Smallest Witnesses for RPQs

Antoine Amarilli[1]
Joint work with Benoît Groz, Nicole Wein

May 28, 2025

[1]Inria Lille

Smallest Witness for Boolean Queries

- Fix a Boolean query Q

Smallest Witness for Boolean Queries

- Fix a Boolean query Q

$Q : \exists xyz\ R(x, y), R(y, z)$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q

  $\rightarrow$ Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$

$Q : \exists xyz\ R(x, y), R(y, z)$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - $\rightarrow$ Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D

$Q : \exists xyz\ R(x, y), R(y, z)$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - $\rightarrow$ Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - → Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D
  - → We will always require that $D \models Q$

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - → Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D
  - → We will always require that $D \models Q$

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - → Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D
  - → We will always require that $D \models Q$
- Witness: a subdatabase $D' \subseteq D$ such that $D' \models Q$

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - → Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D
  - → We will always require that $D \models Q$
- Witness: a subdatabase $D' \subseteq D$ such that $D' \models Q$

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2\} \models Q$

Smallest Witness for Boolean Queries

- Fix a **Boolean query** Q
  - $\rightarrow$ Take Q **monotone**: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a **database** D
  - $\rightarrow$ We will always require that $D \models Q$
- **Witness**: a **subdatabase** $D' \subseteq D$ such that $D' \models Q$
- **Smallest witness**: a witness of **minimum size** (#facts)

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

Smallest Witness for Boolean Queries

- Fix a Boolean query Q
  - → Take Q monotone: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a database D
  - → We will always require that $D \models Q$
- Witness: a subdatabase $D' \subseteq D$ such that $D' \models Q$
- Smallest witness: a witness of minimum size (#facts)

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_3\} \models Q$

Smallest Witness for Boolean Queries

- Fix a **Boolean query** Q
  - $\rightarrow$ Take Q **monotone**: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a **database** D
  - $\rightarrow$ We will always require that $D \models Q$
- **Witness**: a **subdatabase** $D' \subseteq D$ such that $D' \models Q$
- **Smallest witness**: a witness of **minimum size** (#facts)

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

- We study **data complexity**: the query Q is fixed, the input is the database D

Smallest Witness for Boolean Queries

- Fix a **Boolean query** Q
  - → Take Q **monotone**: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a **database** D
  - → We will always require that $D \models Q$
- **Witness**: a **subdatabase** $D' \subseteq D$ such that $D' \models Q$
- **Smallest witness**: a witness of **minimum size** (#facts)

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

- We study **data complexity**: the query Q is fixed, the input is the database D
- We focus on **Boolean queries**
  - → Complexity is different when we have a **set of answers** [Hu and Sintos, 2024]

Smallest Witness for Boolean Queries

- Fix a **Boolean query** Q
  - → Take Q **monotone**: if $D \subseteq D'$ and $D \models Q$ then $D' \models Q$
- Receive as input a **database** D
  - → We will always require that $D \models Q$
- **Witness**: a **subdatabase** $D' \subseteq D$ such that $D' \models Q$
- **Smallest witness**: a witness of **minimum size** (#facts)

$Q : \exists xyz\ R(x, y), R(y, z)$

R		
a	b	f_1
b	c	f_2
a	a	f_3
d	e	f_4

$\{f_1, f_2, f_3, f_4\} \models Q$

- We study **data complexity**: the query Q is fixed, the input is the database D
- We focus on **Boolean queries**
  - → Complexity is different when we have a **set of answers** [Hu and Sintos, 2024]
- For a UCQ Q the task is always **PTIME** (smallest witnesses have size $\leq |Q|$)

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ

$$\Sigma = \{a, b\}$$

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ

$$\Sigma = \{a, b\}$$

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ

$$\Sigma = \{a, b\}$$

$$L = a^* \, b \, a^*$$

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - $\rightarrow$ Vertices V and edges $E \subseteq V \times \Sigma \times V$

$\Sigma = \{a, b\}$

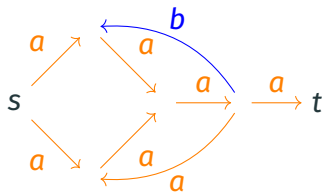
$L = a^* \, b \, a^*$

Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - $\rightarrow$ Vertices V and edges $E \subseteq V \times \Sigma \times V$

$\Sigma = \{a, b\}$

$L = a^* \, b \, a^*$

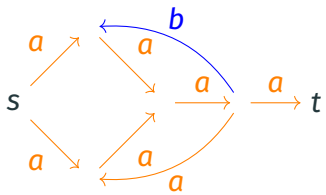


Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - $\rightarrow$ Vertices V and edges $E \subseteq V \times \Sigma \times V$
- We have $D \models \mathrm{RPQ}_L(s, t)$ with constants s, t if:
  - We have a walk w in D from the source s to the target t
  - The concatenation of the edge labels of w is in L

$\Sigma = \{a, b\}$

$L = a^* \, b \, a^*$

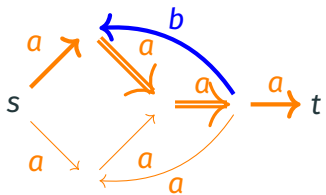


Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - $\rightarrow$ Vertices V and edges $E \subseteq V \times \Sigma \times V$
- We have $D \models \mathrm{RPQ}_L(s, t)$ with constants s, t if:
  - We have a walk w in D from the source s to the target t
  - The concatenation of the edge labels of w is in L

$\Sigma = \{a, b\}$

$L = a^* \, b \, a^*$

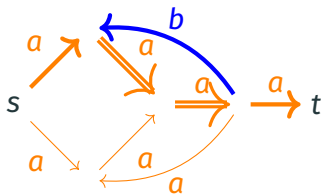


Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - $\rightarrow$ Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - $\rightarrow$ Vertices V and edges $E \subseteq V \times \Sigma \times V$
- We have $D \models \mathrm{RPQ}_L(s, t)$ with constants s, t if:
  - We have a walk w in D from the source s to the target t
  - The concatenation of the edge labels of w is in L
  - Note: w is not necessarily a simple path

$\Sigma = \{a, b\}$

$L = a^* \, b \, a^*$

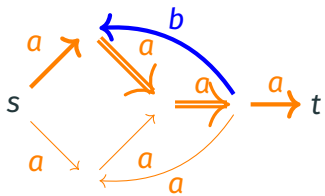


Regular Path Queries (RPQs)

- Fix an finite alphabet Σ
- Regular path query RPQ_L
  - → Given by a regular language L on Σ
- Graph database $D = (V, E)$
  - → Vertices V and edges $E \subseteq V \times \Sigma \times V$
- We have $D \models \mathrm{RPQ}_L(s, t)$ with constants s, t if:
  - We have a walk w in D from the source s to the target t
  - The concatenation of the edge labels of w is in L
  - Note: w is not necessarily a simple path
- We can check in PTIME whether $D \models \mathrm{RPQ}_L(s, t)$
  - → Evaluate in Datalog, or do product construction of D with automaton for L

$\Sigma = \{a, b\}$

$L = a^* \, b \, a^*$



Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s, t)$

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s, t)$

$\rightarrow$ What is the complexity of SW_L depending on the language L?

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s,t)$

$\rightarrow$ What is the complexity of SW_L depending on the language L?

First results:

- SW_L is always in NP
  - $\rightarrow$ Guess the subdatabase D' with $|D'| \leq k$ and verify $D' \models \mathsf{RPQ}_L(s,t)$

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s,t)$

→ What is the complexity of SW_L depending on the language L?

First results:

- SW_L is always in NP
  → Guess the subdatabase D' with $|D'| \leq k$ and verify $D' \models \mathsf{RPQ}_L(s,t)$
- If L is a finite language, then SW_L is in PTIME
  → Follows from tractability for UCQs

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s,t)$

$\rightarrow$ What is the complexity of SW_L depending on the language L?

First results:

- SW_L is always in NP
  - $\rightarrow$ Guess the subdatabase D' with $|D'| \leq k$ and verify $D' \models \mathsf{RPQ}_L(s,t)$
- If L is a finite language, then SW_L is in PTIME
  - $\rightarrow$ Follows from tractability for UCQs
- For $L = a^*$, we have that SW_L is...

Smallest Witness for RPQs

- Decision problem SW_L for a fixed regular language L:
  - Input: graph database D, vertices s and t integer $k \in \mathbb{N}$
  - Output: is there a subdatabase $D' \subseteq D$ with $\leq k$ edges that satisfies $\mathsf{RPQ}_L(s,t)$

→ What is the complexity of SW_L depending on the language L?

First results:

- SW_L is always in NP
  → Guess the subdatabase D' with $|D'| \leq k$ and verify $D' \models \mathsf{RPQ}_L(s,t)$
- If L is a finite language, then SW_L is in PTIME
  → Follows from tractability for UCQs
- For $L = a^*$, we have that SW_L is... in PTIME
  → Compute the shortest path from s to t and check that it has $\leq k$ edges

Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

- First guess the b-edge $b(u, v)$

Tractability by Reducing to ...

What about $L = a^* b a^*$?

- First guess the b-edge $b(u, v)$

Tractability by Reducing to ...

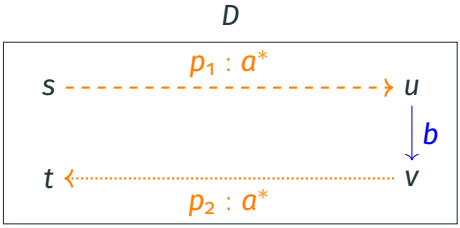
What about $L = a^* \, b \, a^*$?

- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t

Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

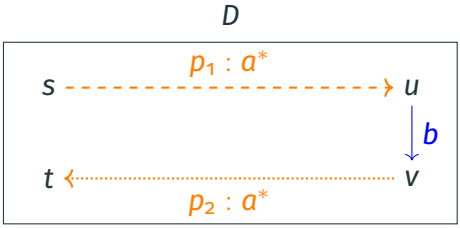
- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t



Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

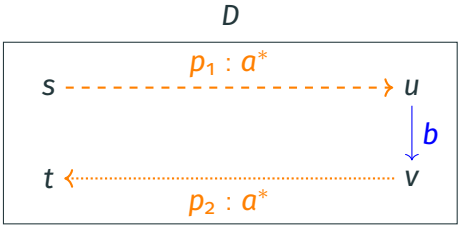
- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- **Problem:** p_1 and p_2 may share edges



Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

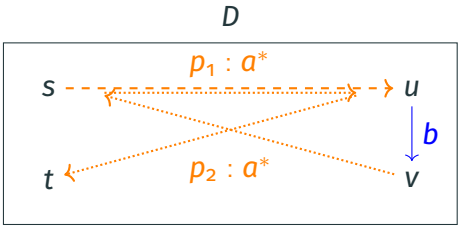
- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- Problem: p_1 and p_2 may share edges
  - What matters is to minimize $|p_1 \cup p_2|$



Tractability by Reducing to ...

What about $L = a^* \, b \, a^*$?

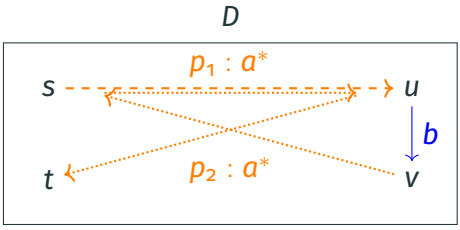
- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- Problem: p_1 and p_2 may share edges
  - What matters is to minimize $|p_1 \cup p_2|$



Tractability by Reducing to … Directed Steiner Network (DSN)

What about $L = a^* b a^*$?

- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- Problem: p_1 and p_2 may share edges
  - What matters is to minimize $|p_1 \cup p_2|$



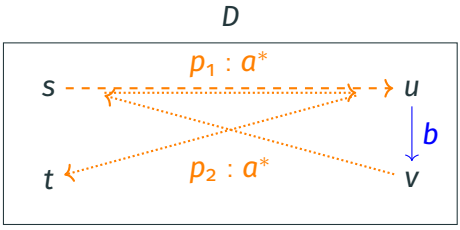
Theorem (Feldman and Ruhl, 2006)

The following Directed Steiner Network problem is in PTIME for any fixed $\ell \in \mathbb{N}$:

Tractability by Reducing to … Directed Steiner Network (DSN)

What about $L = a^* b a^*$?

- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- Problem: p_1 and p_2 may share edges
  - What matters is to minimize $|p_1 \cup p_2|$



Theorem (Feldman and Ruhl, 2006)

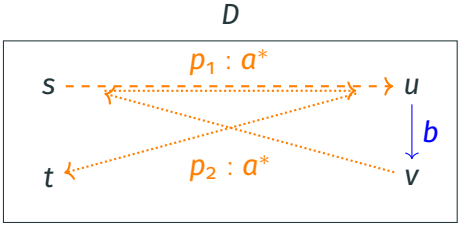
The following Directed Steiner Network problem is in PTIME for any fixed $\ell \in \mathbb{N}$:

- *Input: directed graph G, vertices $s_1, t_1, \ldots, s_\ell, t_\ell$*

Tractability by Reducing to ... Directed Steiner Network (DSN)

What about $L = a^* b a^*$?

- First guess the b-edge $b(u, v)$
- Idea: compute shortest a-paths p_1 from s to u and p_2 from v to t
- Problem: p_1 and p_2 may share edges
  - What matters is to minimize $|p_1 \cup p_2|$



Theorem (Feldman and Ruhl, 2006)

The following Directed Steiner Network problem is in PTIME for any fixed $\ell \in \mathbb{N}$:

- *Input: directed graph G, vertices $s_1, t_1, \ldots, s_\ell, t_\ell$*
- *Output: subgraph $G' \subseteq G$ of minimum cardinality that has a directed path from s_i to t_i for each $1 \leq i \leq \ell$*

Parity Constraints

What about $L = (aa)^*$?

Parity Constraints

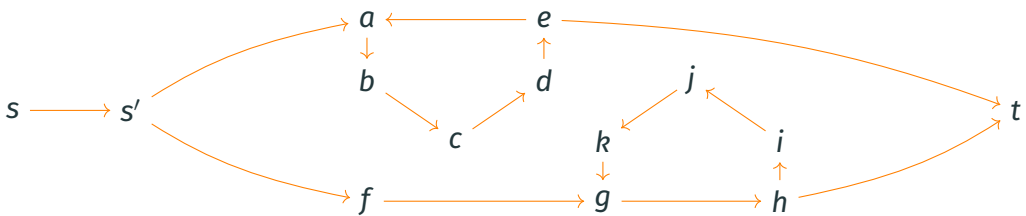
What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

Parity Constraints

What about $L = (aa)^*$?

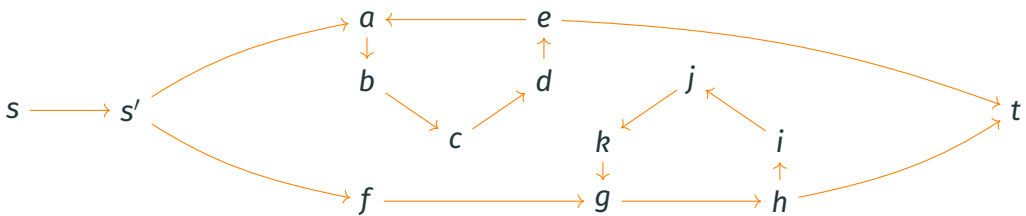
"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"



Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

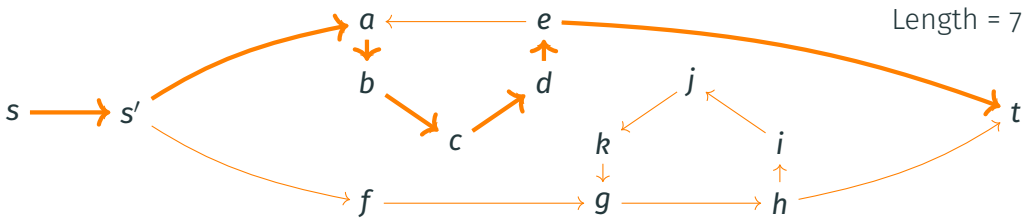


- It may not be a simple path,

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

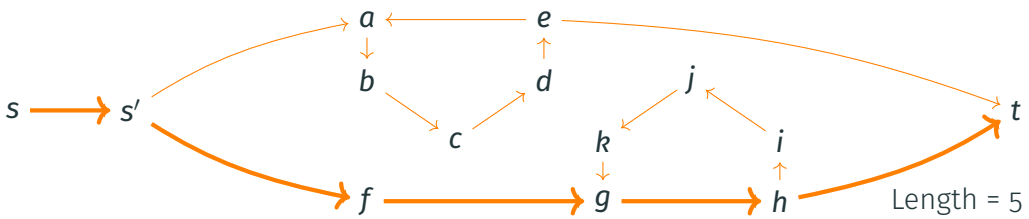


- It may not be a simple path,

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

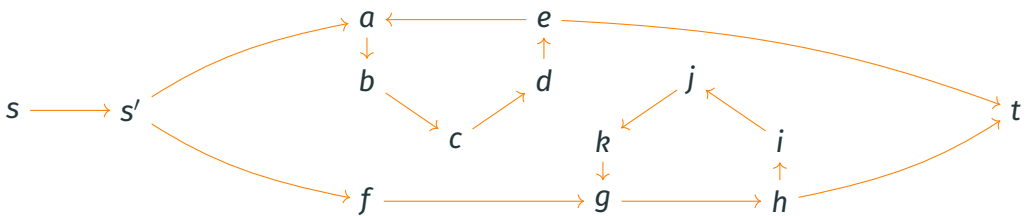


- It may not be a simple path,

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

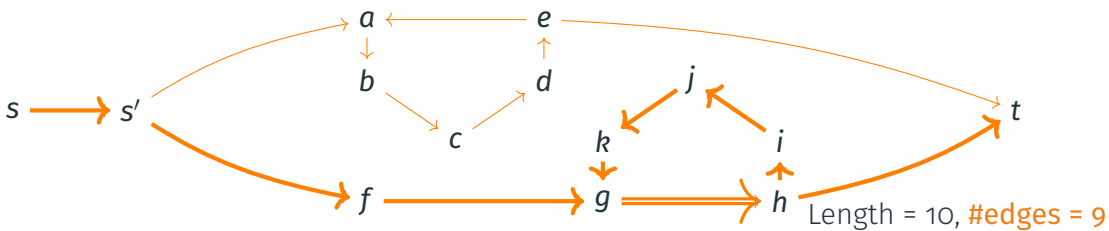


- It may not be a simple path, and it may not be a shortest walk!

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

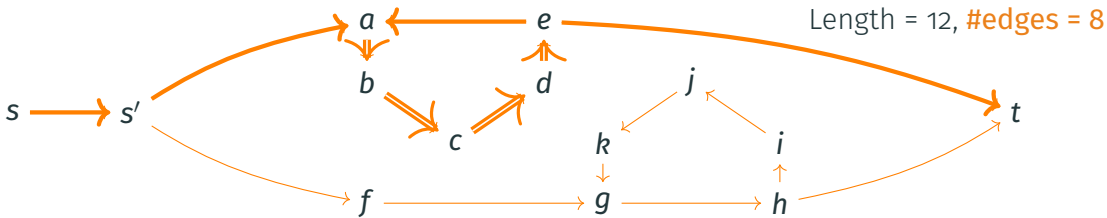


- It may not be a simple path, and it may not be a shortest walk!

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"

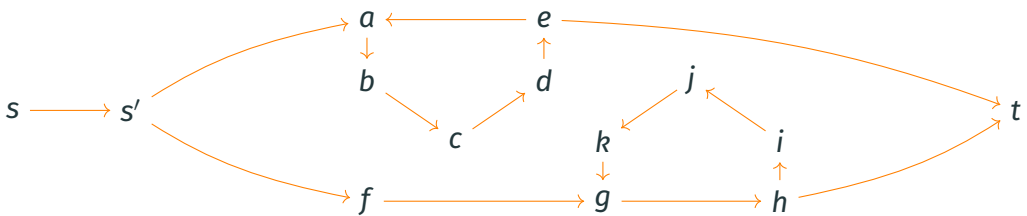


- It may not be a simple path, and it may not be a shortest walk!

Parity Constraints

What about $L = (aa)^*$?

"Given a graph G, source s, and target t, find an st-walk of even length that uses the least number of distinct edges"



- It may not be a simple path, and it may not be a shortest walk!
- Is this problem in PTIME?

Tractability for Parity Constraints (Sketch)

"Find an st-walk w of even length that uses the least number of distinct edges"

Tractability for Parity Constraints (Sketch)

"Find an st-walk w of even length that uses the least number of distinct edges"

- There are no even-length cycles in w

Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***

Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

Tractability for Parity Constraints (Sketch)

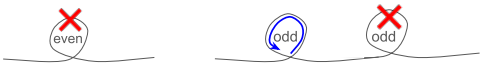
"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***



→ Case 1: ***w*** has no cycle (and length = #edges)

Tractability for Parity Constraints (Sketch)

"Find an st-walk W of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W

$\rightarrow$ Case 1: W has no cycle (and length = #edges)

$\rightarrow$ Case 2: W is a path P_1 then an odd cycle C then a path P_2

Tractability for Parity Constraints (Sketch)

"Find an st-walk W of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W

$\rightarrow$ Case 1: W has no cycle (and length = #edges)

$\rightarrow$ Case 2: W is a path P_1 then an odd cycle C then a path P_2

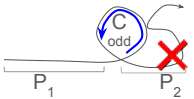
- When P_2 leaves C it does not **re-enter** C

Tractability for Parity Constraints (Sketch)

"Find an st-walk W of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W

$\rightarrow$ Case 1: W has no cycle (and length = #edges)



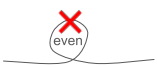
$\rightarrow$ Case 2: W is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C

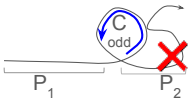
Tractability for Parity Constraints (Sketch)

"Find an st-walk W of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W



$\rightarrow$ Case 1: W has no cycle (and length = #edges)



$\rightarrow$ Case 2: W is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter C**
- P_2 cannot enter P_1, leave P_1, and **re-enter later**

Tractability for Parity Constraints (Sketch)

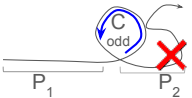
"Find an st-walk W of even length that uses the least number of distinct edges"

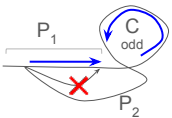
- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W

$\rightarrow$ Case 1: W has no cycle (and length = #edges)

$\rightarrow$ Case 2: W is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**





Tractability for Parity Constraints (Sketch)

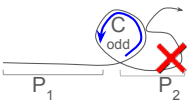
"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

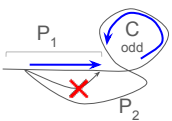
- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

→ Case 1: ***w*** has no cycle (and length = #edges)

→ Case 2: ***w*** is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**



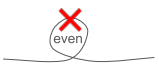


So Case 2 must be:

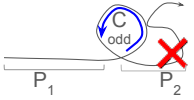
Tractability for Parity Constraints (Sketch)

"Find an st-walk w of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in w
- There are no **two odd-length cycles** in w

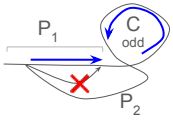


$\rightarrow$ Case 1: w has no cycle (and length = #edges)



$\rightarrow$ Case 2: w is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**

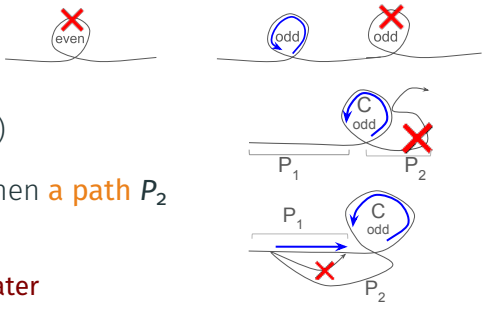


So Case 2 must be:

Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

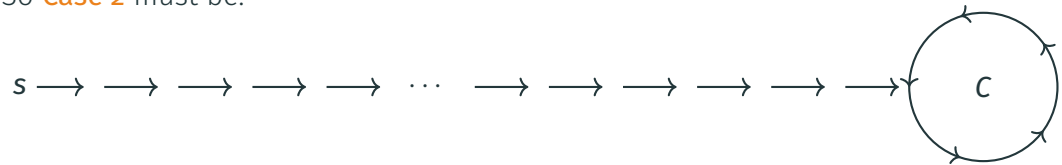


→ Case 1: ***w*** has no cycle (and length = #edges)

→ Case 2: ***w*** is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**

So Case 2 must be:



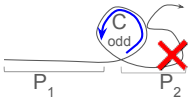
Tractability for Parity Constraints (Sketch)

"Find an st-walk W of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in W
- There are no **two odd-length cycles** in W

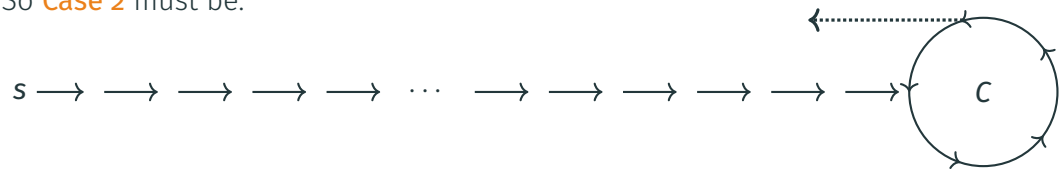
→ Case 1: W has no cycle (and length = #edges)

→ Case 2: W is a path P_1 then an odd cycle C then a path P_2



- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**

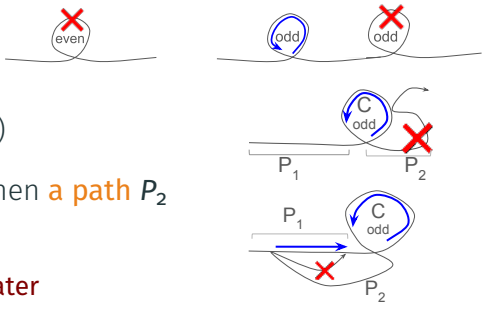
So Case 2 must be:



Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

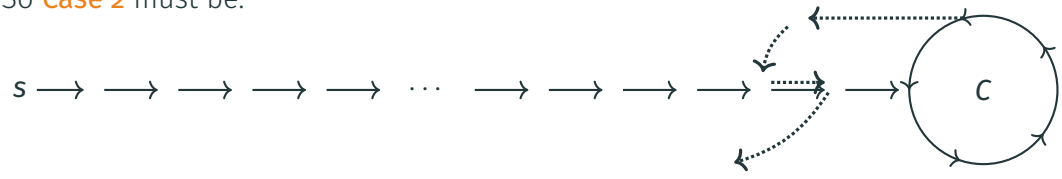


→ Case 1: ***w*** has no cycle (and length = #edges)

→ Case 2: ***w*** is a path P_1 then an odd cycle C then a path P_2

- When P_2 leaves C it does not **re-enter** C
- P_2 cannot enter P_1, leave P_1, and **re-enter later**

So Case 2 must be:

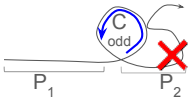


Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

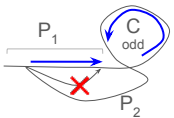
- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

→ Case 1: ***w*** has no cycle (and length = #edges)

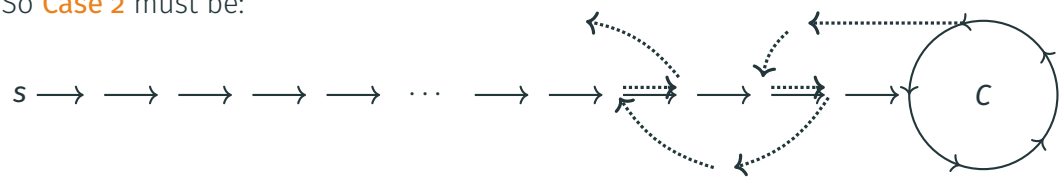


→ Case 2: ***w*** is a path P_1 then an odd cycle ***C*** then a path P_2

- When P_2 leaves ***C*** it does not **re-enter *C***
- P_2 cannot enter P_1, leave P_1, and **re-enter later**



So Case 2 must be:

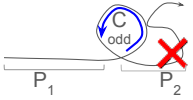


Tractability for Parity Constraints (Sketch)

"Find an ***st***-walk ***w*** of even length that uses the least number of distinct edges"

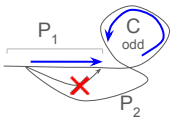
- There are no **even-length cycles** in ***w***
- There are no **two odd-length cycles** in ***w***

→ Case 1: ***w*** has no cycle (and length = #edges)

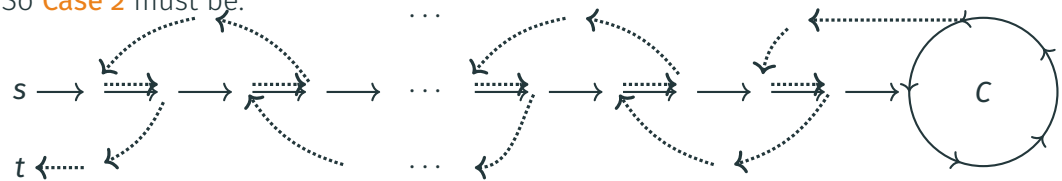


→ Case 2: ***w*** is a path P_1 then an odd cycle ***C*** then a path P_2

- When P_2 leaves ***C*** it does not **re-enter *C***
- P_2 cannot enter P_1, leave P_1, and **re-enter later**



So Case 2 must be:



Tractability for Modularity Constraints

What about $L = (a^q)^* a^r$?

Tractability for Modularity Constraints

What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Tractability for Modularity Constraints

What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Tractability for Modularity Constraints

What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours

Tractability for Modularity Constraints

What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u

Tractability for Modularity Constraints

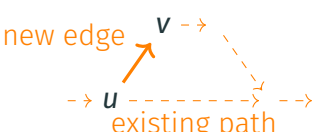
What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u



Tractability for Modularity Constraints

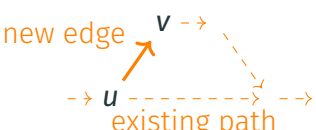
What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u
  - Only useful to change the remainder of the length



Tractability for Modularity Constraints

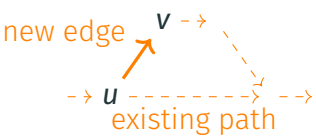
What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u
  - Only useful to change the remainder of the length
  - After $O(\log q)$ detours, all possible remainders achieved



Tractability for Modularity Constraints

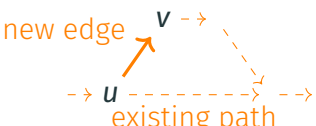
What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u
  - Only useful to change the remainder of the length
  - After $O(\log q)$ detours, all possible remainders achieved



- The cutwidth of a graph spanned by walk w is bounded by $O(\#\text{detours of } w)$

Tractability for Modularity Constraints

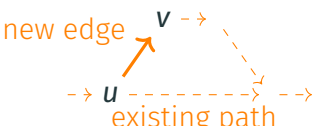
What about $L = (a^q)^* a^r$? (st-walk of length $r \bmod q$ with min. #distinct edges)

Theorem (A., Groz, Wein, ITCS'25)

For any fixed $q > 0$ and $0 \leq r < q$, letting $L = (a^q)^* a^r$, the problem SW_L is in PTIME

Proof sketch:

- An optimal walk w will not have too many detours
  - Detour: taking a new edge (u, v) then going back to a vertex already reachable from u
  - Only useful to change the remainder of the length
  - After $O(\log q)$ detours, all possible remainders achieved



- The cutwidth of a graph spanned by walk w is bounded by $O(\#\text{detours of } w)$
- Bounded-cutwidth subgraphs can be found by dynamic programming

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

An Intractable Case

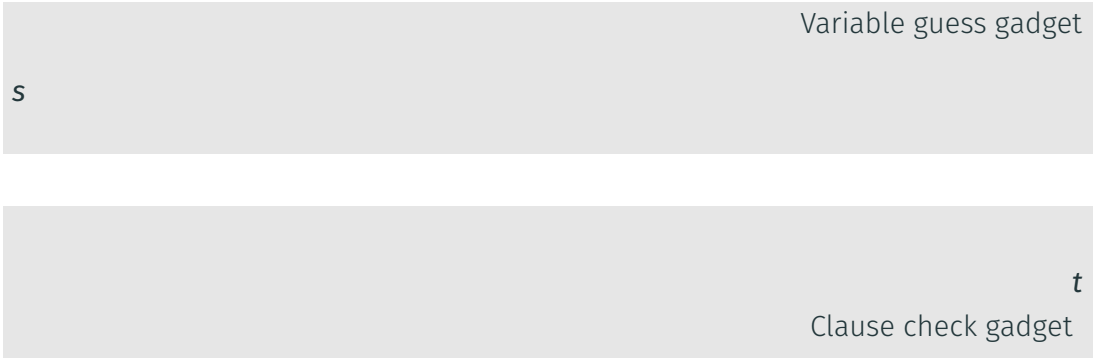
What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is **NP-complete** (from 3-SAT)

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

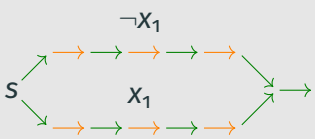
Then SW_L is NP-complete (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is NP-complete (from 3-SAT)



Variable guess gadget

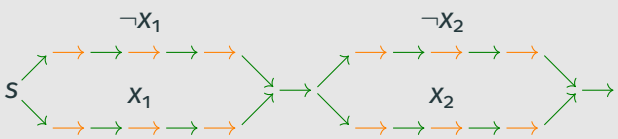
t

Clause check gadget

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is NP-complete (from 3-SAT)



Variable guess gadget

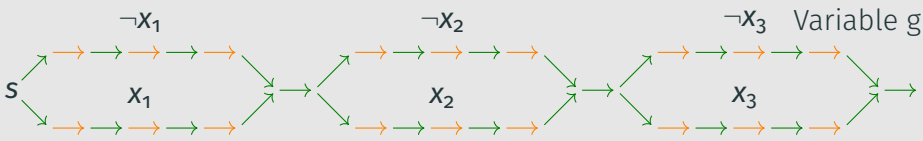
t

Clause check gadget

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is NP-complete (from 3-SAT)



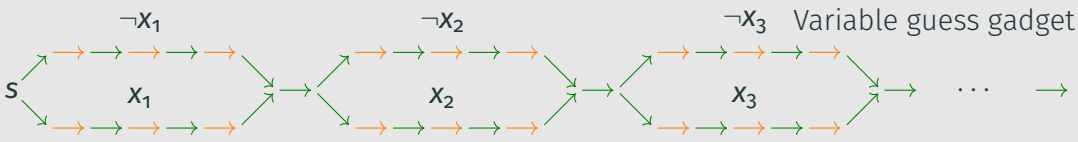
t

Clause check gadget

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is **NP-complete** (from 3-SAT)



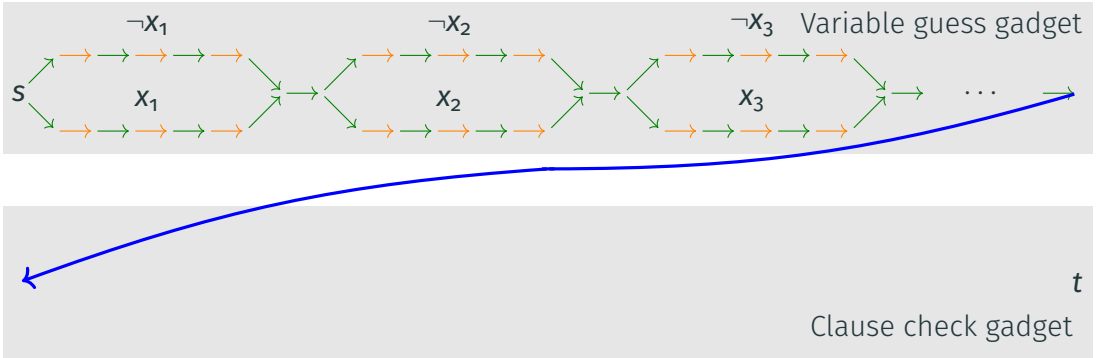
t

Clause check gadget

An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

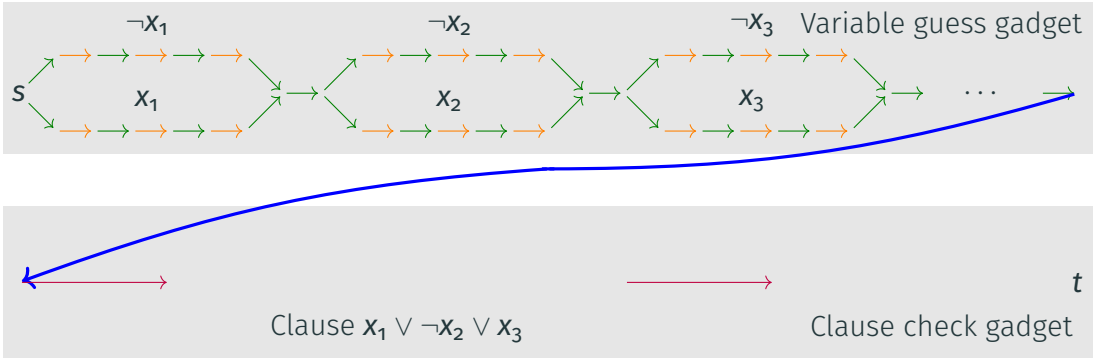
Then SW_L is NP-complete (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

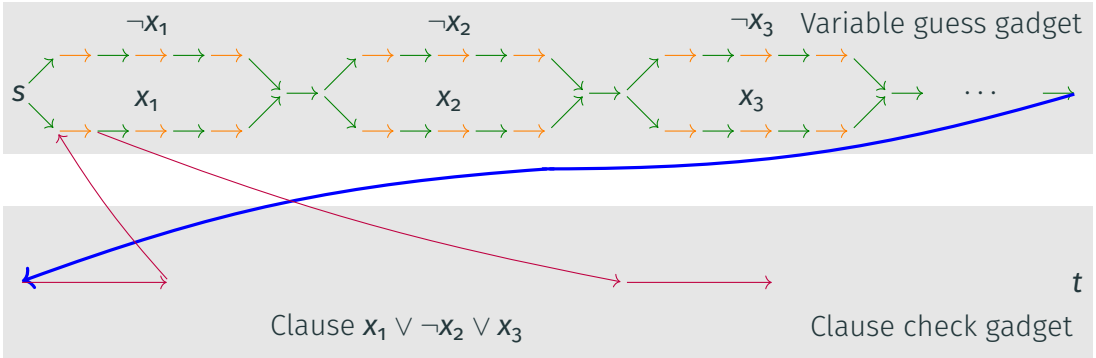
Then SW_L is **NP-complete** (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

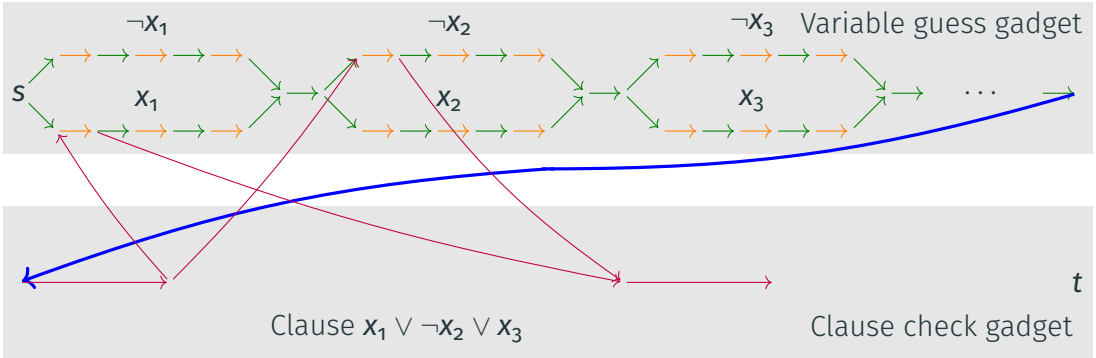
Then SW_L is **NP-complete** (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

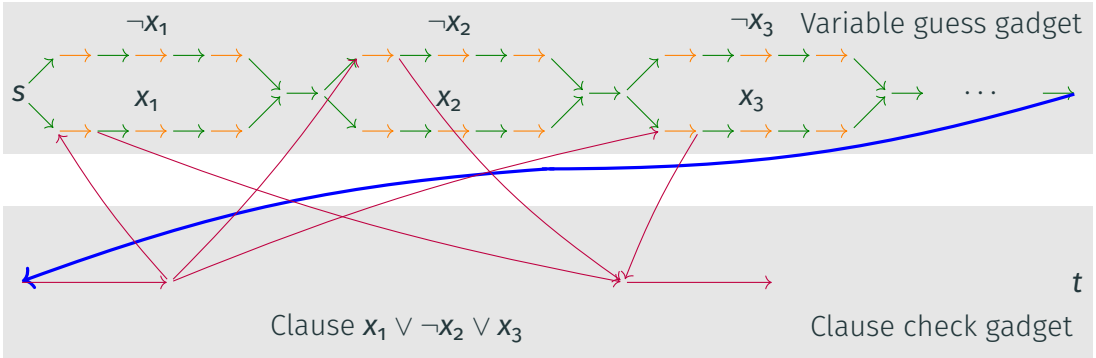
Then SW_L is **NP-complete** (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$?

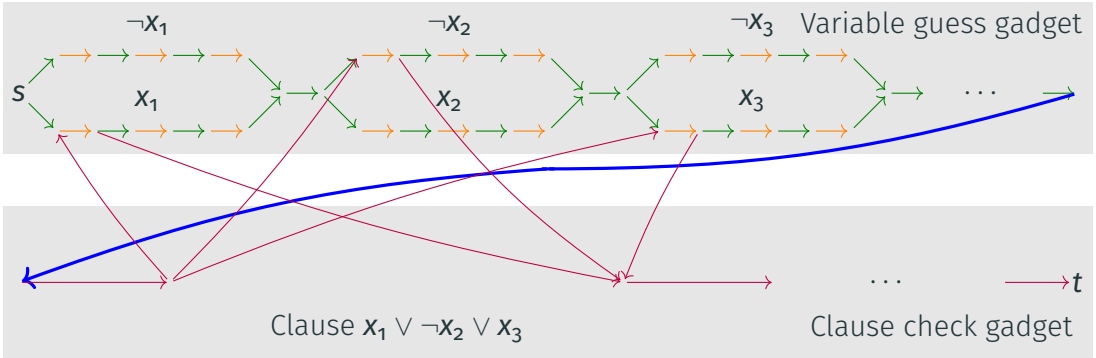
Then SW_L is **NP-complete** (from 3-SAT)



An Intractable Case

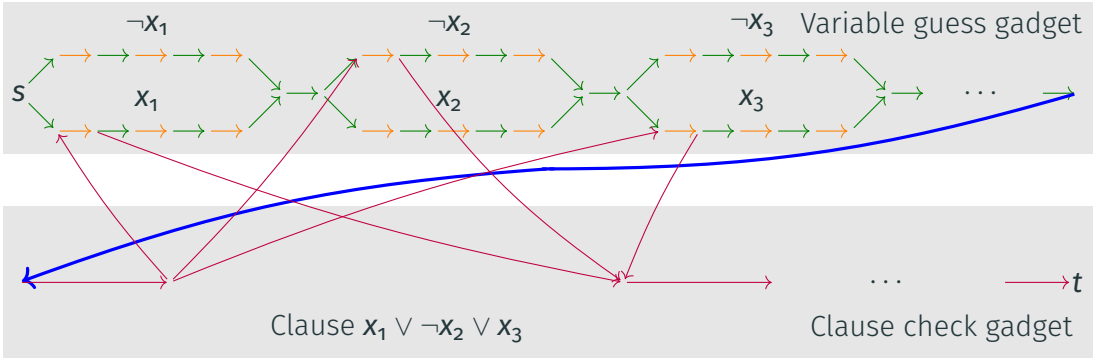
What about $L = (a + c)^* \, b \, (a + d)^*$?

Then SW_L is **NP-complete** (from 3-SAT)



An Intractable Case

What about $L = (a + c)^* \, b \, (a + d)^*$? Then SW_L is **NP-complete** (from 3-SAT)



Minimizing #distinct edges forces that all a-edges in the clause checks are revisits

Open Question 1: Dichotomy on Smallest Witness for RPQs?

Conjecture (with Benoît Groz)

For every regular language L, then SW_L is either in PTIME or NP-complete

Open Question 1: Dichotomy on Smallest Witness for RPQs?

Conjecture (with Benoît Groz)

For every regular language L, then SW_L is either in PTIME or NP-complete

Current status:

- Tractability via Directed Steiner Network or by the $r \bmod q$ argument
- These two techniques can be combined

Open Question 1: Dichotomy on Smallest Witness for RPQs?

Conjecture (with Benoît Groz)

For every regular language L, then SW_L is either in PTIME or NP-complete

Current status:

- Tractability via Directed Steiner Network or by the $r \bmod q$ argument
- These two techniques can be combined
- Similar techniques seem to cover:
  - Arbitrary groups, beyond $\mathbb{Z}/q\mathbb{Z}$
  - The "multilevel" case: $a_1^* \, b_1 \, (a_1 + a_2)^* \, b_2 \cdots (a_1 + \cdots + a_n)^*$

Open Question 1: Dichotomy on Smallest Witness for RPQs?

Conjecture (with Benoît Groz)

For every regular language L, then SW_L *is either in PTIME or NP-complete*

Current status:

- Tractability via Directed Steiner Network or by the $r \bmod q$ argument
- These two techniques can be combined
- Similar techniques seem to cover:
  - Arbitrary groups, beyond $\mathbb{Z}/q\mathbb{Z}$
  - The "multilevel" case: $a_1^* b_1 (a_1 + a_2)^* b_2 \cdots (a_1 + \cdots + a_n)^*$

Reminiscent of CSP/VSCP but no clear connection:

> SW_L: "For fixed RHS structure H_L, given a LHS structure G, find a minimal substructure of G that does not have a homomorphism to H_L"

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- Smallest Witness for Q: a subdatabase $D' \subseteq D$ of minimum size with $D' \models Q$

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- Smallest Witness for Q: a subdatabase $D' \subseteq D$ of minimum size with $D' \models Q$
- Resilience for Q: a subdatabase $D' \subseteq D$ of maximum size with $D' \not\models Q$

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- Smallest Witness for Q: a subdatabase $D' \subseteq D$ of minimum size with $D' \models Q$
- Resilience for Q: a subdatabase $D' \subseteq D$ of maximum size with $D' \not\models Q$

What is the complexity of computing resilience?

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- Smallest Witness for Q: a subdatabase $D' \subseteq D$ of minimum size with $D' \models Q$
- Resilience for Q: a subdatabase $D' \subseteq D$ of maximum size with $D' \not\models Q$

What is the complexity of computing resilience?

- sometimes NP-complete for CQ/UCQs [Freire et al., 2015]

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- **Smallest Witness** for Q: a subdatabase $D' \subseteq D$ of **minimum size** with $D' \models Q$
- **Resilience** for Q: a subdatabase $D' \subseteq D$ of **maximum size** with $D' \not\models Q$

What is the complexity of computing resilience?

- **sometimes NP-complete** for CQ/UCQs [Freire et al., 2015]
- **in PTIME** for some Boolean RPQs, e.g., by computing a **minimum cut**

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- **Smallest Witness** for Q: a subdatabase $D' \subseteq D$ of **minimum size** with $D' \models Q$
- **Resilience** for Q: a subdatabase $D' \subseteq D$ of **maximum size** with $D' \not\models Q$

What is the complexity of computing resilience?

- **sometimes NP-complete** for CQ/UCQs [Freire et al., 2015]
- **in PTIME** for some Boolean RPQs, e.g., by computing a **minimum cut**
- **Dichotomy for (2)RPQs** via VCSPs [Bodirsky, Semanišinová, Lutz, 2024]
  - But for databases **with weighted facts**, and opaque (but decidable) criterion

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- **Smallest Witness** for Q: a subdatabase $D' \subseteq D$ of **minimum size** with $D' \models Q$
- **Resilience** for Q: a subdatabase $D' \subseteq D$ of **maximum size** with $D' \not\models Q$

What is the complexity of computing resilience?

- **sometimes NP-complete** for CQ/UCQs [Freire et al., 2015]
- **in PTIME** for some Boolean RPQs, e.g., by computing a **minimum cut**
- **Dichotomy for (2)RPQs** via VCSPs [Bodirsky, Semanišinová, Lutz, 2024]
  - But for databases **with weighted facts**, and opaque (but decidable) criterion

→ Is there a **unified understanding** of the tractability frontier of both problems?

Open Question 2: Connections to Resilience?

Fix a monotone Boolean query Q, read as input a database D

- **Smallest Witness** for Q: a subdatabase $D' \subseteq D$ of **minimum size** with $D' \models Q$
- **Resilience** for Q: a subdatabase $D' \subseteq D$ of **maximum size** with $D' \not\models Q$

What is the complexity of computing resilience?

- **sometimes NP-complete** for CQ/UCQs [Freire et al., 2015]
- **in PTIME** for some Boolean RPQs, e.g., by computing a **minimum cut**
- **Dichotomy for (2)RPQs** via VCSPs [Bodirsky, Semanišinová, Lutz, 2024]
  - But for databases **with weighted facts**, and opaque (but decidable) criterion

→ Is there a **unified understanding** of the tractability frontier of both problems?

Thanks for your attention!

Acknowledgements

Slide 7: Thanks to Nicole Wein for preparing some of the drawings.

References i

- Amarilli, A., Gatterbauer, W., Makhija, N., and Monet, M. (2025a).
  Resilience for regular path queries: Towards a complexity classification.
  In *PODS*.
- Amarilli, A., Groz, B., and Wein, N. (2025b).
  Edge-minimum walk of modular length in polynomial time.
  In *ITCS*.
- Bodirsky, M., Semanišinová, Ž., and Lutz, C. (2024).
  The complexity of resilience problems via valued constraint satisfaction problems.
  In *LICS*.

References ii

- Feldman, J. and Ruhl, M. (2006).
  The directed steiner network problem is tractable for a constant number of terminals.
  SIAM Journal on Computing, 36(2).
- Freire, C., Gatterbauer, W., Immerman, N., and Meliou, A. (2015).
  The complexity of resilience and responsibility for self-join-free conjunctive queries.
  PVLDB, 9(3).
- Hu, X. and Sintos, S. (2024).
  Finding smallest witnesses for conjunctive queries.
  In *ICDT*.

References iii

Miao, Z., Roy, S., and Yang, J. (2019).
Explaining wrong queries using small examples.
In *SIGMOD*.