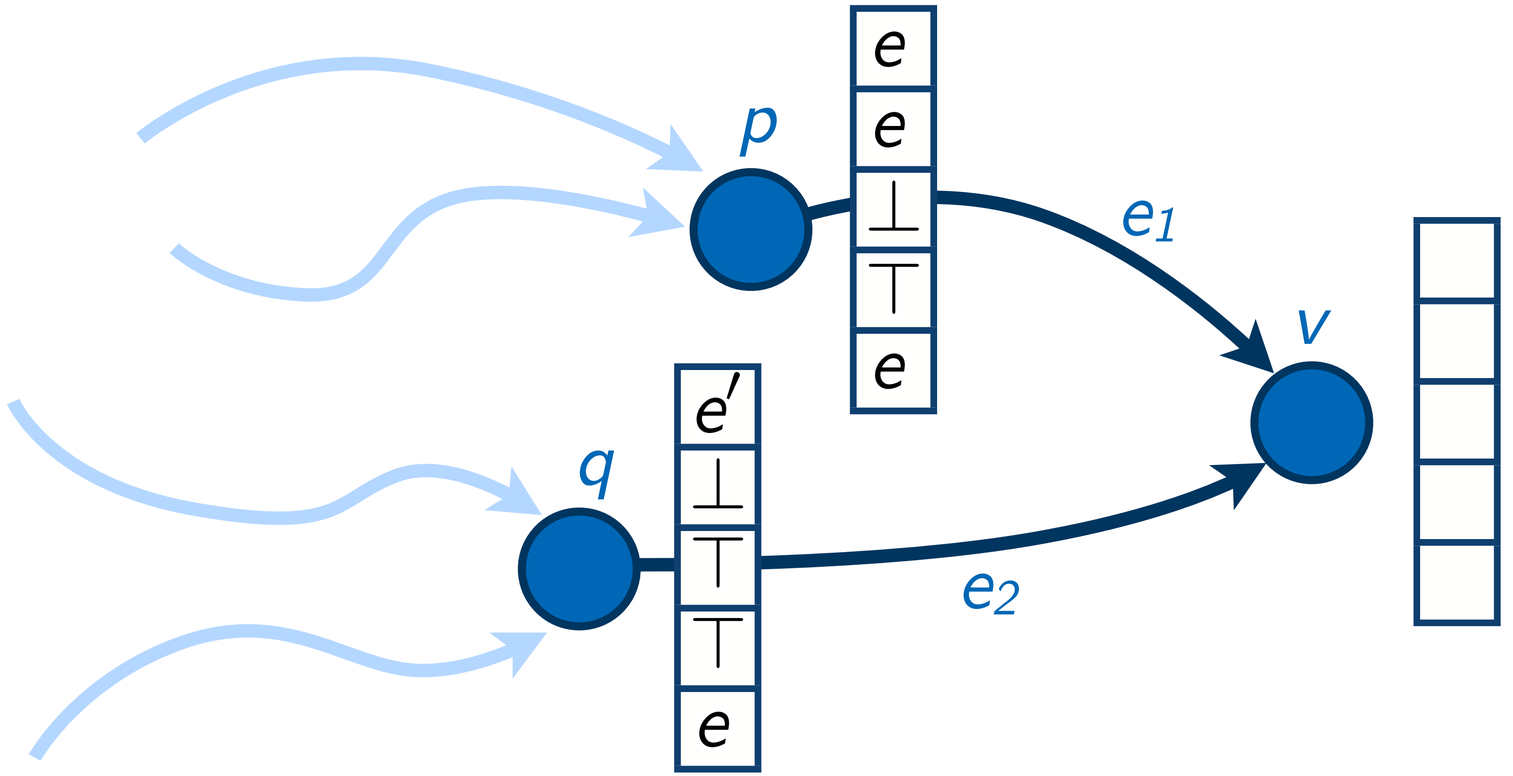
acyclic + strongly connected + combine



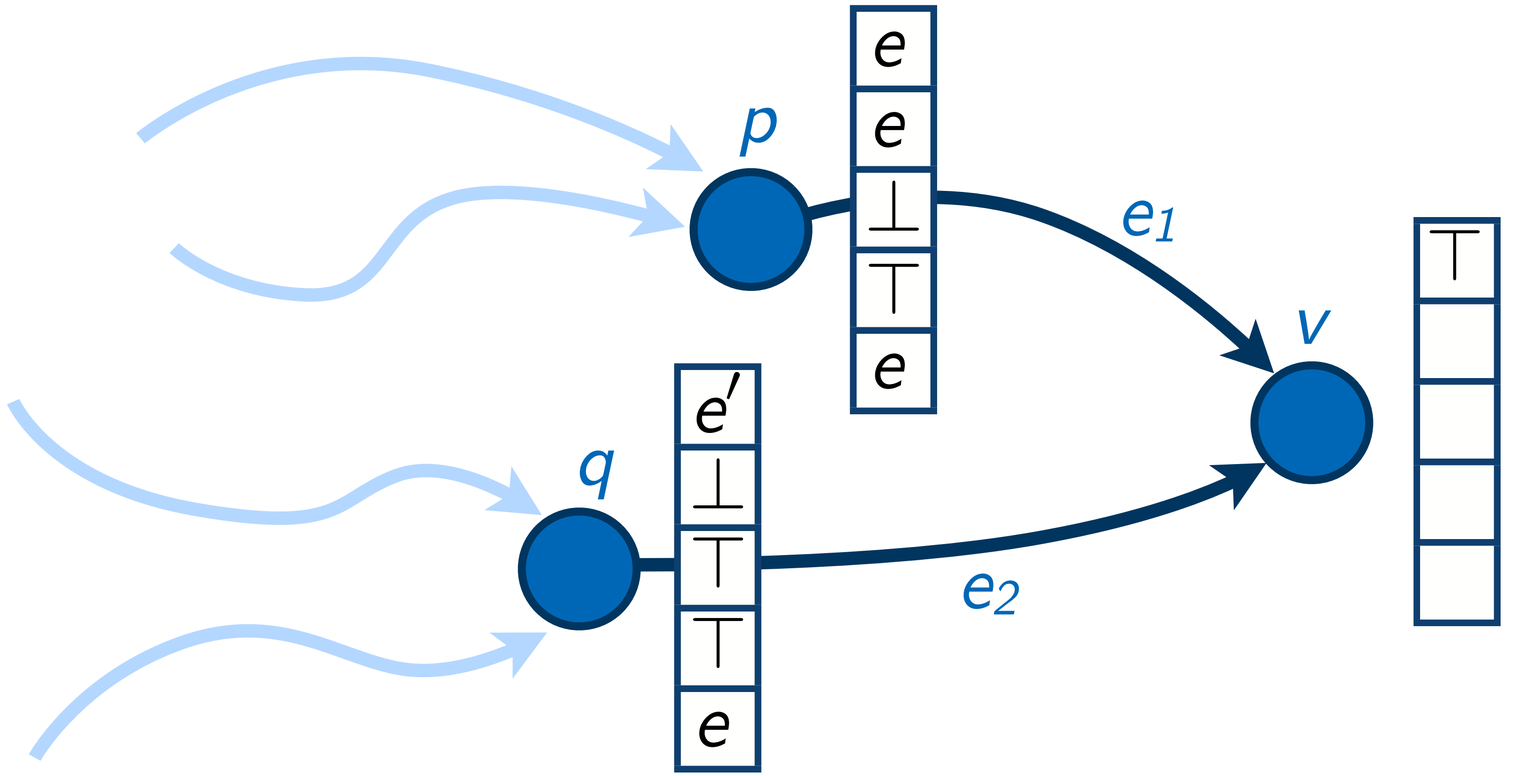
tricky + tricky + non-trivial

State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$

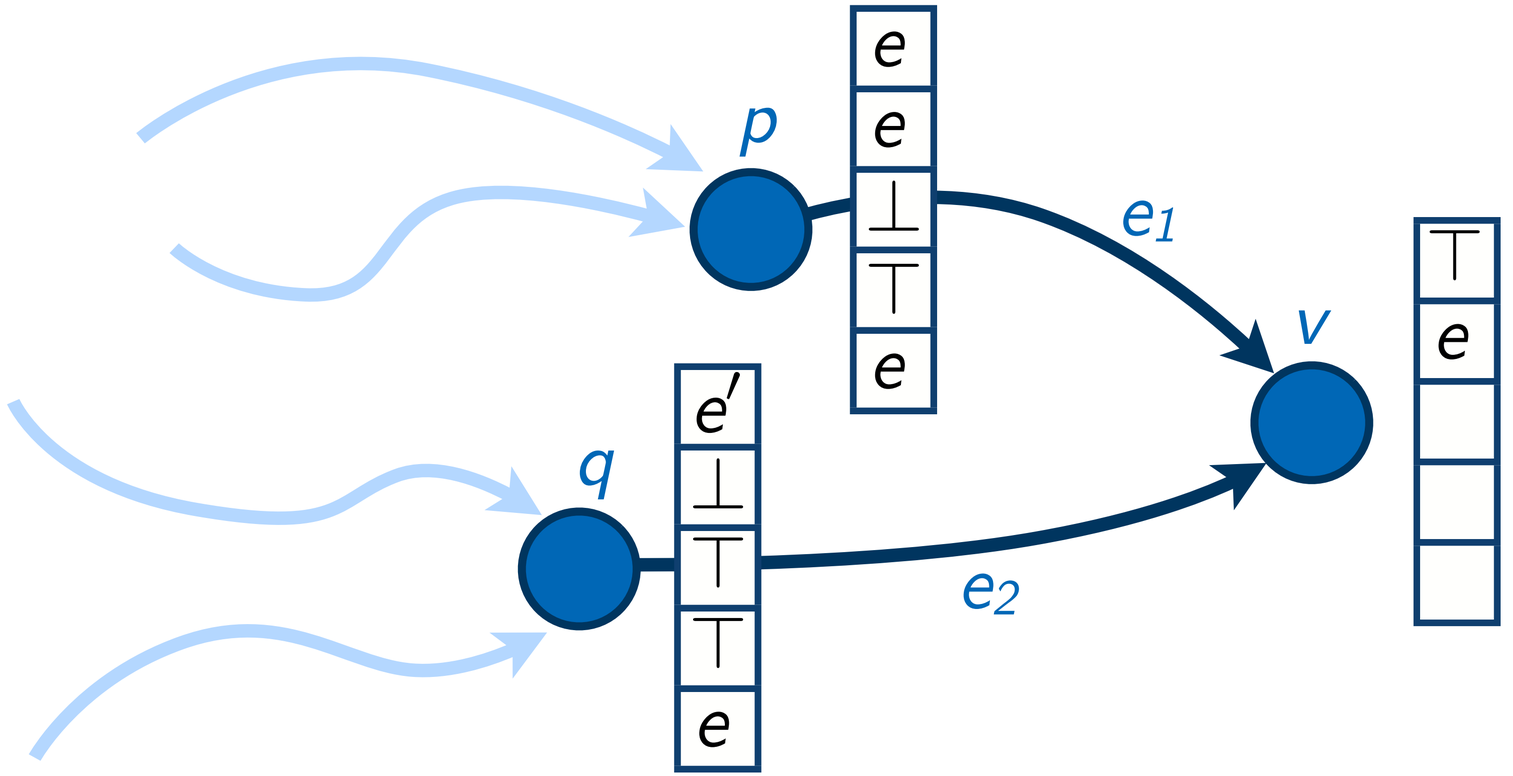
State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$



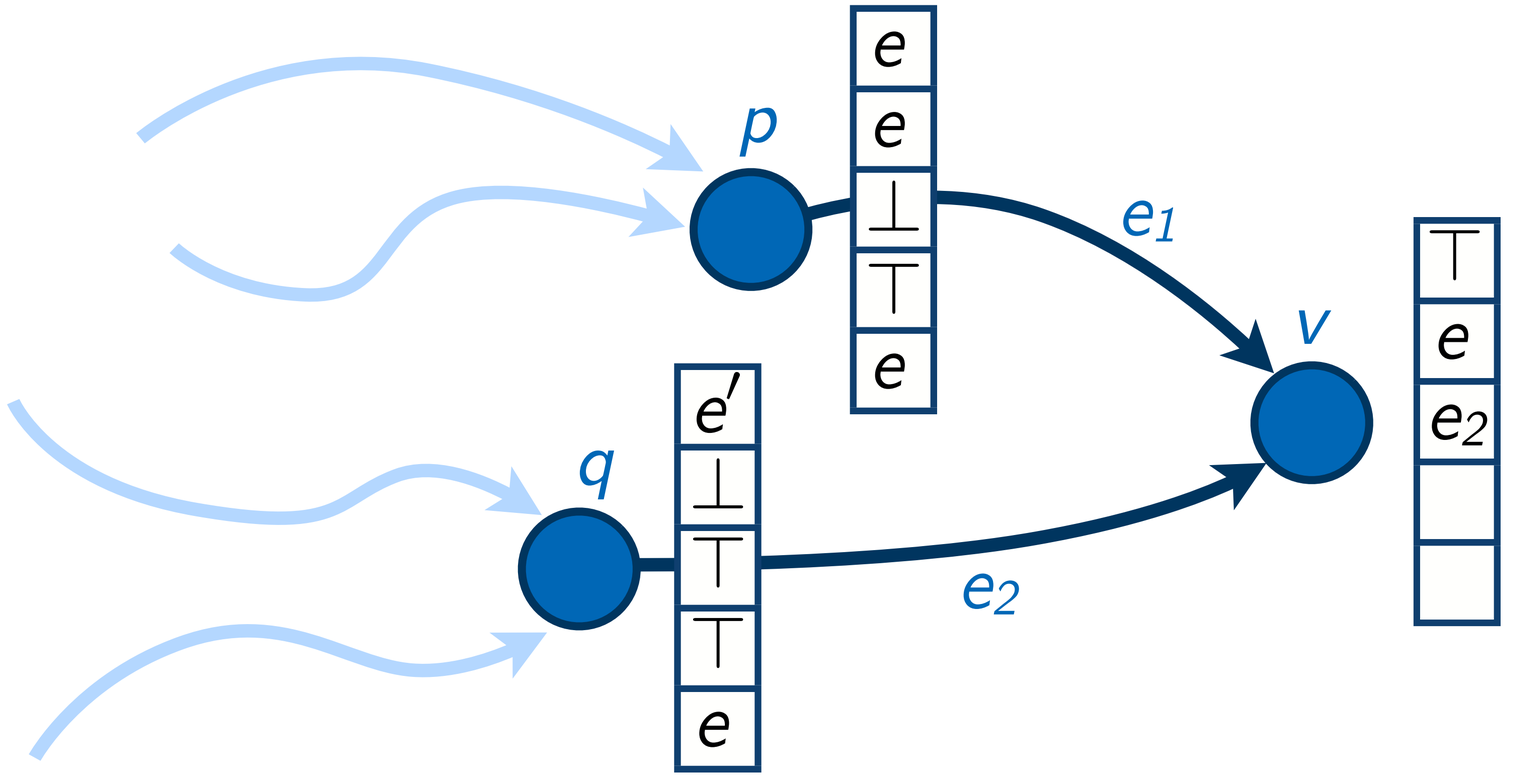
State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$



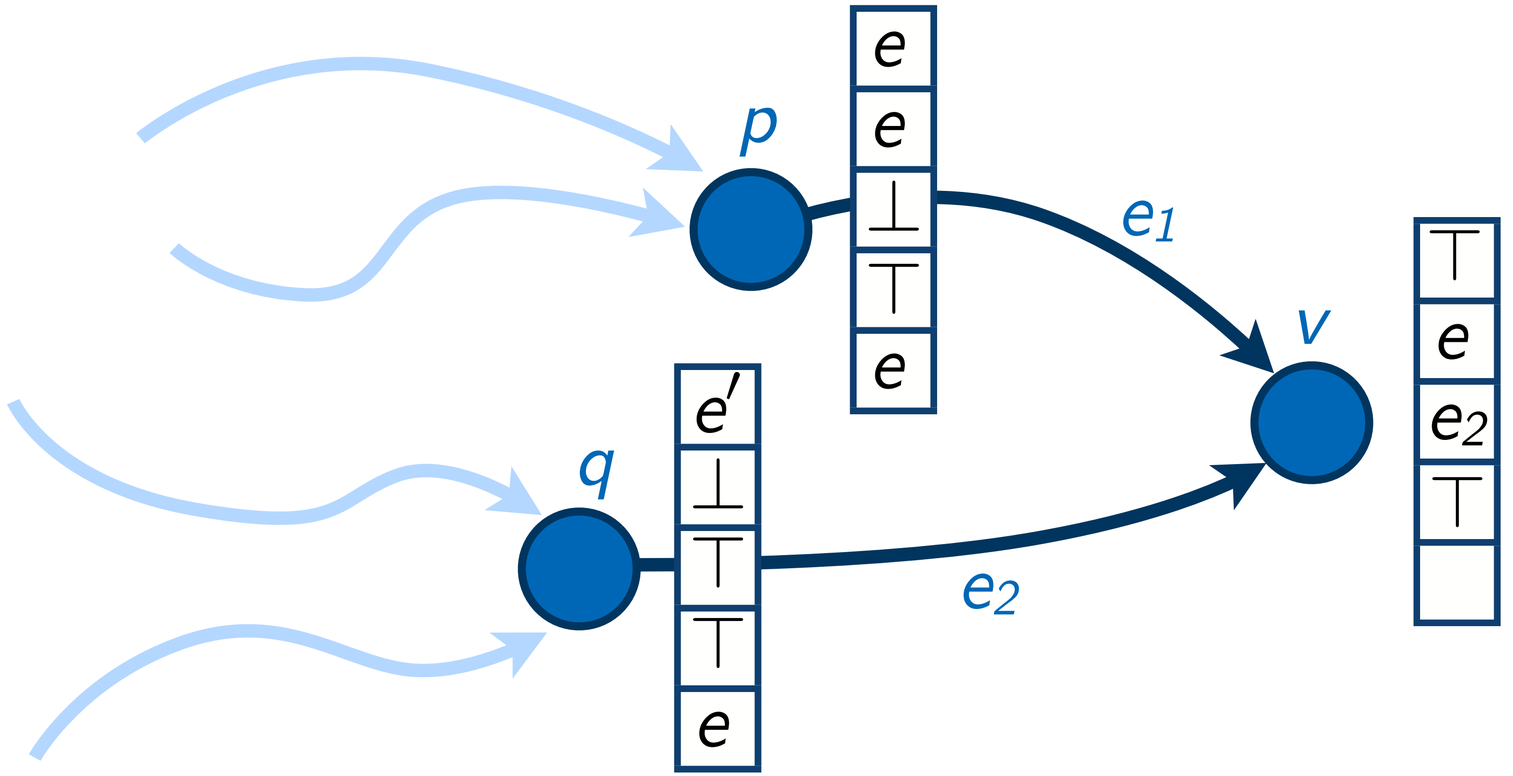
State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$



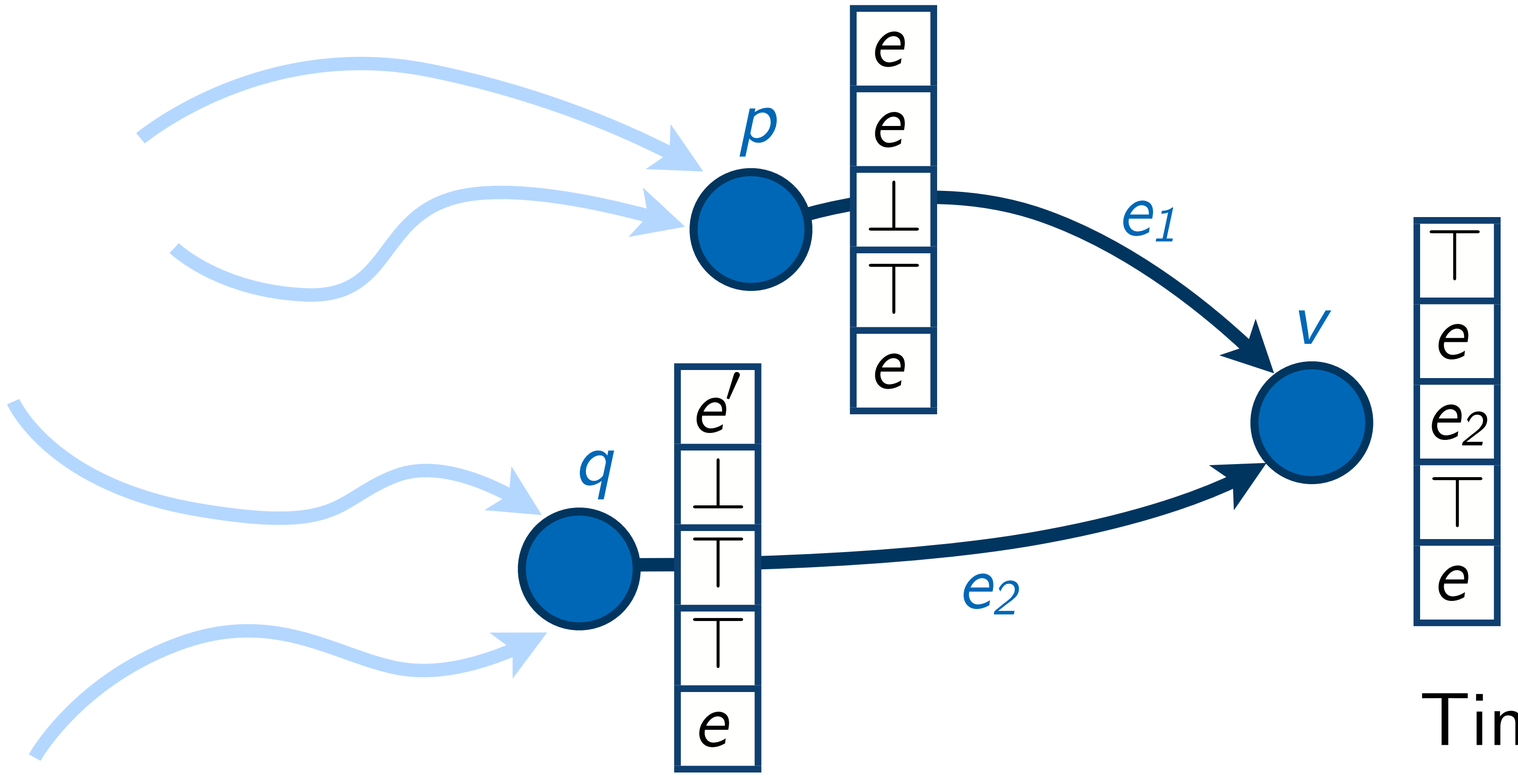
State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$



State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$

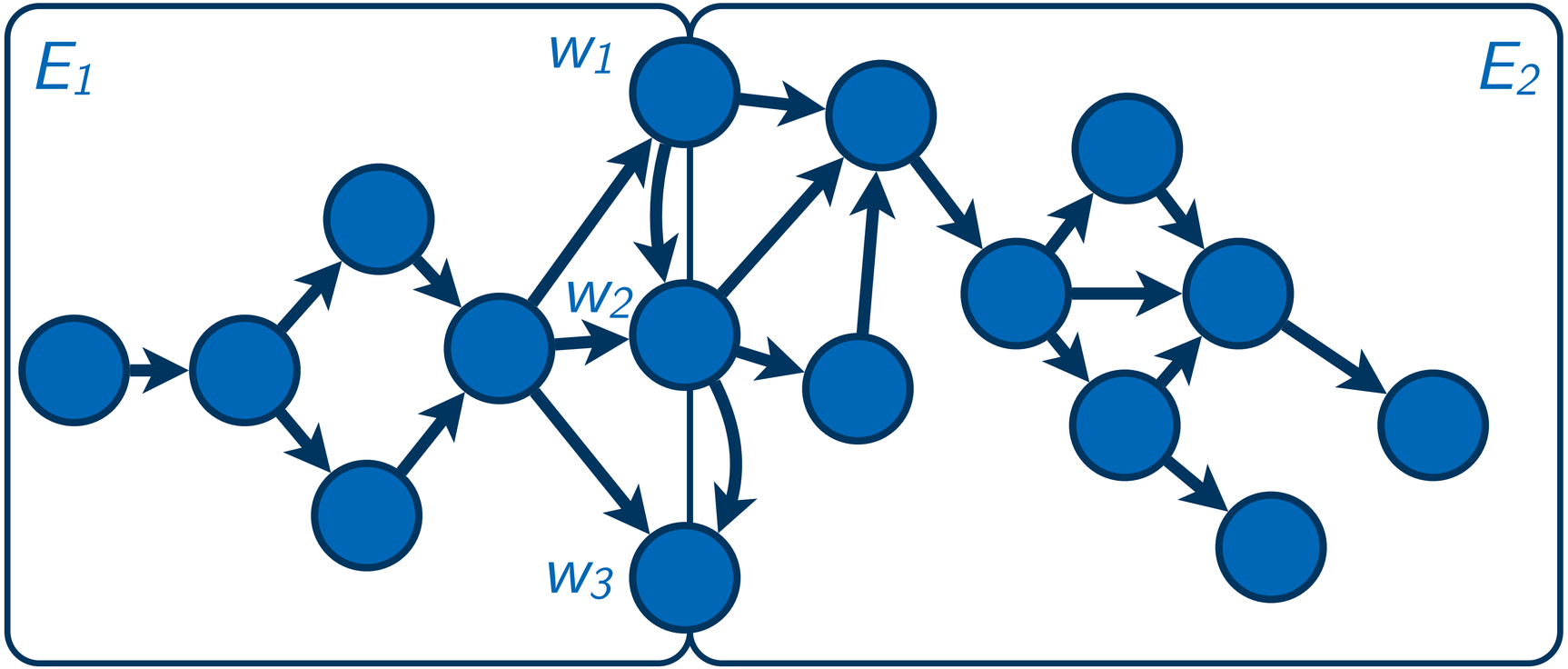


State: $L[u,v] = \begin{cases} \bot & \text{no } u\text{-}v\text{-path} \\ e & \text{first common edge of all } u\text{-}v\text{-paths} \\ \top & \text{multiple disjoint } u\text{-}v\text{-paths} \end{cases}$

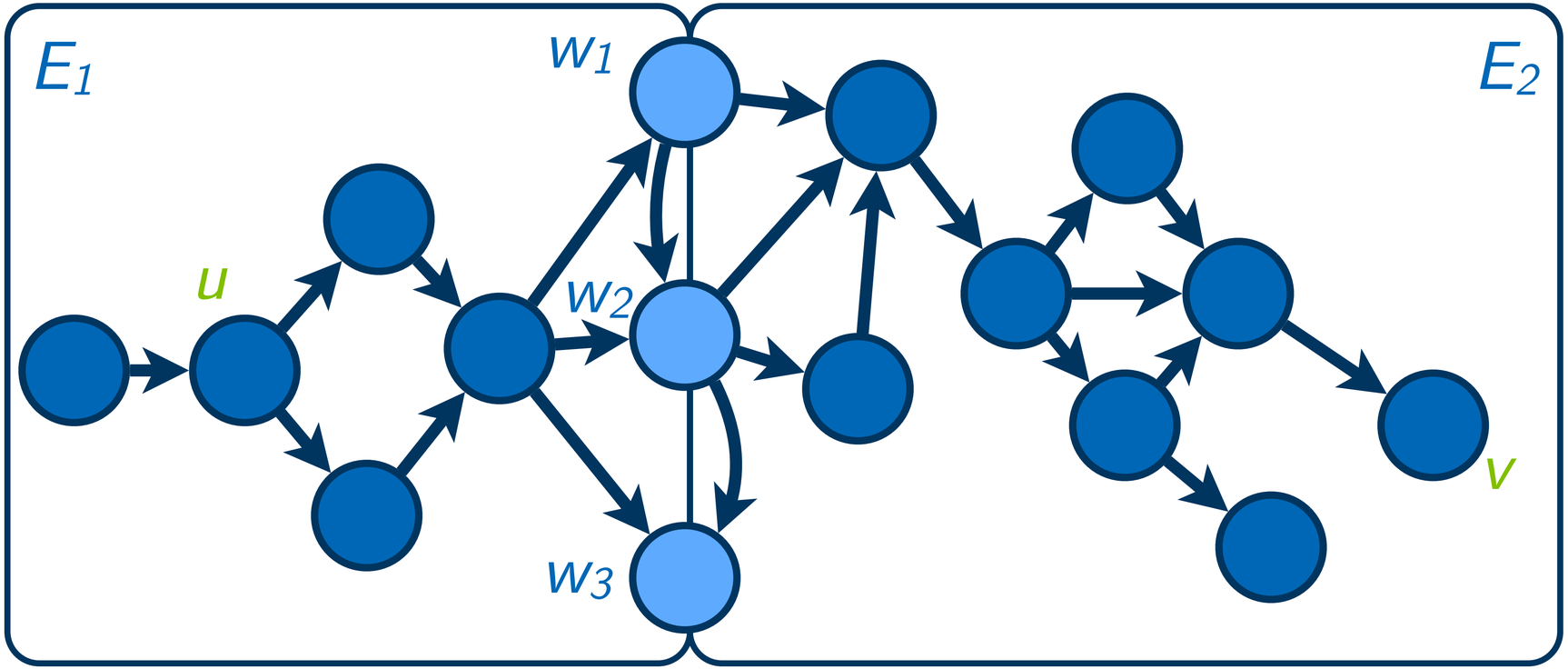


Time: $\mathcal{O}(n \cdot m) \subseteq \mathcal{O}(n^3)$

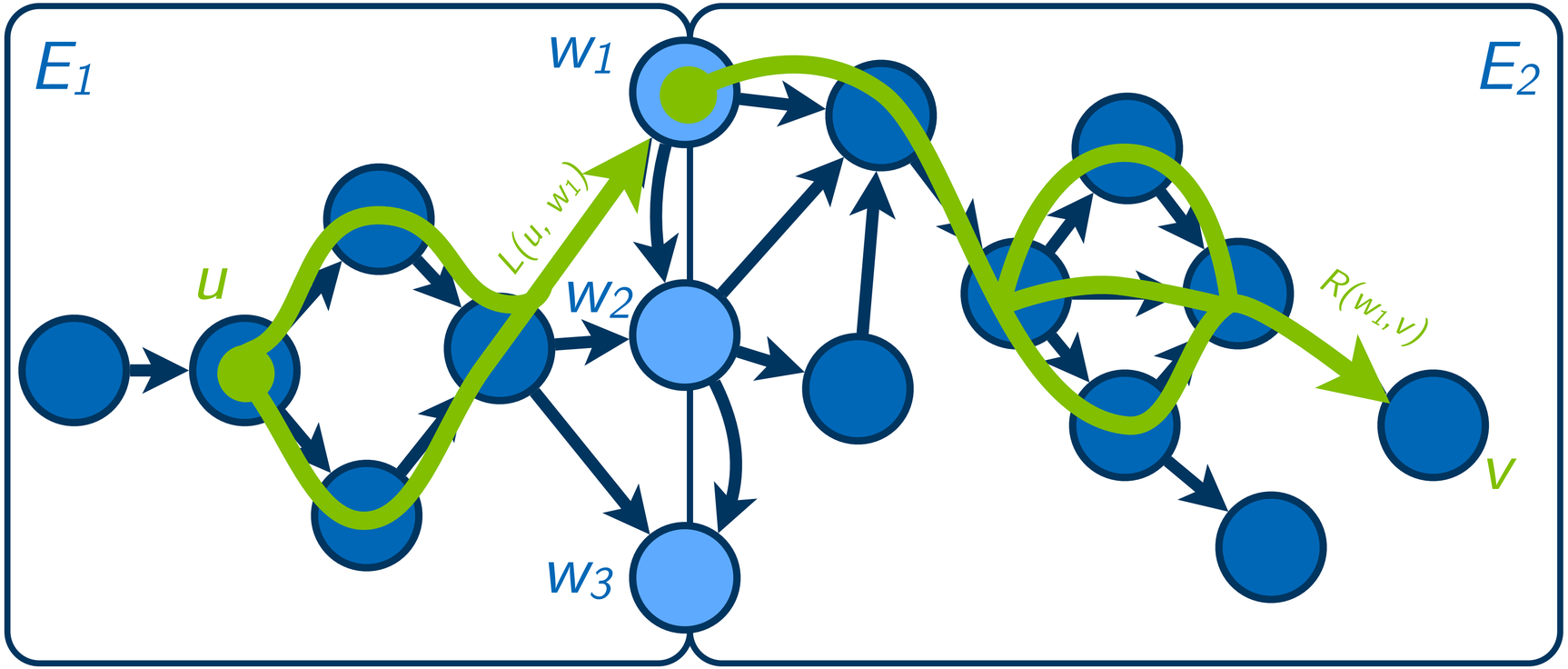
Edge split: $E = E_1 \dot\cup E_2$ once you leave E_1, you can never go back



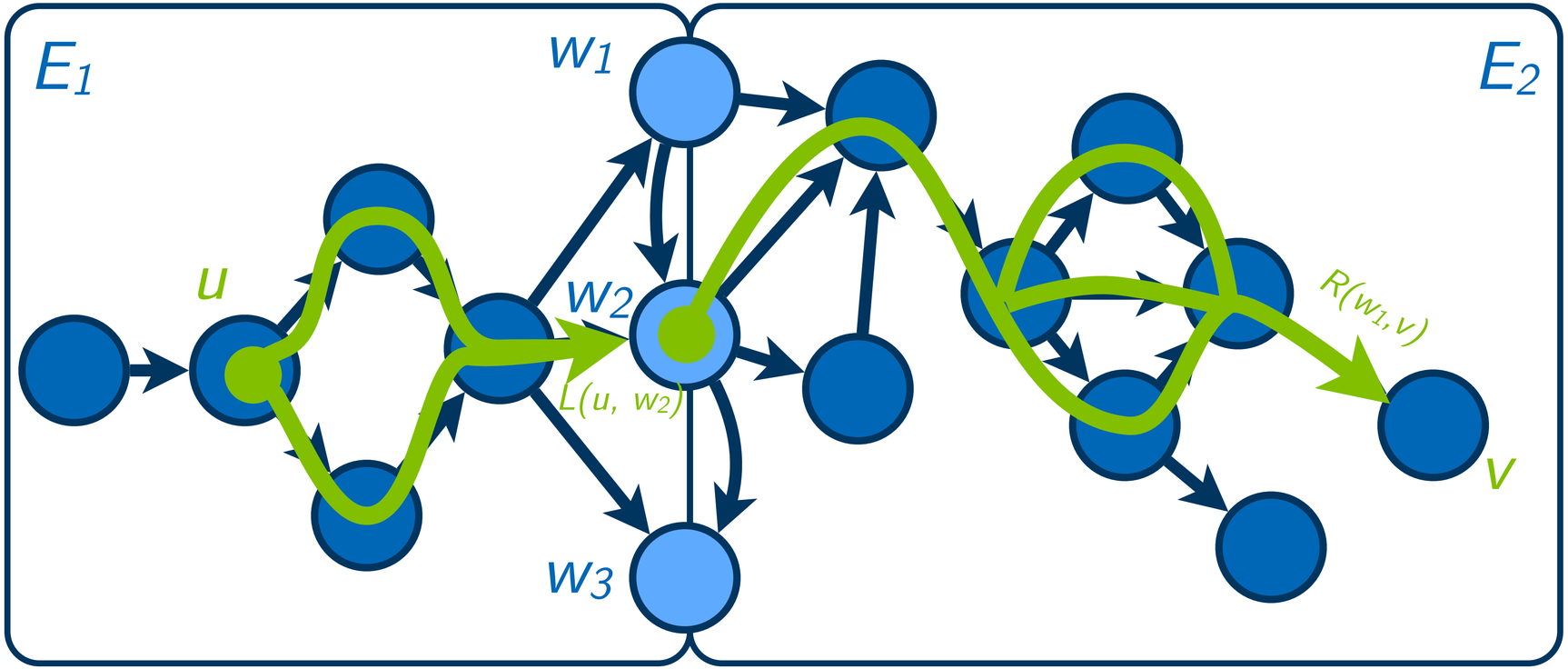
Path families:



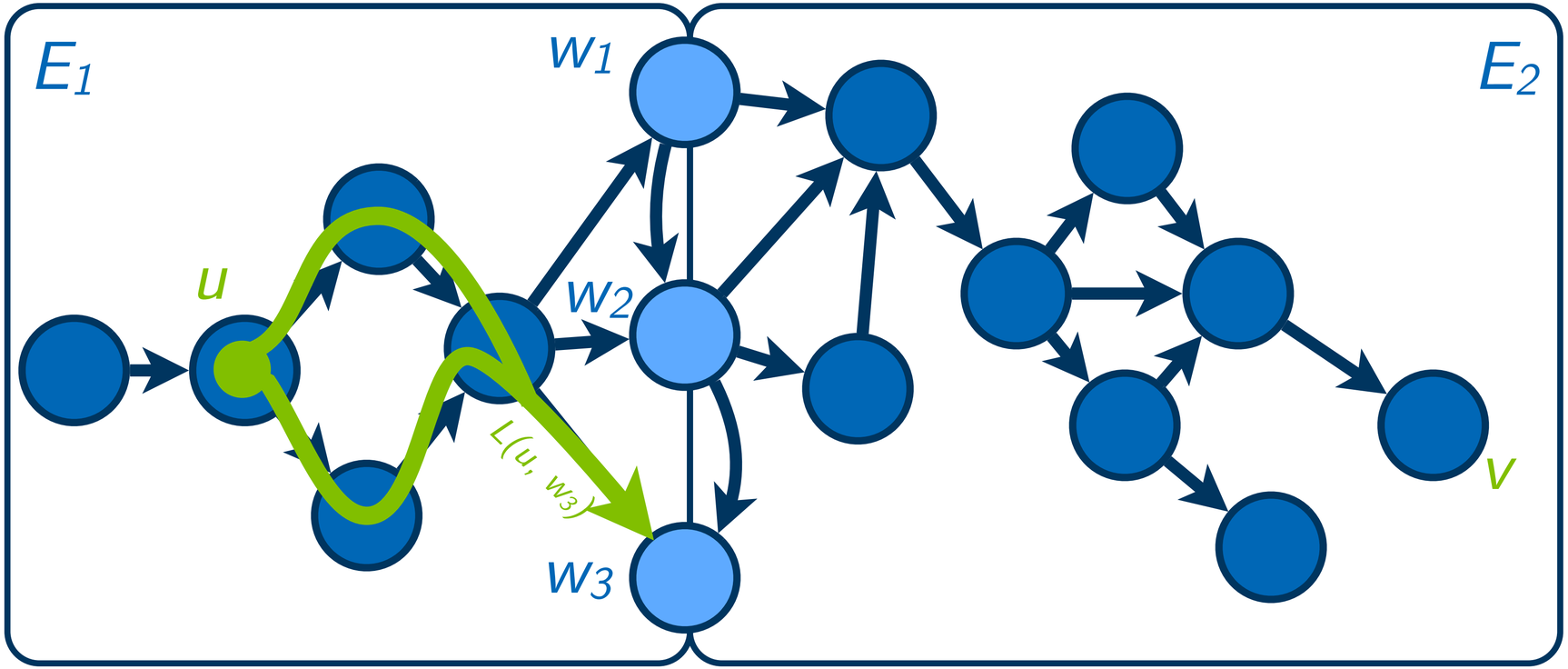
Path families:



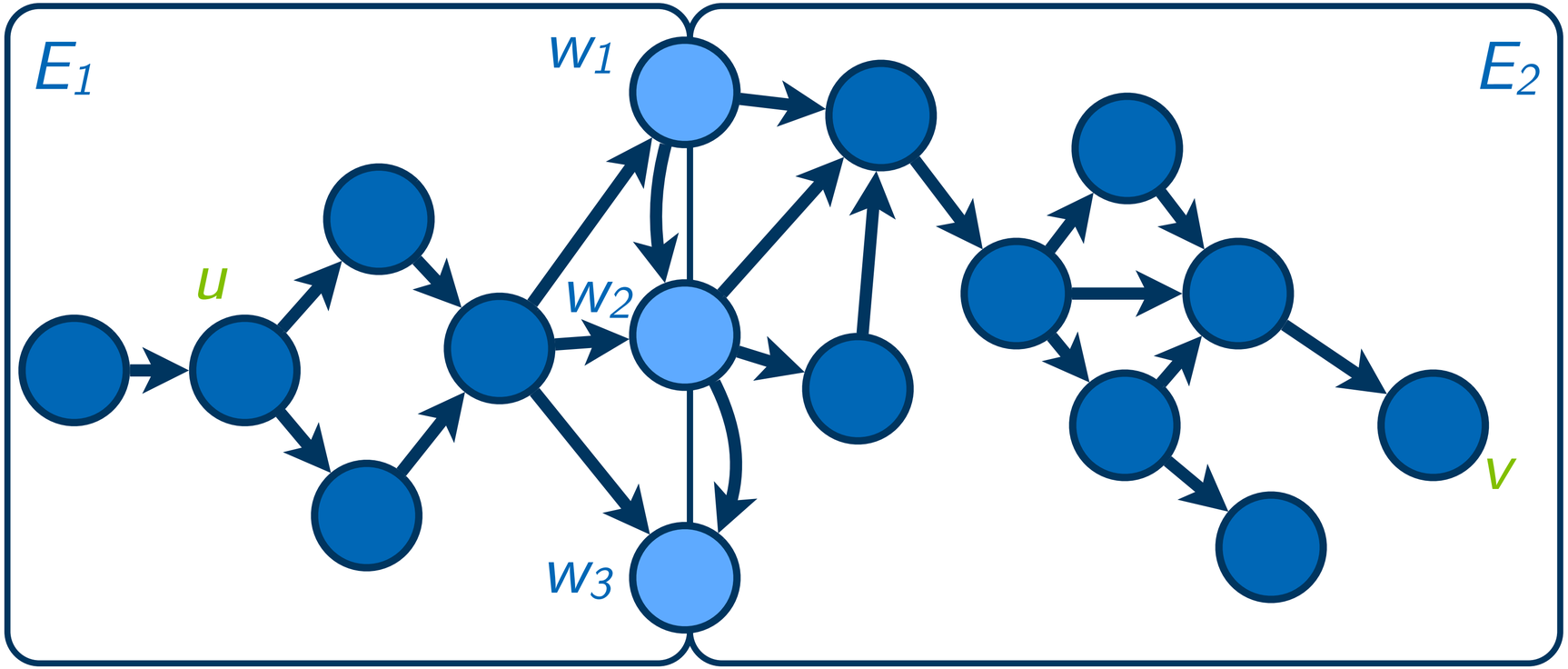
Path families:



Path families:

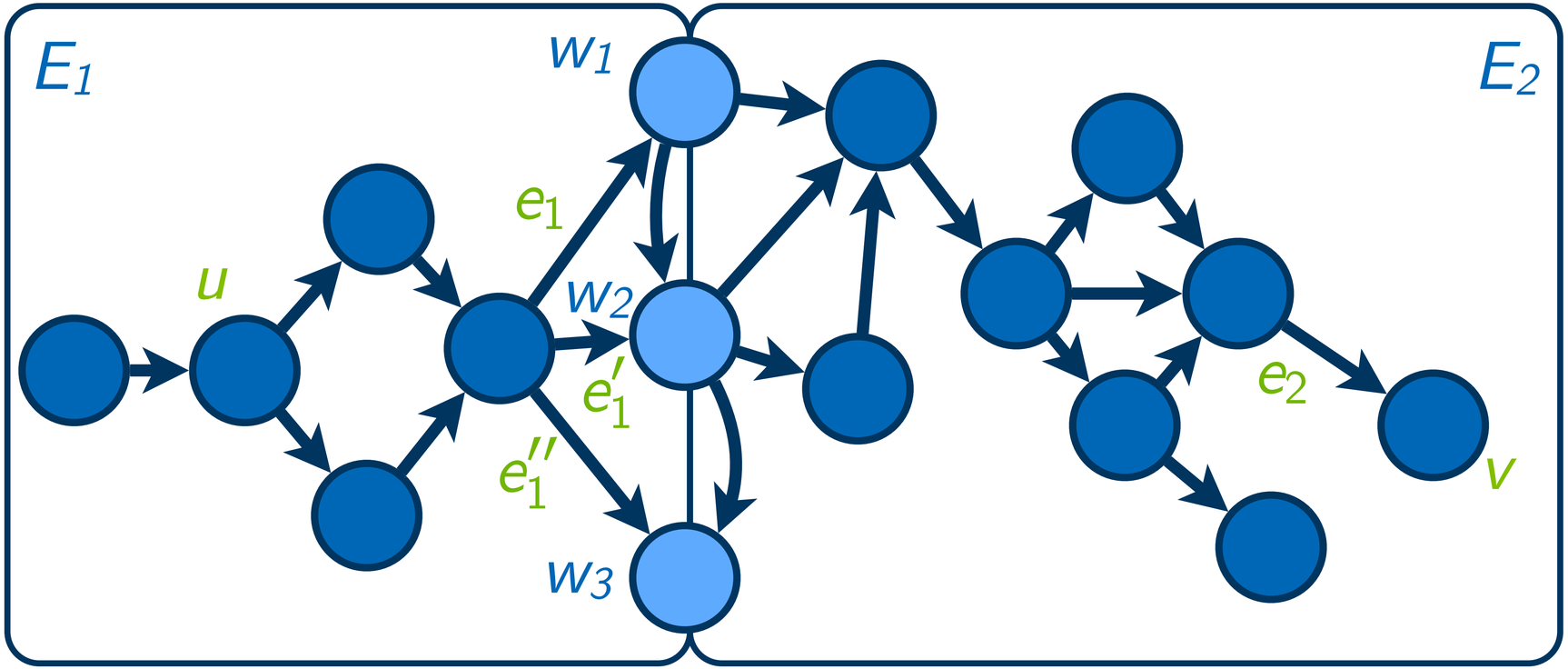


Path families:



Extend DP-cases to: $Any[u, v] = \bigoplus_w (L[u, w] \otimes R[w, v])$

Path families:



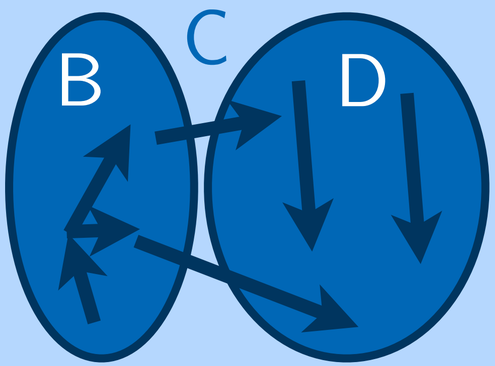
Example:

$$(e_1 \otimes e_2)$$
$$\oplus (e_1' \otimes e_2)$$
$$\oplus (e_1'' \otimes \bot)$$
$$= e_2 \oplus e_2 \oplus \bot$$
$$= e_2$$

Extend DP-cases to: $Any[u, v] = \bigoplus_w (L[u, w] \otimes R[w, v])$

Recursion

vertex split

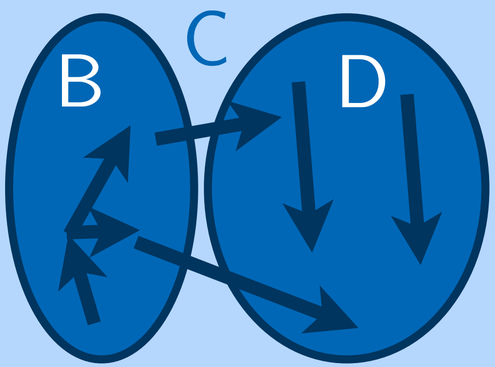


combine C, D

combine B, CD

Recursion

vertex split



combine C, D
combine B, CD

Path Algebra

serial paths

$e \otimes \bot = \bot$

$e \otimes \top = e$

$e \otimes e' = e$ or e'

parallel paths

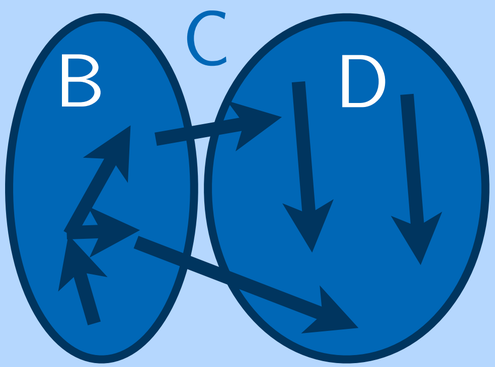
$\bot \oplus e = e$

$\top \oplus e = \top$

$e \oplus e' = \top$

Recursion

vertex split



combine C, D
combine B, CD

Path Algebra

serial paths

$e \otimes \bot = \bot$
$e \otimes \top = e$
$e \otimes e' = e$ or e'

parallel paths

$\bot \oplus e = e$
$\top \oplus e = \top$
$e \oplus e' = \top$

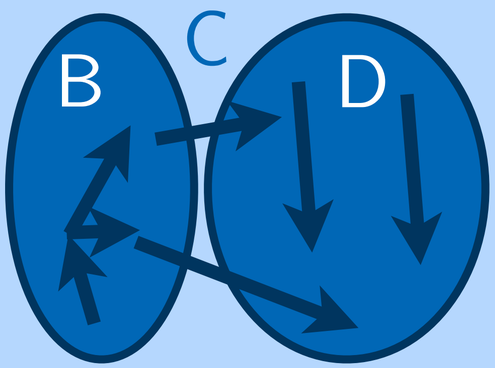
Bit representations

simulate $\otimes, \oplus$ with $\wedge, \vee$

$\{\bot, \top\} \cup E \to \{0,1\}^{\Theta(\log n)}$

Recursion

vertex split



combine C, D
combine B, CD

Path Algebra

serial paths

$e \otimes \bot = \bot$
$e \otimes \top = e$
$e \otimes e' = e$ or e'

parallel paths

$\bot \oplus e = e$
$\top \oplus e = \top$
$e \oplus e' = \top$

Bit representations

simulate $\otimes, \oplus$ with $\wedge, \vee$

$\{\bot, \top\} \cup E \to \{0,1\}^{\Theta(\log n)}$

$\bot \to 0 \ldots 0 \mid 0 \ldots 0$

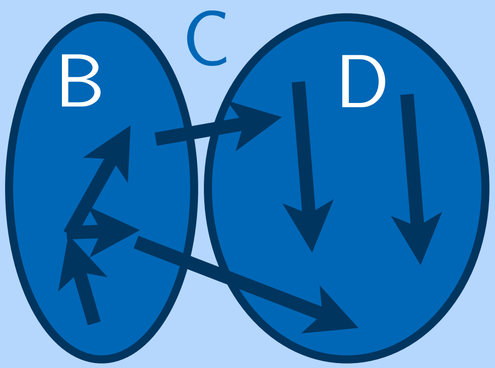
$\top \to 1 \ldots 1 \mid 1 \ldots 1$

$e_1 \to b_{e_1} \overline{b_{e_1}} \mid 1 \ldots 1$

$e_2 \to 1 \ldots 1 \mid b_{e_2} \overline{b_{e_2}}$

Recursion

vertex split



combine C, D
combine B, CD

Path Algebra

serial paths

$e \otimes \bot = \bot$
$e \otimes \top = e$
$e \otimes e' = e$ or e'

parallel paths

$\bot \oplus e = e$
$\top \oplus e = \top$
$e \oplus e' = \top$

Bit representations

simulate $\otimes, \oplus$ with $\wedge, \vee$

$\{\bot, \top\} \cup E \to \{0,1\}^{\Theta(\log n)}$

$\bot \to 0 \ldots 0 \mid 0 \ldots 0$

$\top \to 1 \ldots 1 \mid 1 \ldots 1$

$e_1 \to b_{e_1} \overline{b_{e_1}} \mid 1 \ldots 1$

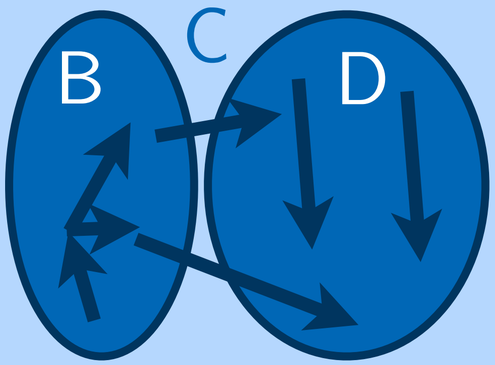
$e_2 \to 1 \ldots 1 \mid b_{e_2} \overline{b_{e_2}}$

allows 0, 1, 2+ counting

DAGs / Divide & Conquer / Details / ETH zürich

Recursion

vertex split



combine C, D
combine B, CD

Path Algebra

serial paths

$e \otimes \bot = \bot$
$e \otimes \top = e$
$e \otimes e' = e$ or e'

parallel paths

$\bot \oplus e = e$
$\top \oplus e = \top$
$e \oplus e' = \top$

Bit representations

simulate $\otimes, \oplus$ with $\wedge, \vee$

$\{\bot, \top\} \cup E \to \{0,1\}^{\Theta(\log n)}$

$\bot \to 0 \ldots 0 \mid 0 \ldots 0$

$\top \to 1 \ldots 1 \mid 1 \ldots 1$

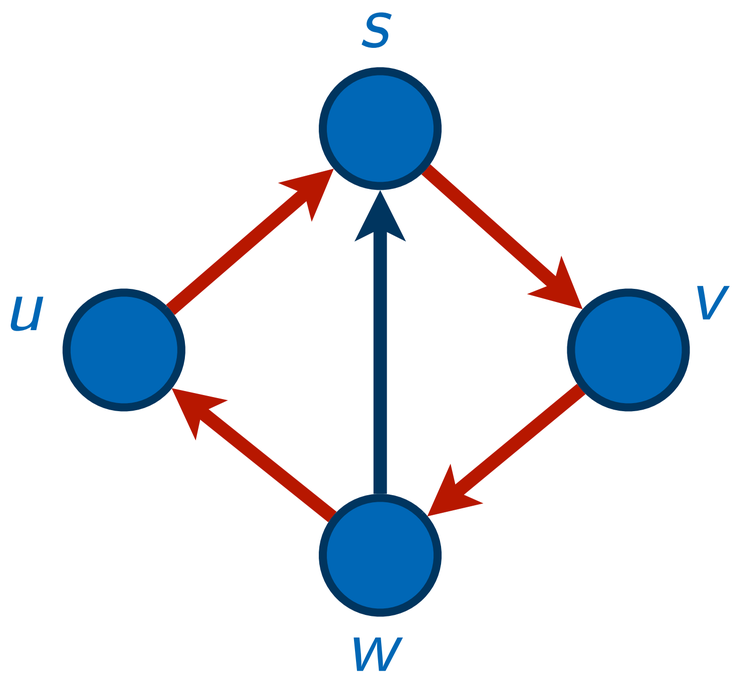
$e_1 \to b_{e_1} \overline{b_{e_1}} \mid 1 \ldots 1$

$e_2 \to 1 \ldots 1 \mid b_{e_2} \overline{b_{e_2}}$

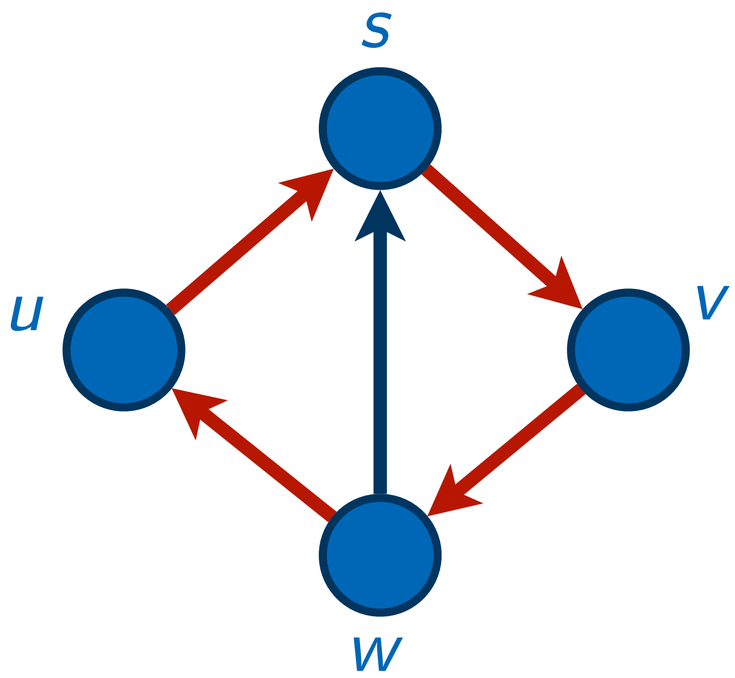
allows 0, 1, 2+ counting

overall:
$\tilde{\mathcal{O}}(n^\omega)$

Strong bridge: edge whose removal increases number of SCCs



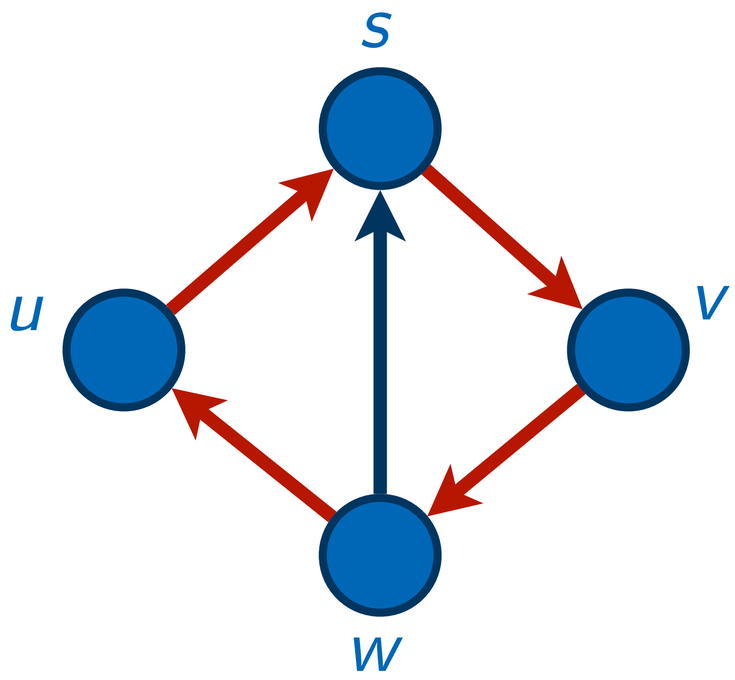
Strong bridge: edge whose removal increases number of SCCs



How to quickly find them all?

break the symmetry: „root“ G at s

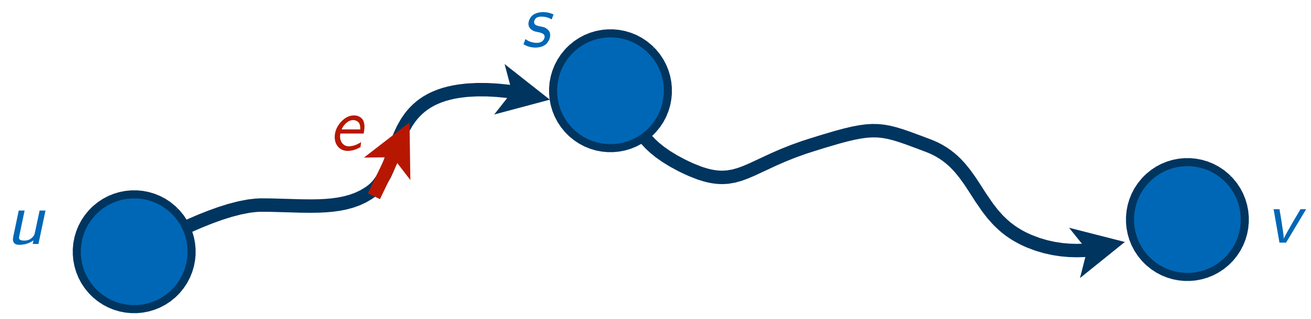
Strong bridge: edge whose removal increases number of SCCs



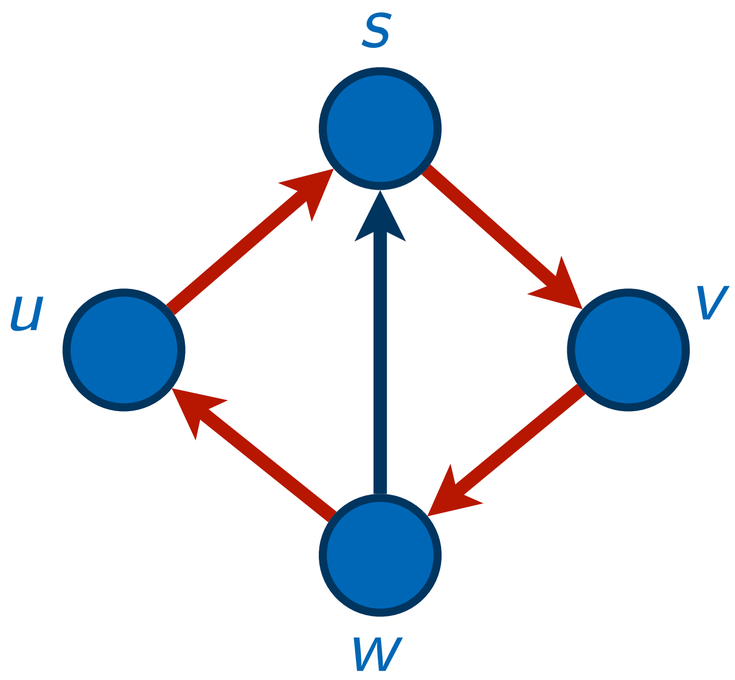
How to quickly find them all?

break the symmetry: „root“ G at s

Observe: any strong bridge e disconnects some u from s or s from some v



Strong bridge: edge whose removal increases number of SCCs

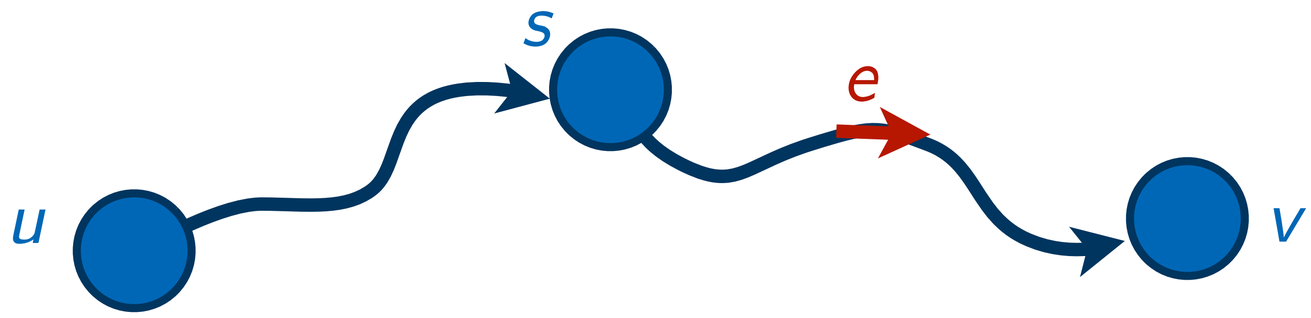


How to quickly find them all?

break the symmetry: „root“ G at s

Observe: any strong bridge e disconnects

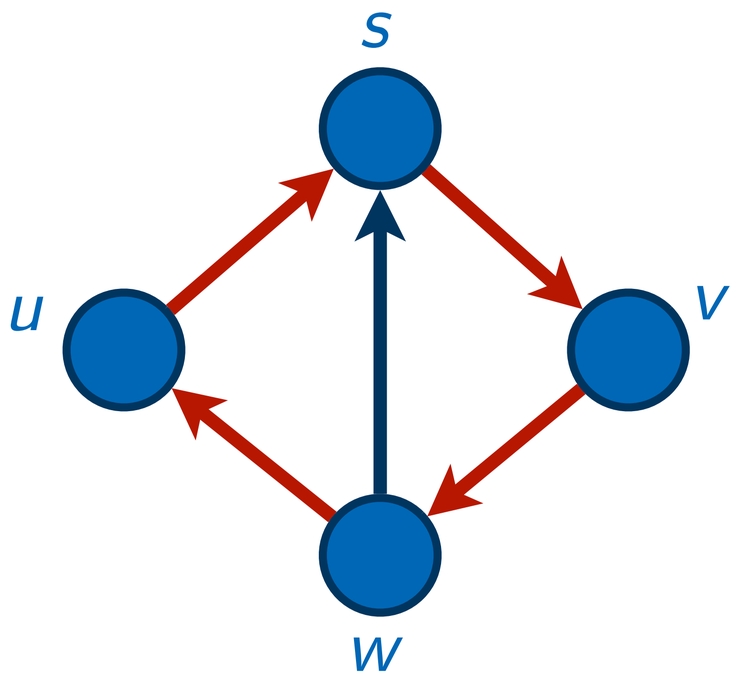
some u from s or s from some v



Strongly Connected / Strong Bridges

ETH zürich

Strong bridge: edge whose removal increases number of SCCs

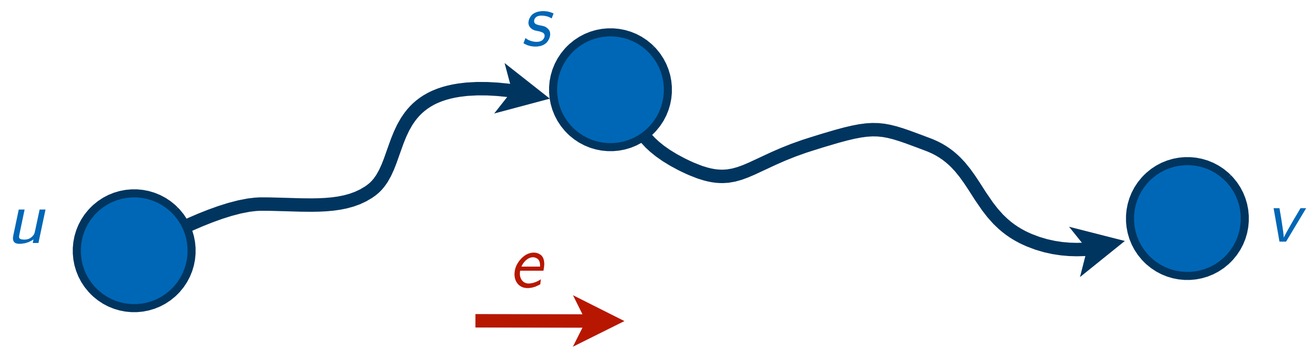


How to quickly find them all?

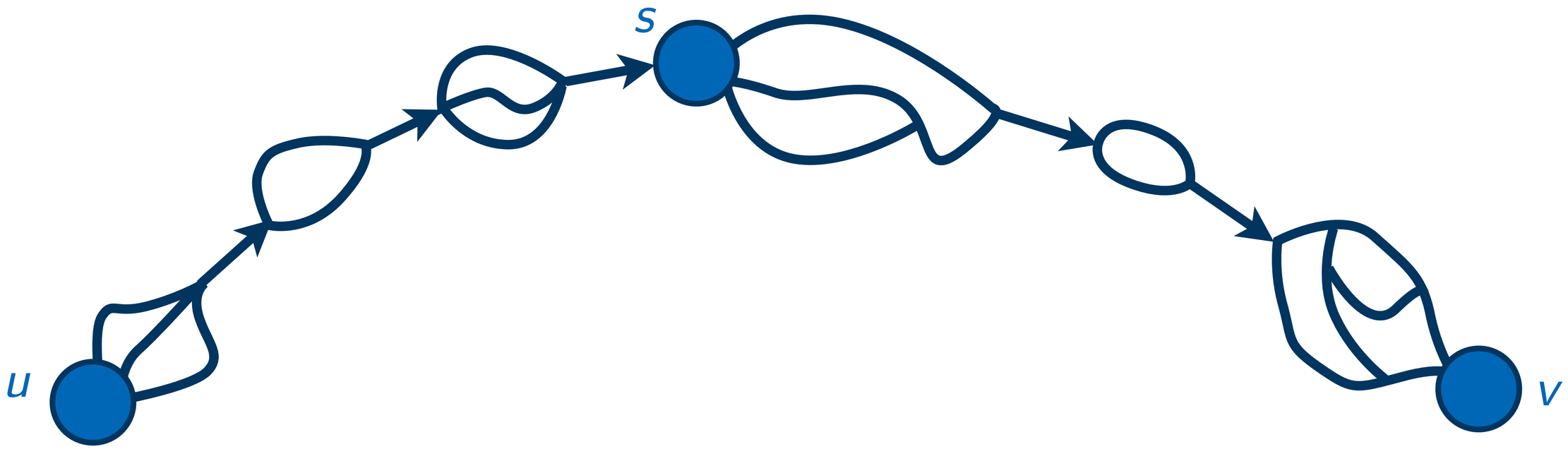
break the symmetry: „root“ G at s

Observe: any strong bridge e disconnects

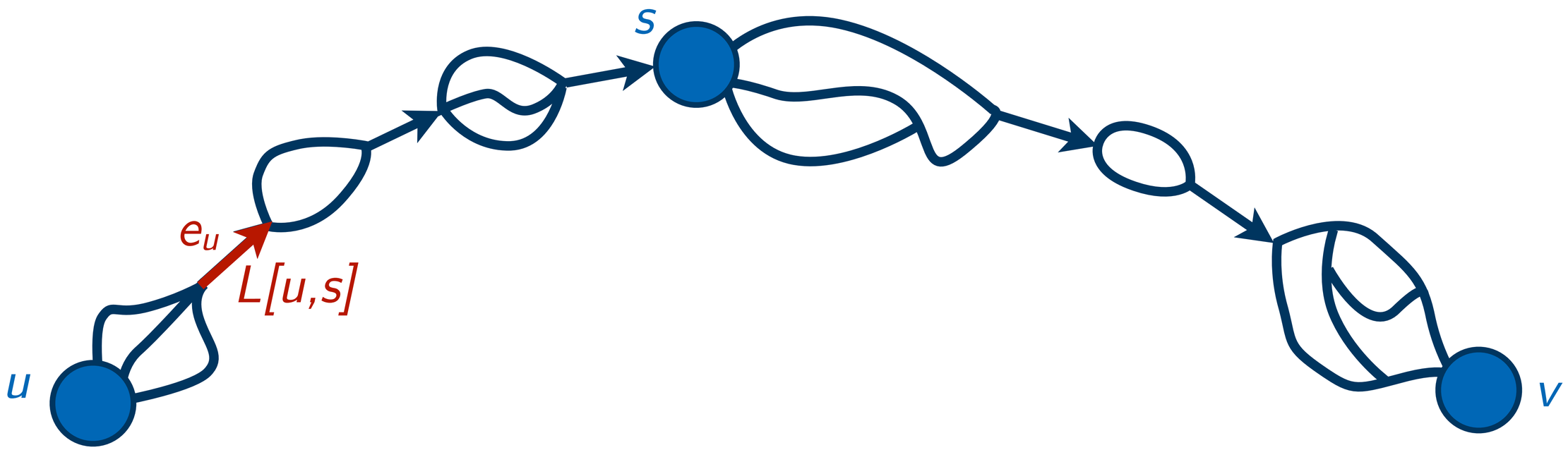
some u from s or s from some v



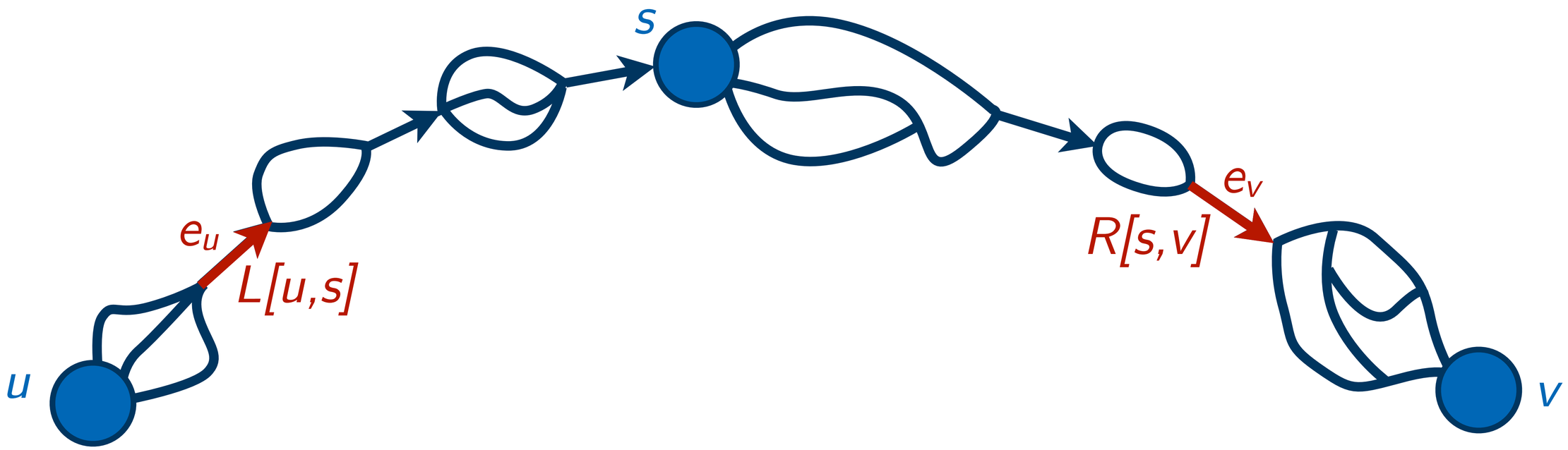
Testing 2-reachability after breaking symmetry



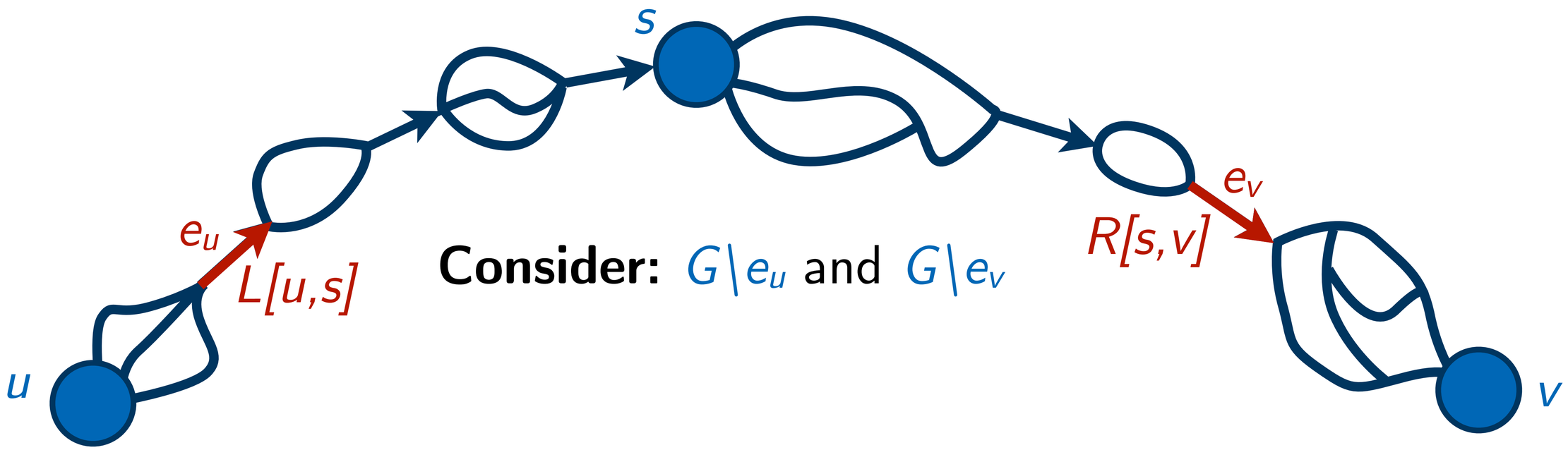
Testing 2-reachability after breaking symmetry



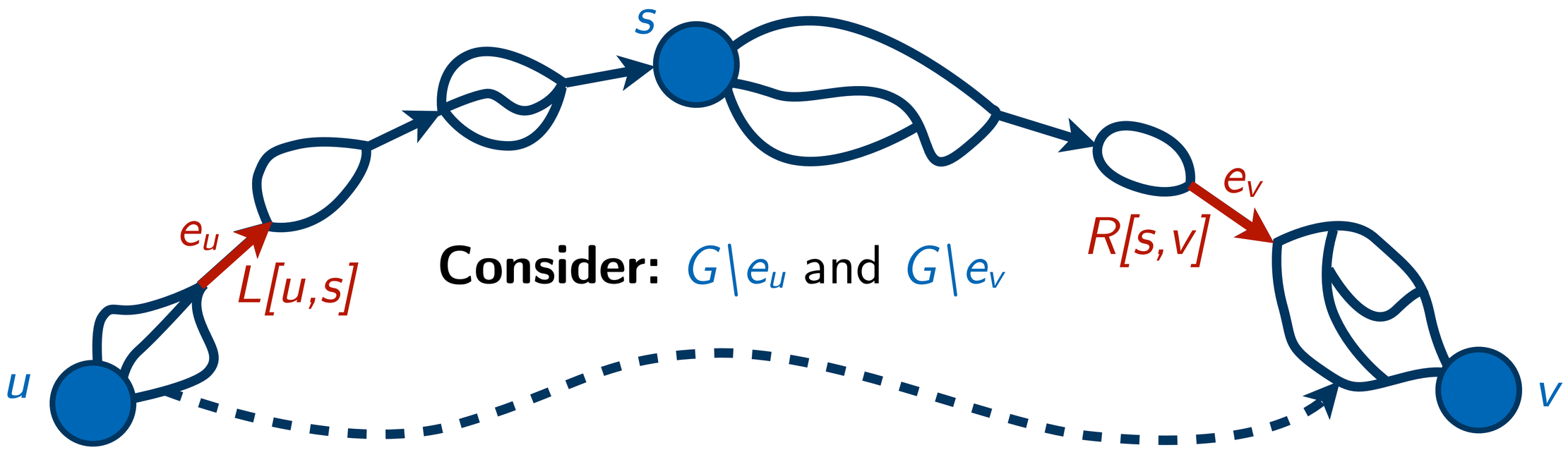
Testing 2-reachability after breaking symmetry



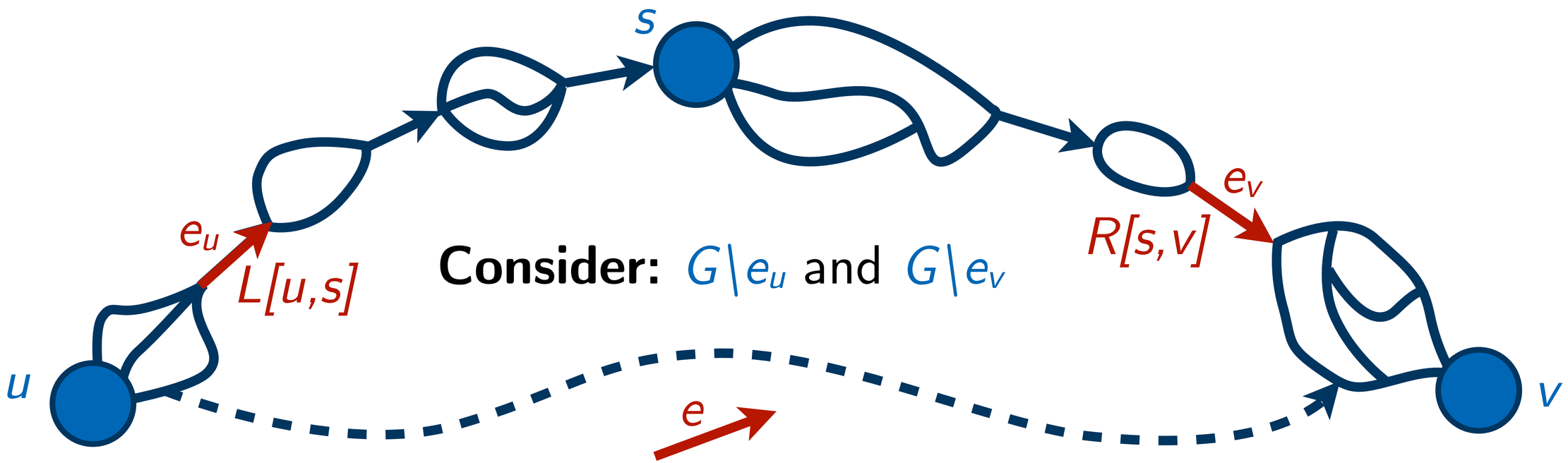
Testing 2-reachability after breaking symmetry



Testing 2-reachability after breaking symmetry



Testing 2-reachability after breaking symmetry



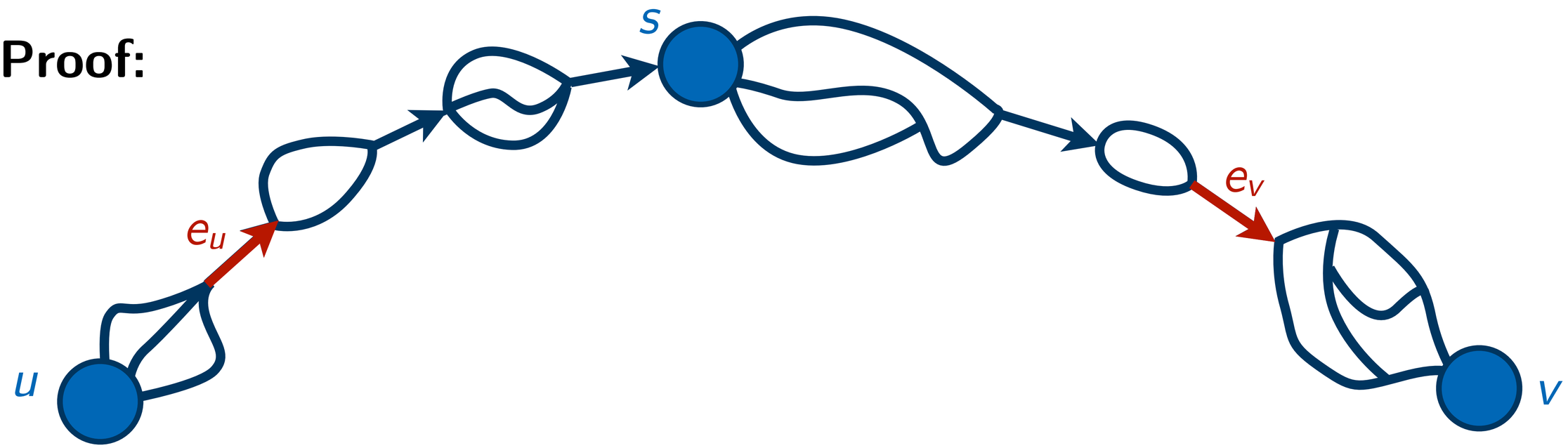
Claim	If some e disconnects u from v, then so does e_u or e_v.

Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:

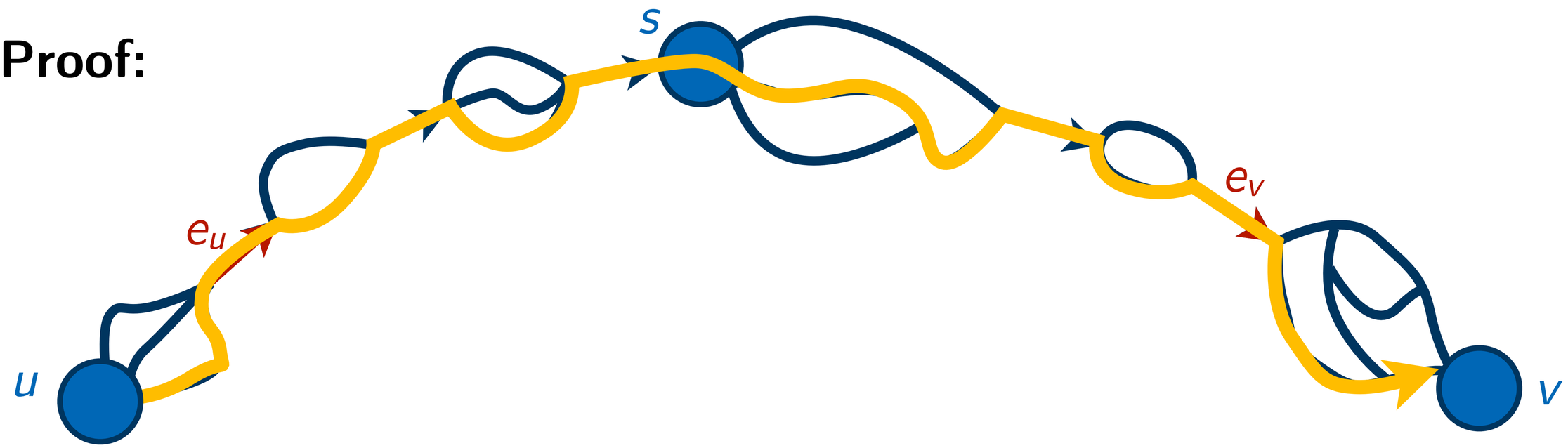


Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:

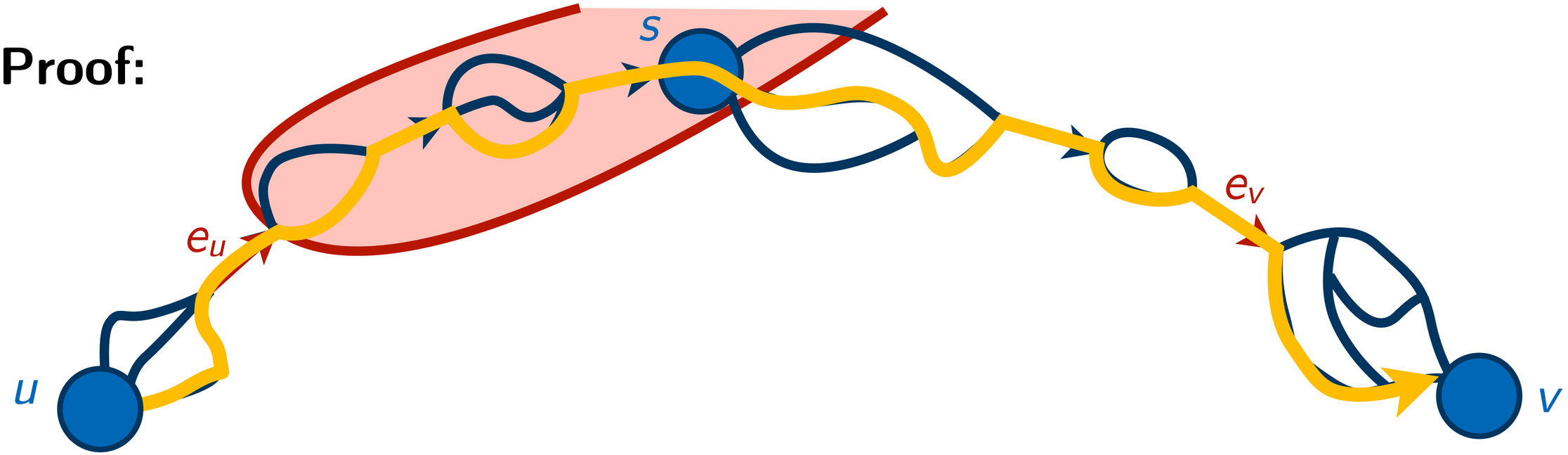


Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:

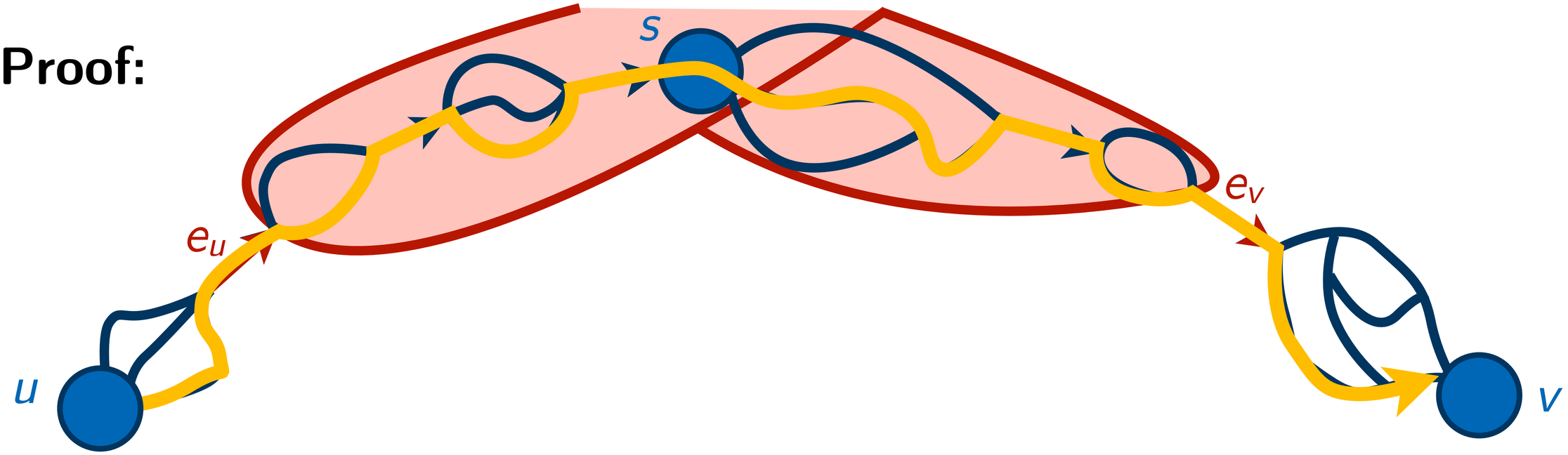


Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:

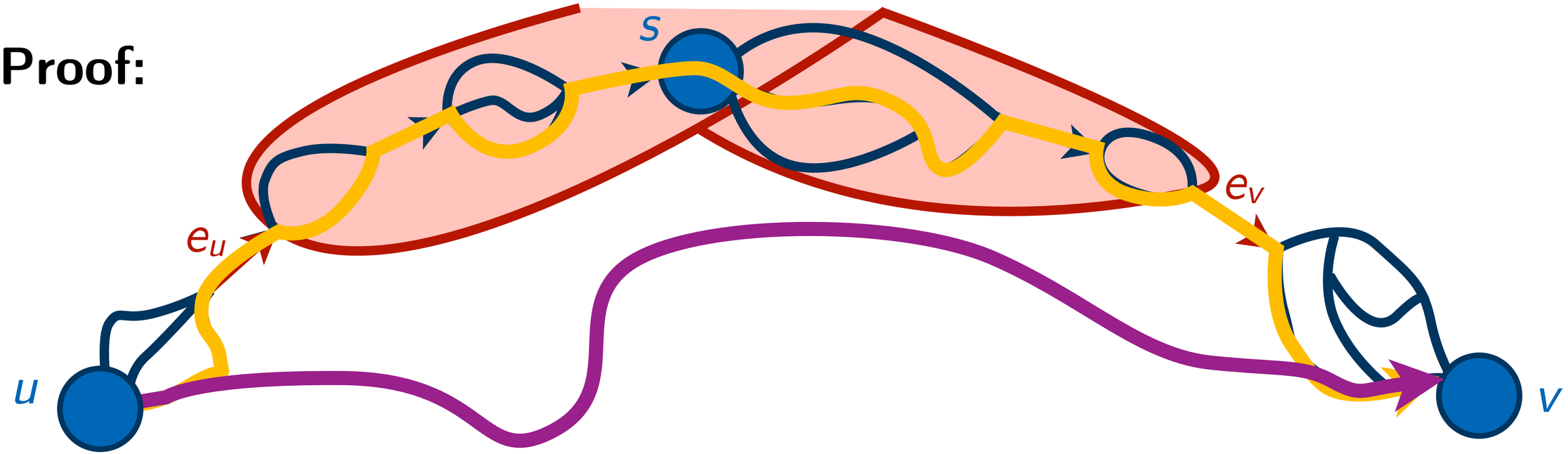


Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:



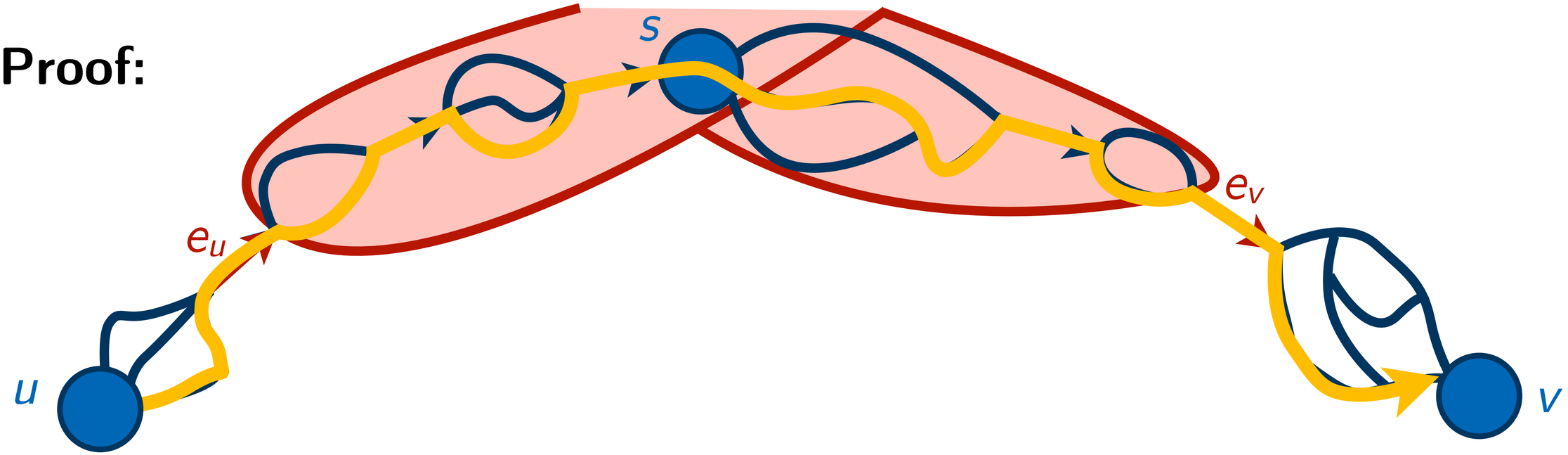
Case 1: path in $G \setminus \{e_u, e_v\} \Rightarrow$ 2 u-v-paths

Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:



Case 1: path in $G \setminus \{e_u, e_v\} \Rightarrow$ 2 u-v-paths

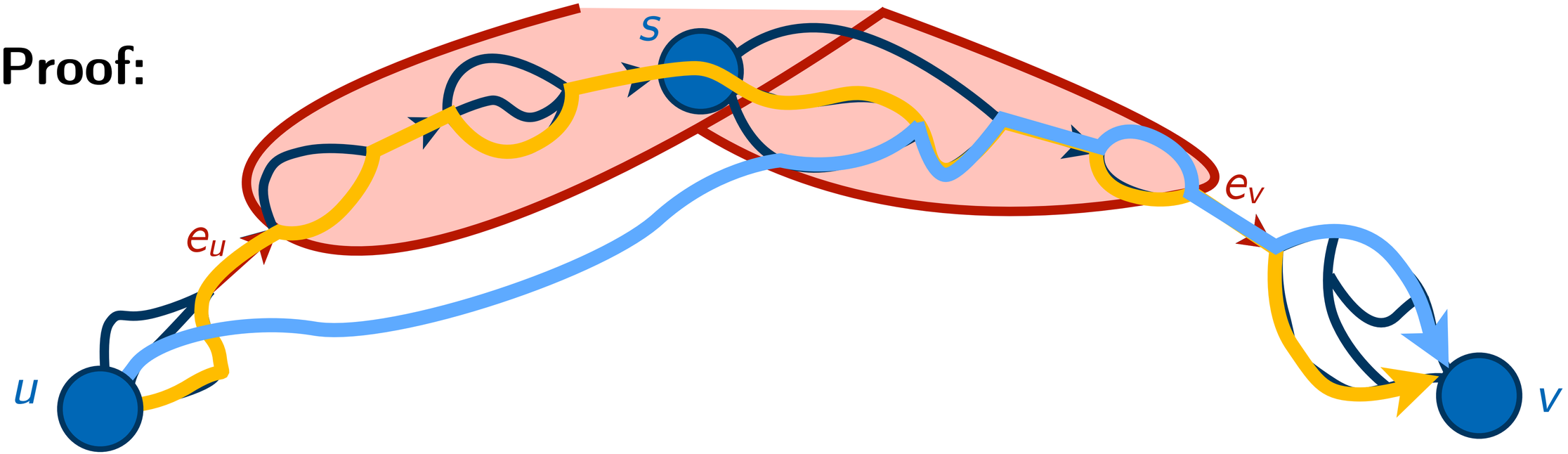
Case 2: path in $G \setminus e_u$ and path in $G \setminus e_v \Rightarrow$ 2 u-v-paths

Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:



Case 1: path in $G \setminus \{e_u, e_v\} \Rightarrow$ 2 u-v-paths

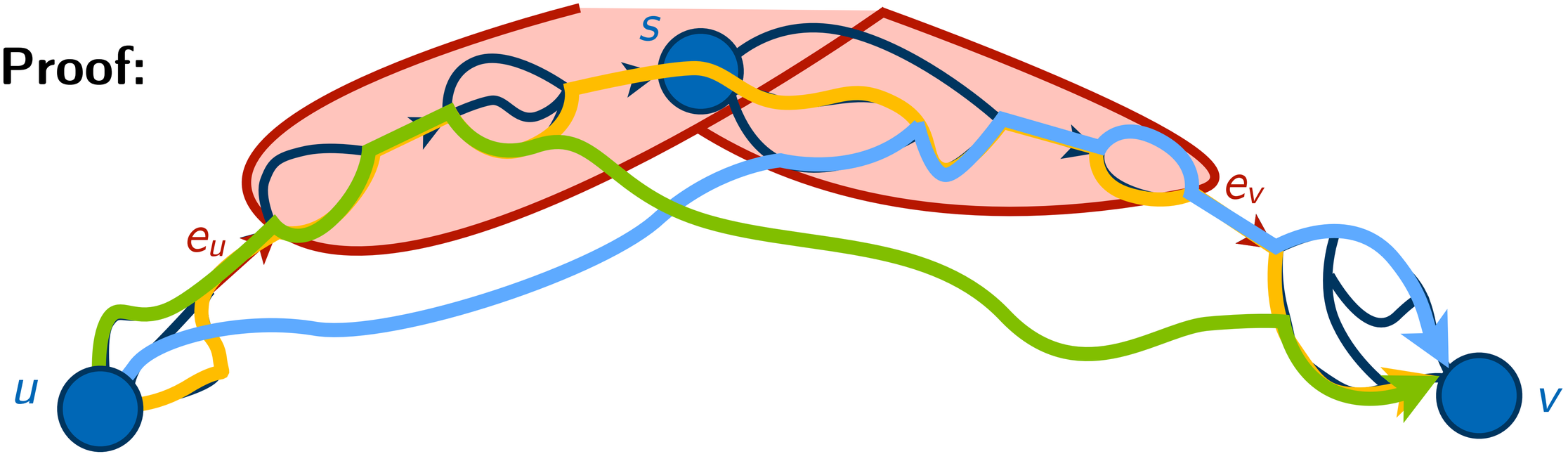
Case 2: path in $G \setminus e_u$ and path in $G \setminus e_v \Rightarrow$ 2 u-v-paths

Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:



Case 1: path in $G \setminus \{e_u, e_v\} \Rightarrow$ 2 u-v-paths

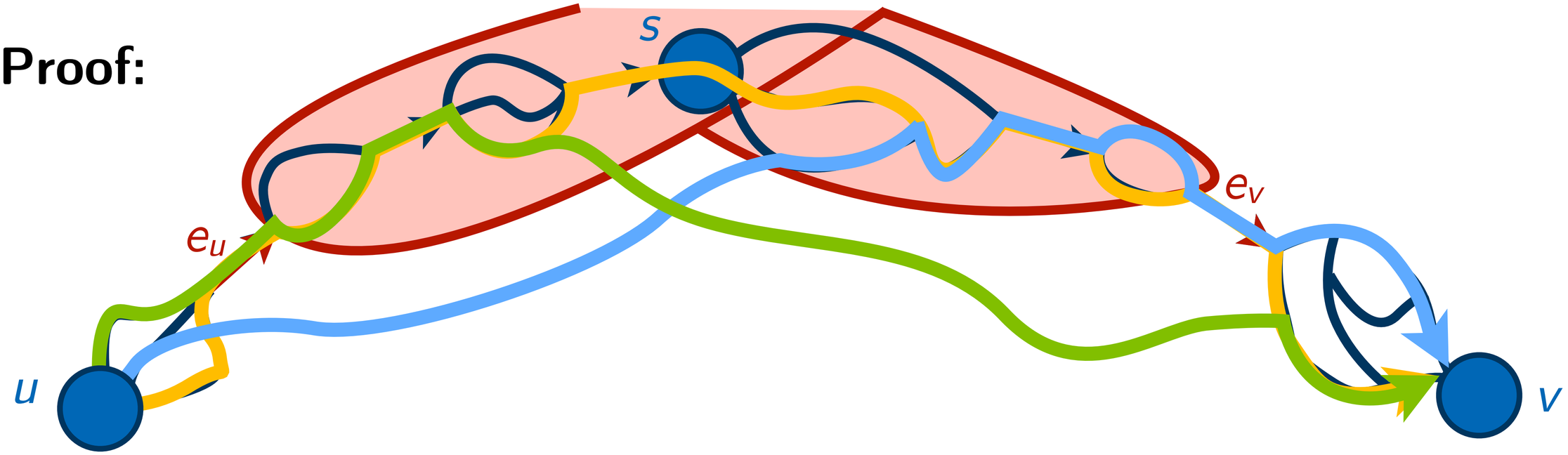
Case 2: path in $G \setminus e_u$ and path in $G \setminus e_v \Rightarrow$ 2 u-v-paths

Strongly Connected / Asymmetric

ETH zürich

Claim u-v-path in both $G \setminus e_u$ and $G \setminus e_v \Rightarrow$ multiple u-v-paths in G

Proof:



Case 1: path in $G \setminus \{e_u, e_v\} \Rightarrow$ 2 u-v-paths

Case 2: path in $G \setminus e_u$ and path in $G \setminus e_v \Rightarrow$ 2 u-v-paths

Strongly Connected / Auxiliary Graphs / ETH zürich

Query

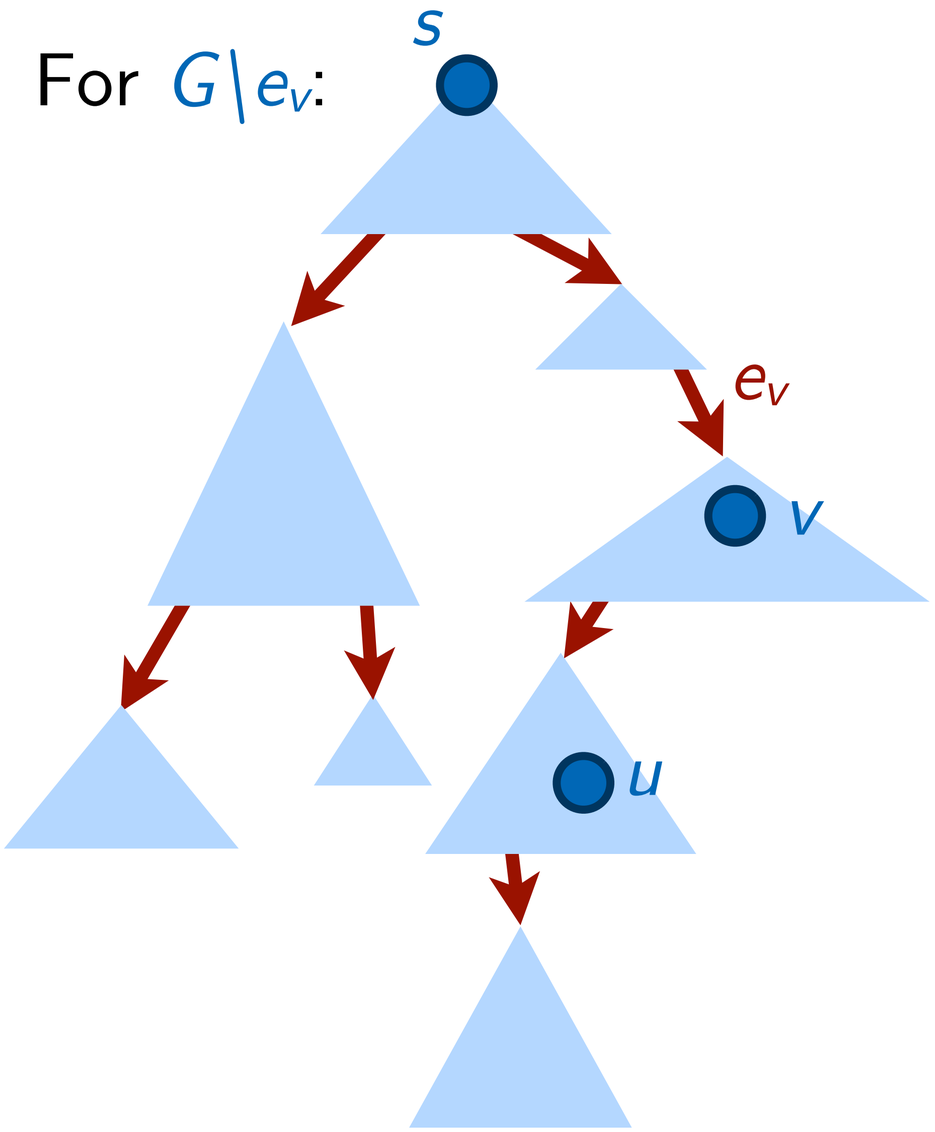
u-v-path in $G \setminus e_u$ and

u-v-path in $G \setminus e_v$?

Strongly Connected / Auxiliary Graphs

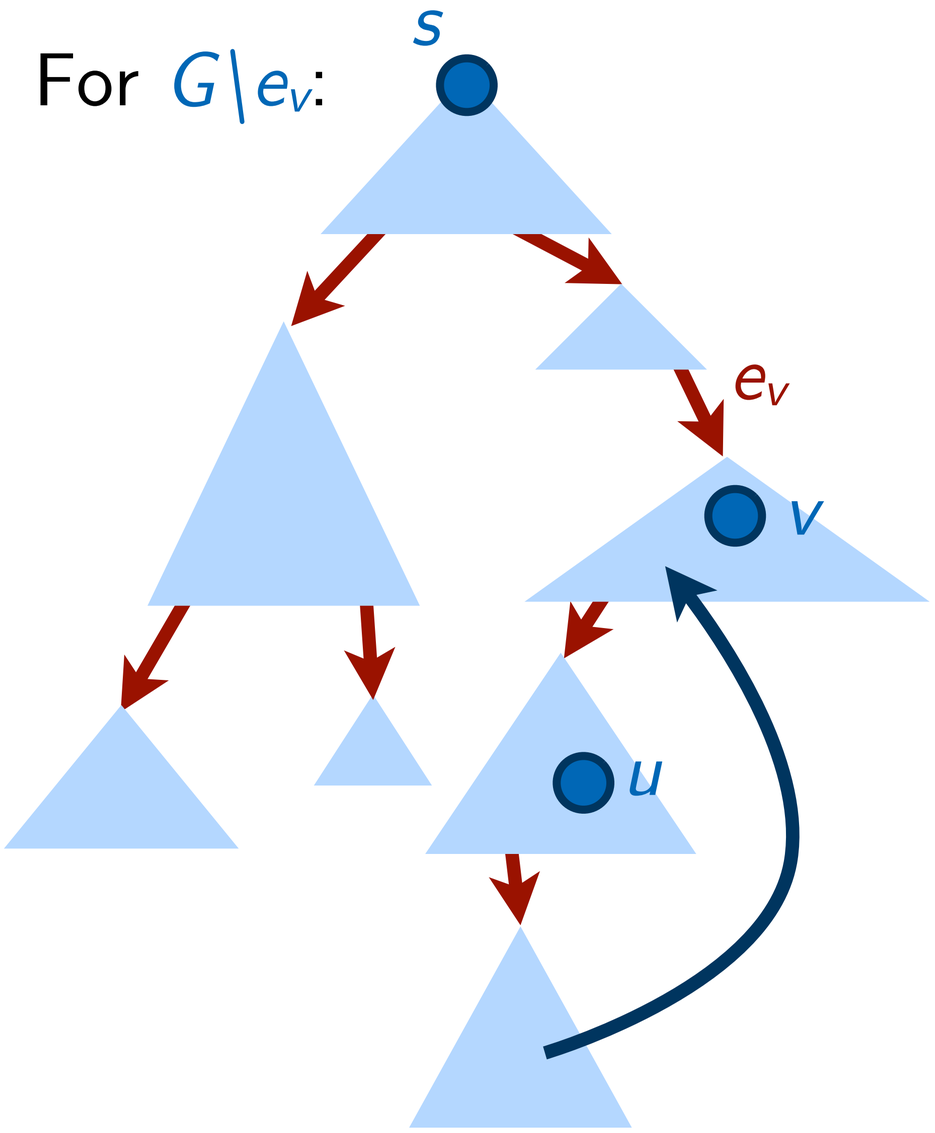
ETH zürich

Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?



Strongly Connected / Auxiliary Graphs / ETH zürich

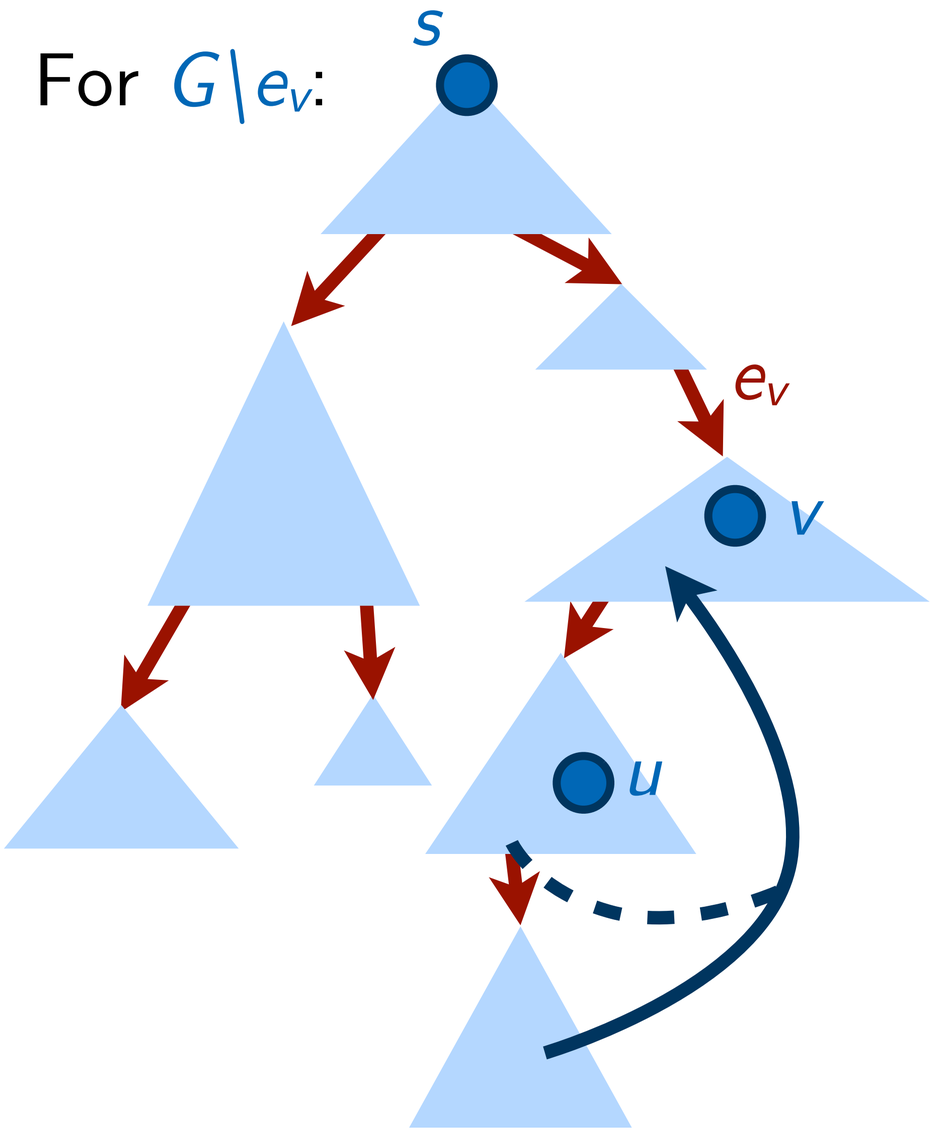
Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?



Strongly Connected / Auxiliary Graphs

ETH zürich

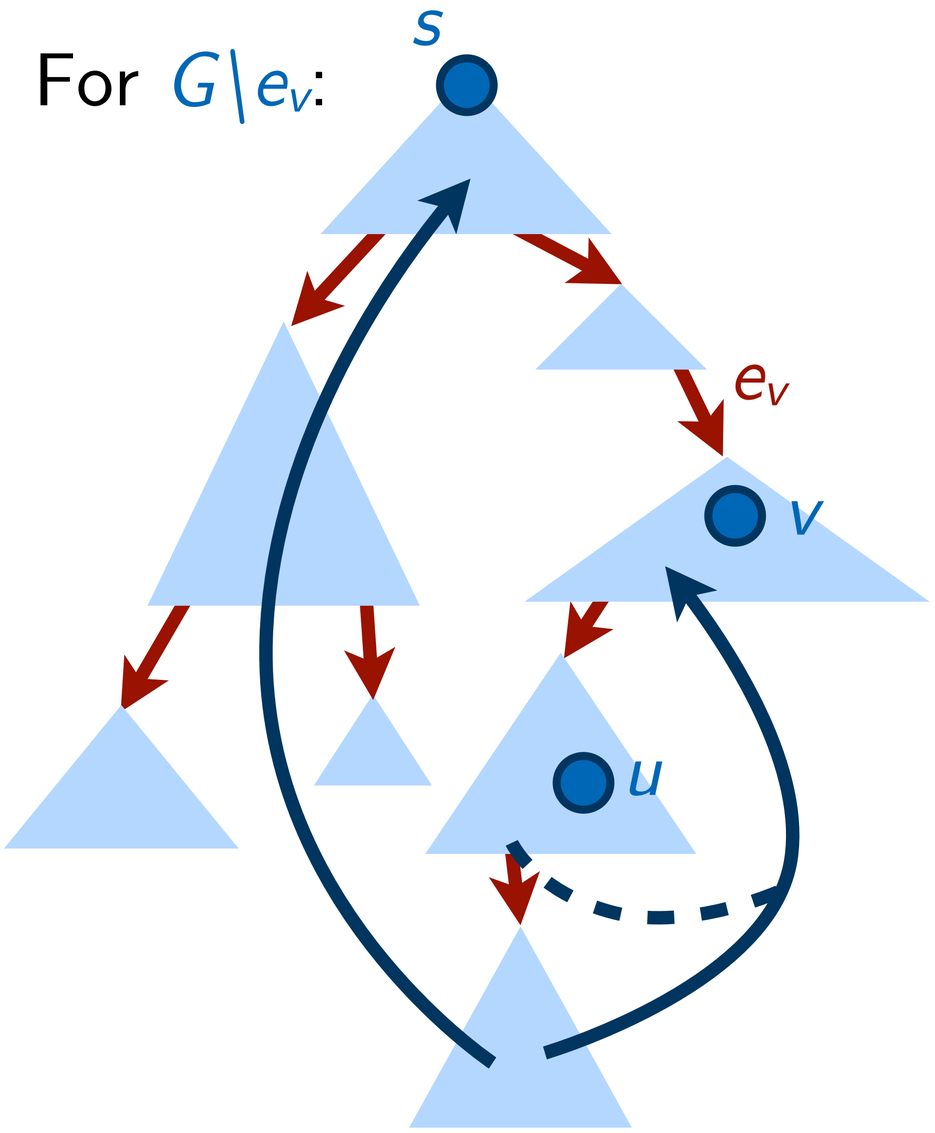
Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?



Strongly Connected / Auxiliary Graphs

ETH zürich

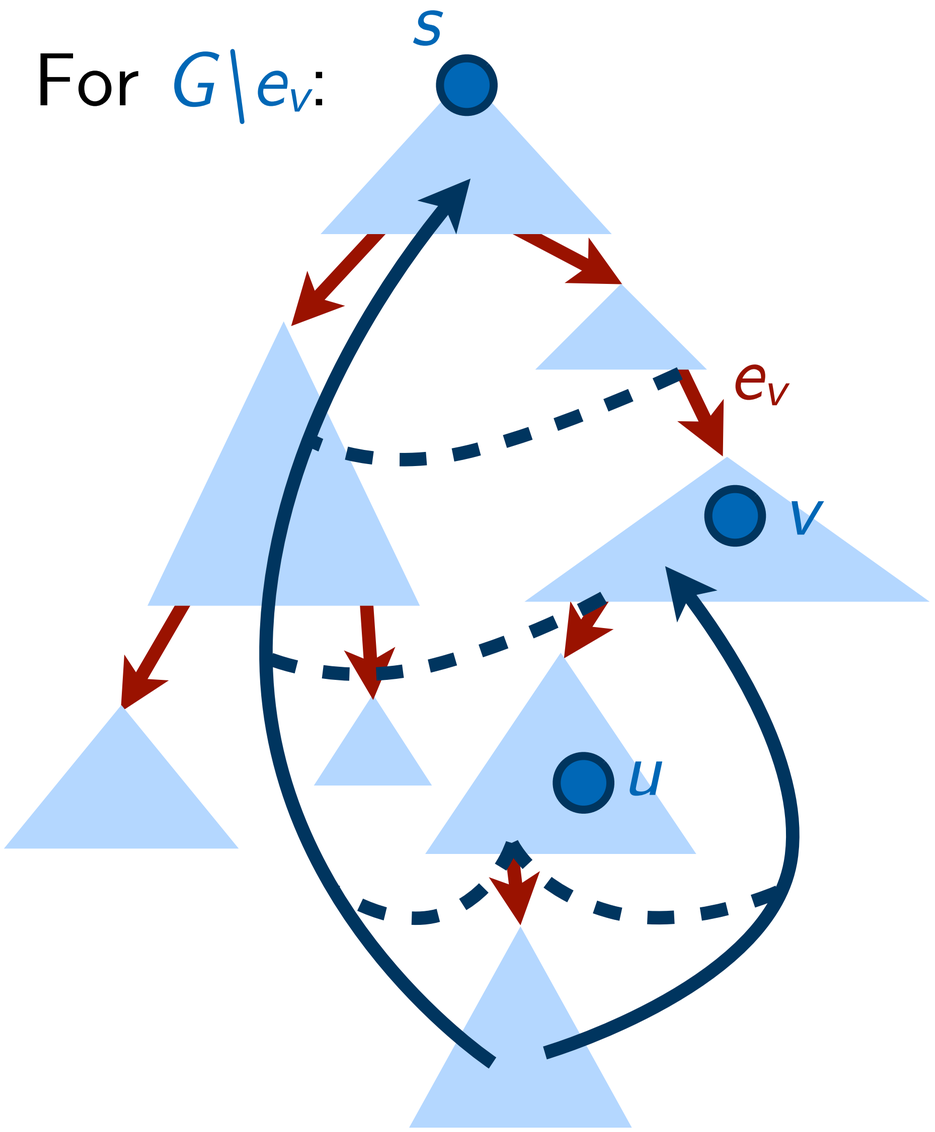
Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?



Strongly Connected Auxiliary Graphs

ETH zürich

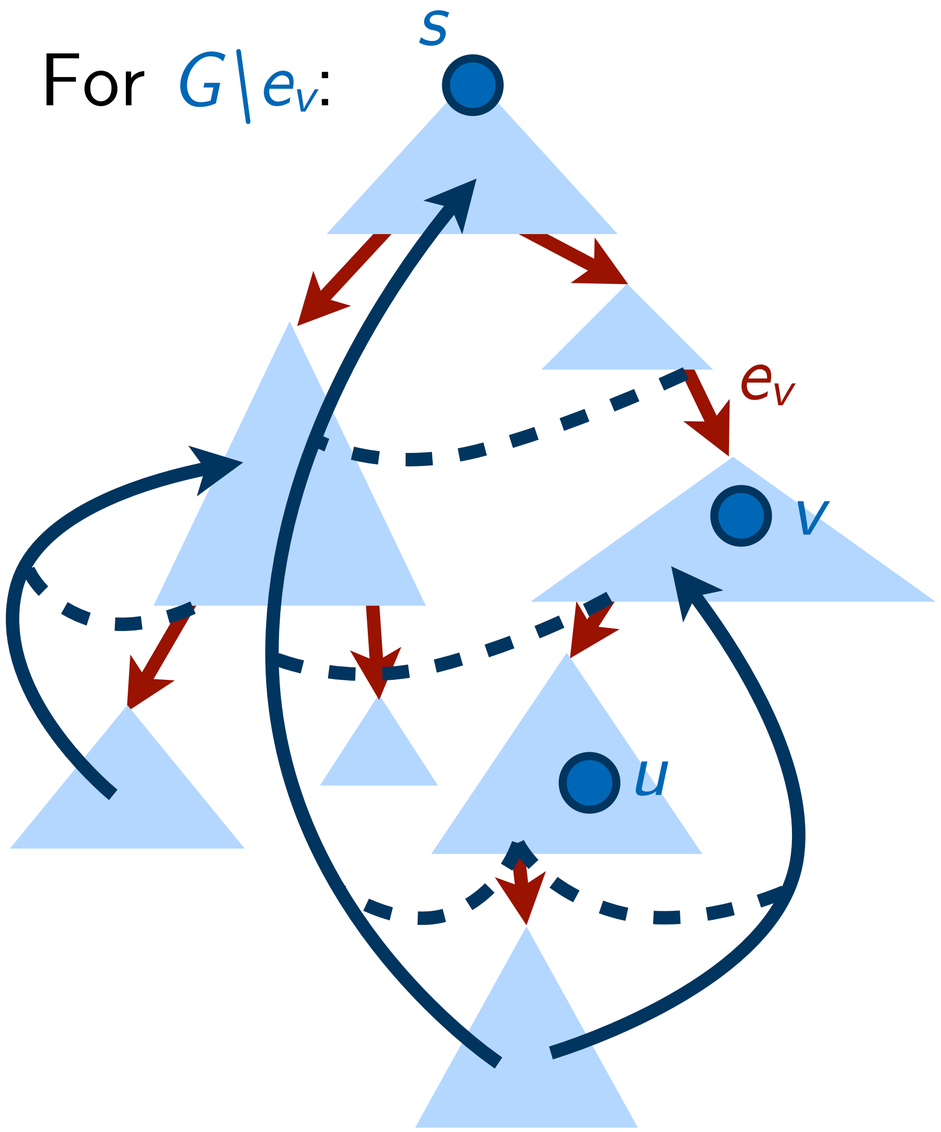
Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?



Strongly Connected Auxiliary Graphs

ETH zürich

Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?

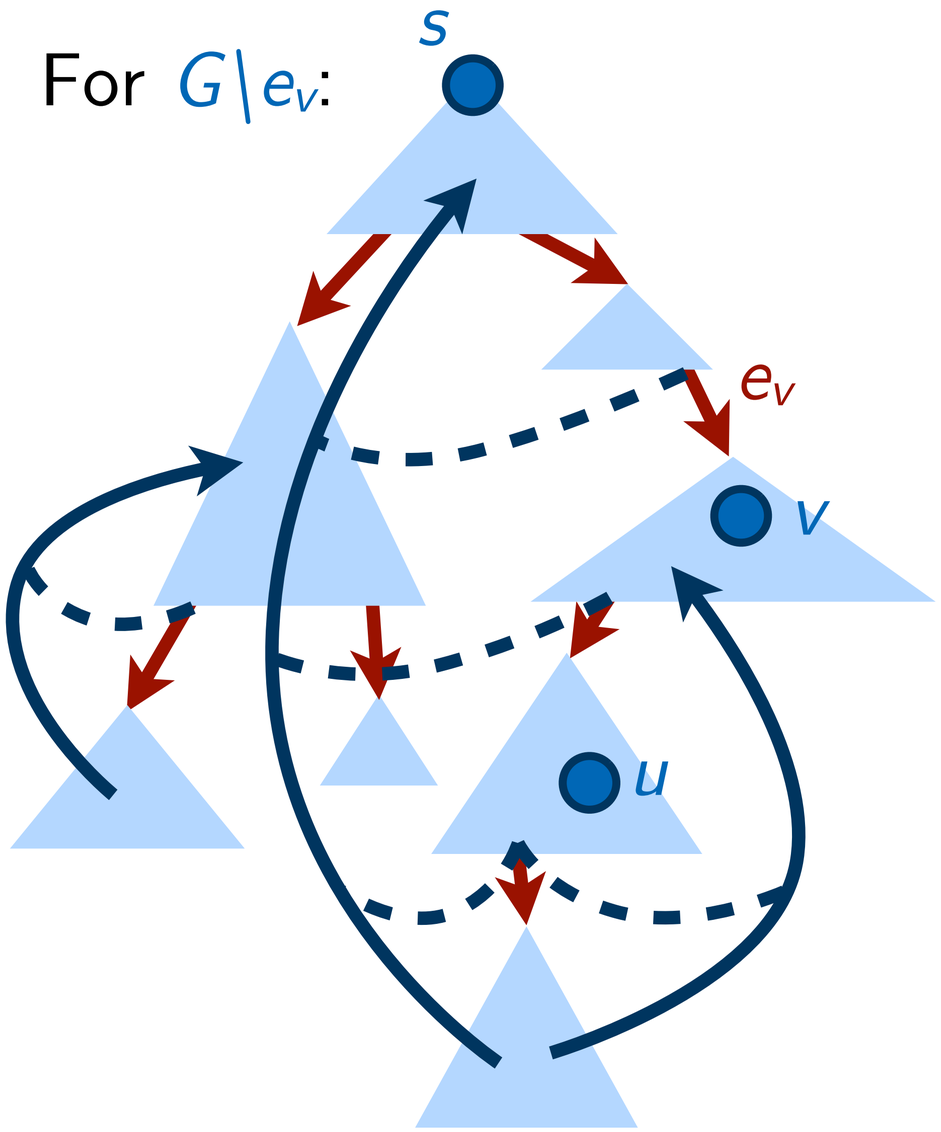


Strongly Connected / Auxiliary Graphs / ETH zürich

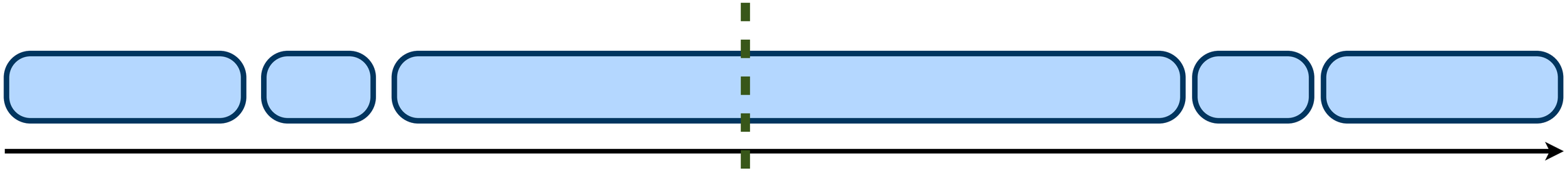
Query	u-v-path in $G \setminus e_u$ and u-v-path in $G \setminus e_v$?

Algorithm:

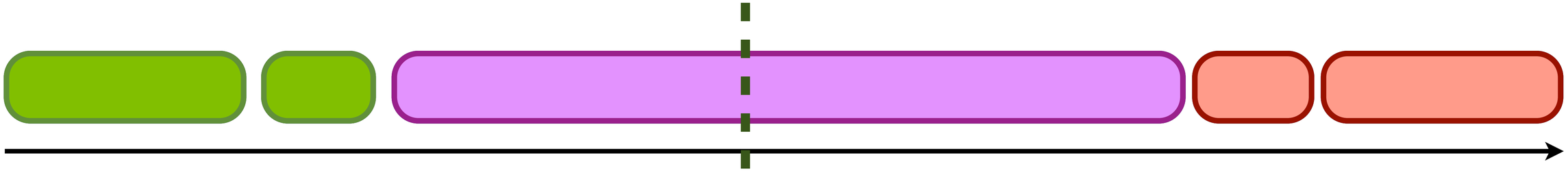
- build two auxiliary graphs in $\mathcal{O}(n^2)$
- take transitive closure in $\mathcal{O}(n^\omega)$
- answer queries in $\mathcal{O}(1)$



First: compute SCCs and any topological order in $\mathcal{O}(n+m)$

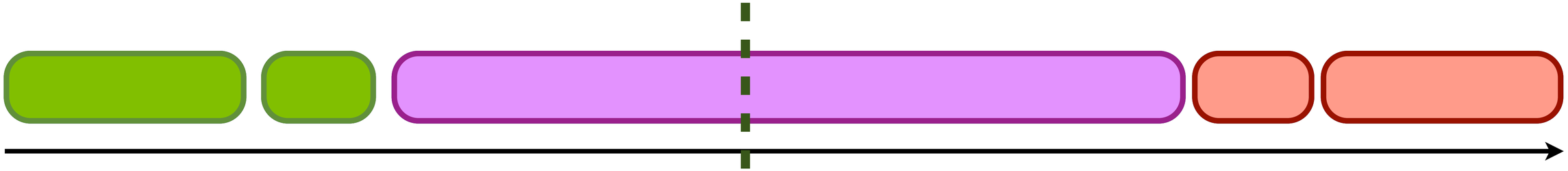


First: compute SCCs and any topological order in $\mathcal{O}(n+m)$



Is there a giant in the middle?

First: compute SCCs and any topological order in $\mathcal{O}(n+m)$

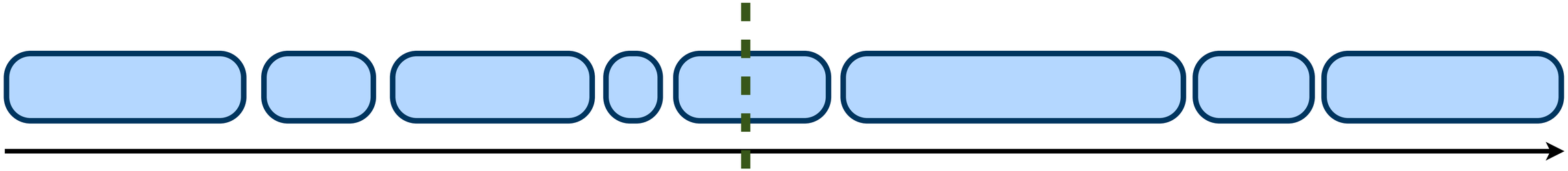


Is there a giant in the middle?

- Yes: $T(n) \leq T(n/3) + T(n/3) + \mathcal{O}(n^\omega) + \mathcal{O}(n^\omega \log n)$

recursive parts | giant SCC | merge edge-cuts

First: compute SCCs and any topological order in $\mathcal{O}(n+m)$

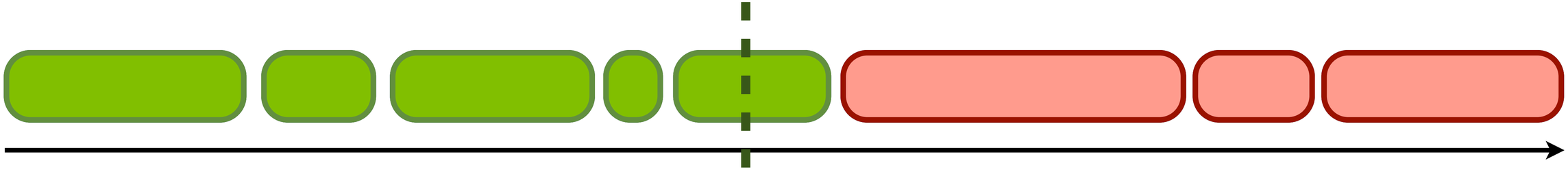


Is there a giant in the middle?

- Yes: $T(n) \leq T(n/3) + T(n/3) + \mathcal{O}(n^\omega) + \mathcal{O}(n^\omega \log n)$

recursive parts | giant SCC | merge edge-cuts

First: compute SCCs and any topological order in $\mathcal{O}(n+m)$

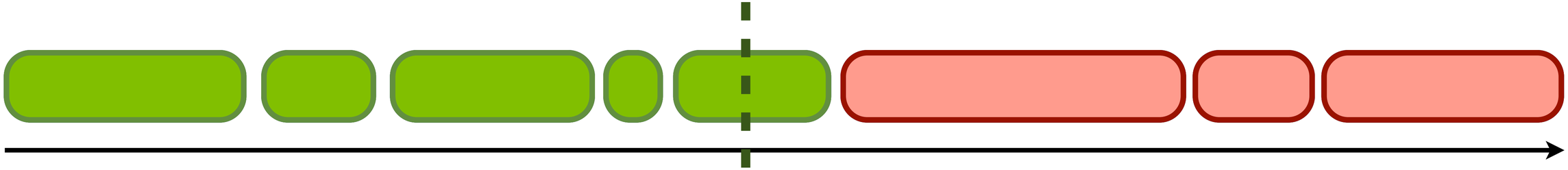


Is there a giant in the middle?

- Yes: $T(n) \leq T(n/3) + T(n/3) + \mathcal{O}(n^\omega) + \mathcal{O}(n^\omega \log n)$
- No: $T(n) \leq T(2n/3) + T(n/3) + \mathcal{O}(n^\omega \log n)$

recursive parts | giant SCC | merge edge-cuts

First: compute SCCs and any topological order in $\mathcal{O}(n+m)$



Is there a giant in the middle?

- Yes: $T(n) \leq T(n/3) + T(n/3) + \mathcal{O}(n^\omega) + \mathcal{O}(n^\omega \log n)$
- No: $T(n) \leq T(2n/3) + T(n/3) + \mathcal{O}(n^\omega \log n)$

recursive parts | giant SCC | merge edge-cuts

Dominating: $\mathcal{O}(n^\omega \log n)$

More 2-Reachability

ETH zürich

Main Result All-pairs 2-edge-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

More 2-Reachability

ETH zürich

Main Result All-pairs 2-edge-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?
- u-v-path avoiding e?
- u-v-path avoiding w?

More 2-Reachability

ETH zürich

Main Result All-pairs 2-edge-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?
- u-v-path avoiding e?
- u-v-path avoiding w?

Preprocessing: $\mathcal{O}(n^\omega \log n)$

Query for arbitrary u,v,w,e: $\mathcal{O}(1)$

More 2-Reachability

ETH zürich

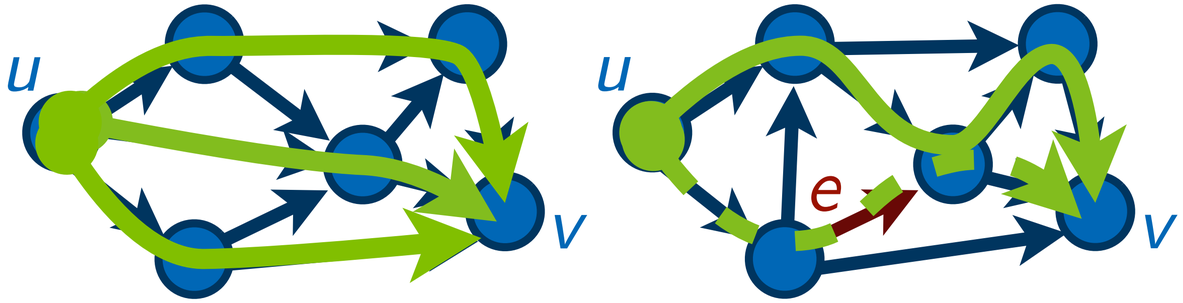
Main Result All-pairs 2-edge-reachability with witnesses in $\mathcal{O}(n^\omega \log n)$

2-Reachability

- two edge-disjoint u-v-path?
- two vertex-disjoint u-v-path?
- u-v-path avoiding e?
- u-v-path avoiding w?

Open whether also in $\tilde{\mathcal{O}}(n^\omega)$

- k edge-disjoint u-v-path?
- replacement paths



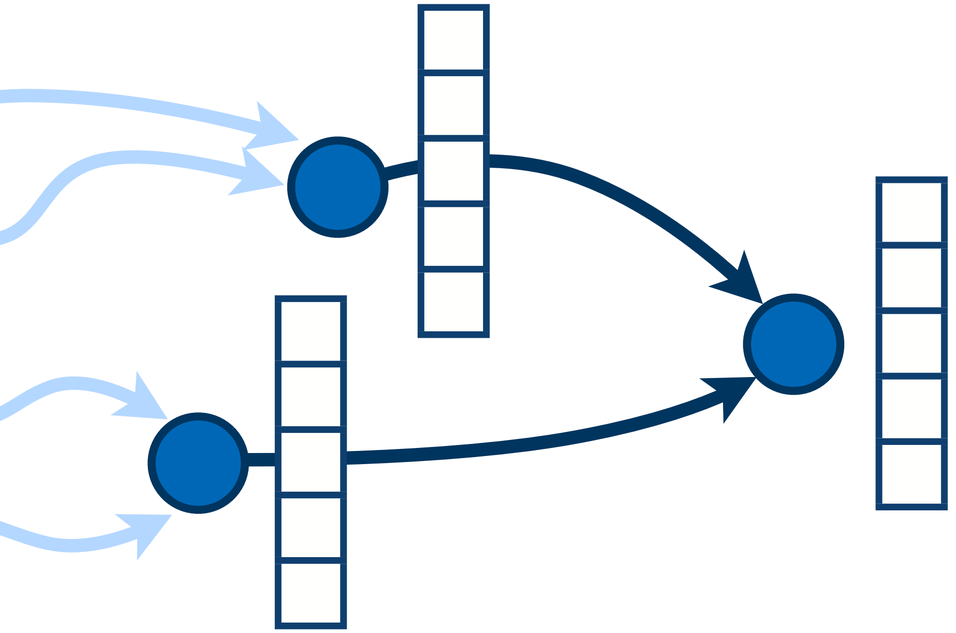
Related work: fast digraph queries after preprocessing

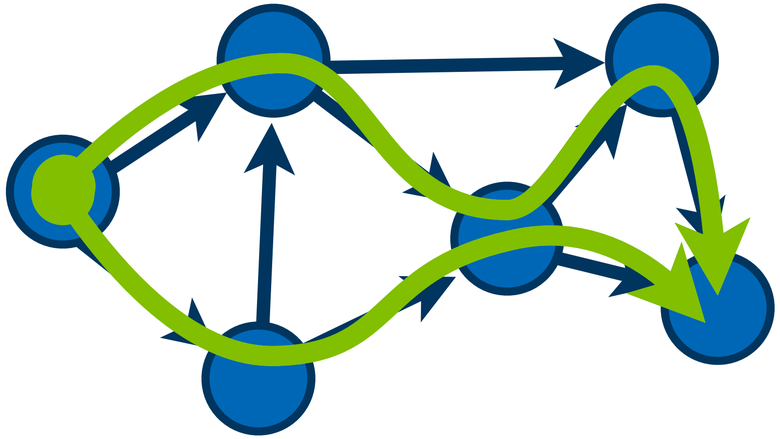
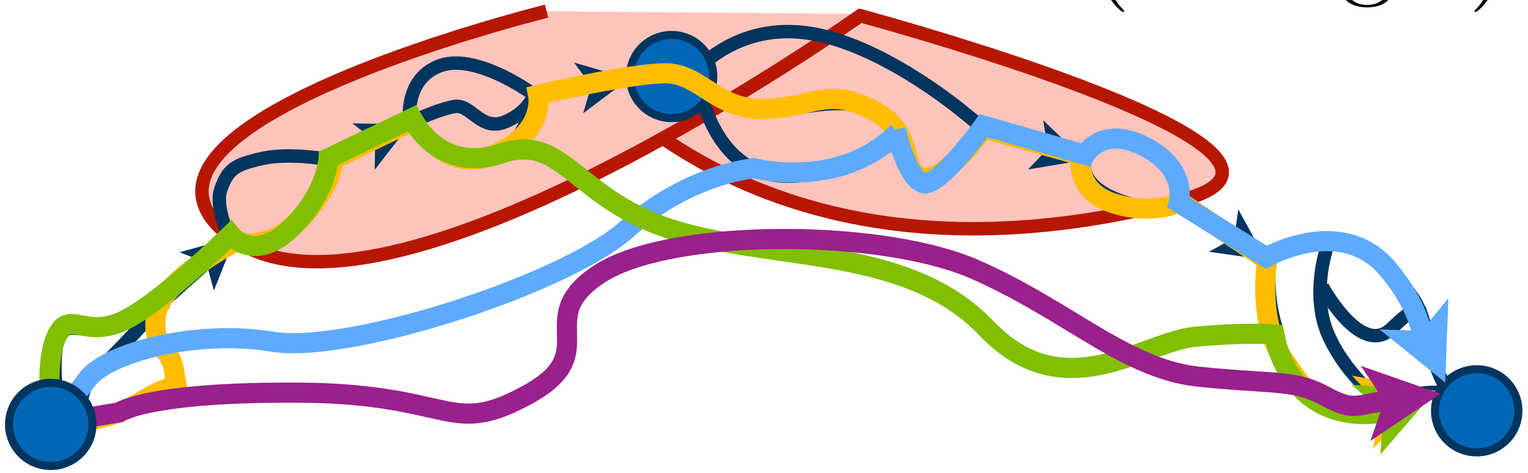
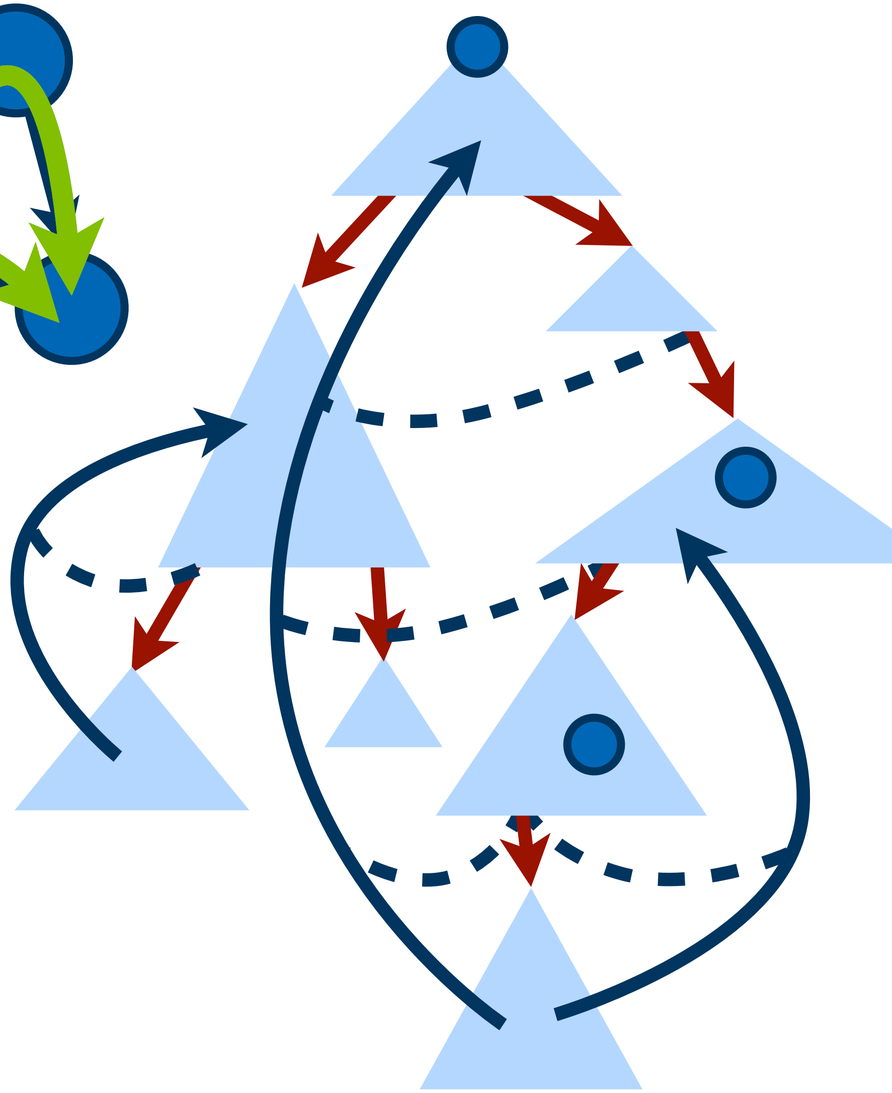
Related work: fast digraph queries after preprocessing

- Boolean mat. mult. with witnesses in $\mathcal{O}(n^\omega \log n)$ [Alon et al.'92]
- undirected shortest paths in $\mathcal{O}(n^\omega \log n)$ [Seidel'92]
- directed shortest paths in $\mathcal{O}(n^{(3+\omega)/2} \log n)$ [Alon et al.'92]
- bottleneck paths in $\mathcal{O}(n^{(3+\omega)/2} \log n)$ [Duan, Pettie'09]
- s-t-replacement shortest paths in $\tilde{\mathcal{O}}(n^\omega)$ [Williams'10]
- common ancestor in $\mathcal{O}(n^\omega)$ [Bender et al.'05]
- lowest common ancestor in $\mathcal{O}(n^{2+1/(4-\omega)})$ [Kowaluk, Lingas'05]
- junction in $\tilde{\mathcal{O}}(n^\omega)$ [Yuster'08]

Thank you for your attention!

ETH zürich





$$\mathcal{O}(n^{\omega} \log n)$$