Combined Complexity of Probabilistic Query Evaluation

Mikaël Monet

October 12th, 2018

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
Jb	12/10	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
Jb	12/10	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve patients having an appointment with a radiologist

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve patients having an appointment with a radiologist

→ SELECT patient FROM Appointment, Has_Specialty
WHERE Appointment.doctor = Has_Specialty.doctor
AND Has_Specialty.specialty = 'radiology'

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve patients having an appointment with a radiologist

$\rightarrow$ $p := \exists d'\, t\, d : \ \textbf{Appointment}(p, d', t, d) \wedge \textbf{Has\_Specialty}(d, \text{'radiology'})$

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve doctors having at least one appointment

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve doctors having at least one appointment

→ SELECT doctor FROM Appointment

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve doctors having at least one appointment

→ $d := \exists p\, d'\, t : \ \textbf{Appointment}(p, d', t, d)$

Relational Databases

- **Databases:** store information and query it later

Has_Specialty

doctor	specialty
Dr. Sneeze	allergologist
Dr. Bone	radiology
Dr. Nail	hand surgeon
⋮	⋮

Appointment

patient	date	time	doctor
Nelly	17/04	11h	Dr. Sneeze
Jb	30/05	14h	Dr. Bone
Jb	05/11	15h	Dr. Sneeze
⋮	⋮	⋮	⋮

- **Query:** Retrieve doctors having at least one appointment

$\rightarrow$ $d := \exists p\, d'\, t : \ \textbf{Appointment}(p, d', t, d)$

- Applications: banks, institutions, libraries, movies, recipes, etc.

Uncertainty

- One usually assumes that the data is correct...

Uncertainty

- One usually assumes that the data is correct...
- ... but in many cases it is not

Uncertainty

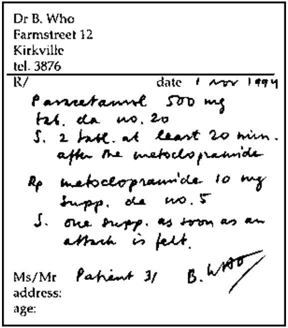
- One usually assumes that the data is correct…
- … but in many cases it is not

$\rightarrow$ Untrustworthy sources, automated information extraction, imprecise sensors in experimental sciences, etc.

Example: Optical Character Recognition

- Hospital wants to digitize and store all doctors' prescriptions

- Hospital wants to digitize and store all doctors’ prescriptions


Dr B. Who
Farmstreet 12
Kirkville
tel. 3876

R/ date 1 nov 1994

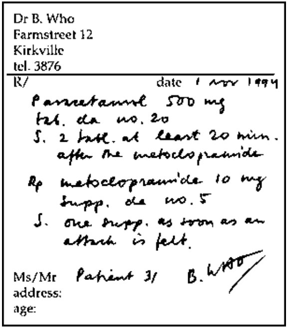
Paracetamol 500 mg
tab. da no. 20
S. 2 tabl. at least 20 min.
after the metoclopramide

Rp metoclopramide 10 mg
supp. da no. 5
S. one supp. as soon as an
attack is felt.

Ms/Mr Patient 31 B. Who
address:
age:

Example: Optical Character Recognition

- Hospital wants to digitize and store all doctors' prescriptions


Dr B. Who
Farmstreet 12
Kirkville
tel. 3876

R/ date 1 nov 1994

Paracetamol 500 mg
tab. da no. 20
S. 2 tabl. at least 20 min.
after the metoclopramide

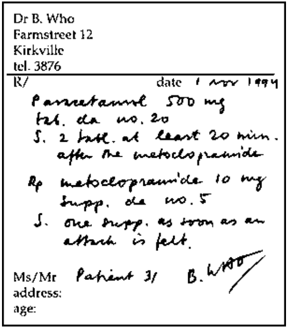
Rp metoclopramide 10 mg
supp. da no. 5
S. one supp. as soon as an
attack is felt.

Ms/Mr Patient 31 B. Who
address:
age:

OCR + NLP

Example: Optical Character Recognition

- Hospital wants to digitize and store all doctors' prescriptions

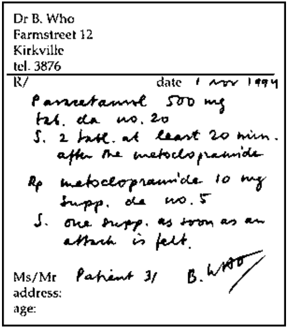

Dr B. Who
Farmstreet 12
Kirkville
tel. 3876
R/ date 1 nov 1994
Paracetamol 500 mg
tab. da no. 20
S. 2 tabl. at least 20 min.
after the metoclopramide
Rp metoclopramide 10 mg
supp. da no. 5
S. one supp. as soon as an
attack is felt.
Ms/Mr Patient 31 B. Who
address:
age:

OCR + NLP

Prescription

what	when	quantity
paracetamol	anytime	500 mg
metoclopramide	during attack	10 mg

- Hospital wants to digitize and store all doctors' prescriptions


Dr B. Who
Farmstreet 12
Kirkville
tel. 3876
R/ date 1 nov 1994
Paracetamol 500 mg
tab. da no. 20
S. 2 tabl. at least 20 min.
after the metoclopramide
Rp metoclopramide 10 mg
supp. da no. 5
S. one supp. as soon as an
attack is felt.
Ms/Mr Patient 31 B. Who
address:
age:

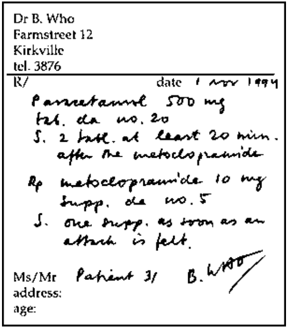
OCR + NLP

Prescription

what	**when**	**quantity**
paracetamol	anytime	500 mg
meboclaprauicle??	during attack	10 mg

Example: Optical Character Recognition

- Hospital wants to digitize and store all doctors’ prescriptions


Dr B. Who
Farmstreet 12
Kirkville
tel. 3876
R/ date 1 nov 1994
Paracetamol 500 mg
tab. da no. 20
S. 2 tabl. at least 20 min.
after the metoclopramide
Rp metoclopramide 10 mg
supp. da no. 5
S. one supp. as soon as an
attack is felt.
Ms/Mr Patient 3/ B. Who
address:
age:

OCR + NLP

Prescription

what	when	quantity
paracetamol	anytime	500 mg
meboclopprauicle??	during attack	10 mg
supp??	??	5

Why is it difficult?

- You are invited to a PhD defense

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Allergies

person	ingredient
Billis	milk
Billis	shrimps
Bernard	eggs
⋮	⋮

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Allergies

person	**ingredient**
Billis	milk
Billis	shrimps
Bernard	eggs
⋮	⋮

- You know what will be served

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Allergies

person	**ingredient**
Billis	milk
Billis	shrimps
Bernard	eggs
⋮	⋮

- You know what will be served

dishes
tiramisu
flapjacks
couscous
kougelhopf
⋮

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Allergies

person	**ingredient**
Billis	milk
Billis	shrimps
Bernard	eggs
⋮	⋮

- You know what will be served

dishes
tiramisu
flapjacks
couscous
kougelhopf
⋮

- But you can't ask the candidate what the ingredients are (he might be too busy giving the presentation)

Why is it difficult?

- You are invited to a PhD defense
- You have some allergies

Allergies

person	ingredient
Billis	milk
Billis	shrimps
Bernard	eggs
⋮	⋮

- You know what will be served

dishes
tiramisu
flapjacks
couscous
kougelhopf
⋮

- But you can’t ask the candidate what the ingredients are (he might be too busy giving the presentation)
- What are the chances that you’ll be allergic to his tiramisu?

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone

$\rightarrow$ sugar

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

→ mascarpone

→ sugar

→ strawberries

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

→ mascarpone

→ sugar

→ strawberries

→ shrimps

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone

$\rightarrow$ sugar

$\rightarrow$ strawberries

$\rightarrow$ shrimps

$\rightarrow$ coffee

$\rightarrow$ $\cdots$

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

→ mascarpone

→ sugar

→ strawberries

→ shrimps

→ coffee

→ …

So maybe the tiramisu's ingredients will be:

- mascarpone, sugar, eggs, and strawberries

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone
$\rightarrow$ sugar
$\rightarrow$ strawberries
$\rightarrow$ shrimps
$\rightarrow$ coffee
$\rightarrow$ $\dots$

So maybe the tiramisu's ingredients will be:

- mascarpone, sugar, eggs, and strawberries
- or: sugar, shrimps, coffee, and potatoes

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone
$\rightarrow$ sugar
$\rightarrow$ strawberries
$\rightarrow$ shrimps
$\rightarrow$ coffee
$\rightarrow$ ...

So maybe the tiramisu's ingredients will be:

- mascarpone, sugar, eggs, and strawberries
- or: sugar, shrimps, coffee, and potatoes
- or: shrimps and mascarpone
- ...

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

→ mascarpone
→ sugar
→ strawberries
→ shrimps
→ coffee
→ ...

So maybe the tiramisu's ingredients will be:

- mascarpone, sugar, eggs, and strawberries
- or: sugar, shrimps, coffee, and potatoes
- or: shrimps and mascarpone
- ...

- 20 different ingredients $\rightarrow 2^{20} \approx 1$ million possible recipes

Why is it difficult?

- Gather tiramisu recipes from books or from the web and make a list of possible ingredients

$\rightarrow$ mascarpone

$\rightarrow$ sugar

$\rightarrow$ strawberries

$\rightarrow$ shrimps

$\rightarrow$ coffee

$\rightarrow$ ...

So maybe the tiramisu's ingredients will be:

- mascarpone, sugar, eggs, and strawberries
- or: sugar, shrimps, coffee, and potatoes
- or: shrimps and mascarpone
- ...

- 20 different ingredients $\rightarrow 2^{20} \approx 1$ million possible recipes
- Real-world databases: more like 2^{1000} $\rightarrow$ **way too big**

Probabilistic Databases

- Need a framework to efficiently model this uncertainty and reason about it

- Need a framework to efficiently model this uncertainty and reason about it

$\rightarrow$ Probabilistic Databases

- In this thesis: tuple independent databases (**TID**)

$\rightarrow$ Idea: assume independence across tuples

Tuple-independent databases (TID)

- Succinctly represent probabilistic data:
  - A relational database D
  - A probability valuation π mapping each fact of D to $[0, 1]$

Tuple-independent databases (TID)

- Succinctly represent probabilistic data:
  - A relational database D
  - A probability valuation π mapping each fact of D to $[0, 1]$
- Semantics of a TID (D, π): a probability distribution on $D' \subseteq D$:
  - Each fact $F \in D$ is either present or absent with probability $\pi(F)$
  - Assume independence across facts

Tuple-independent databases (TID)

- Succinctly represent probabilistic data:
  - A relational database D
  - A probability valuation π mapping each fact of D to $[0, 1]$
- Semantics of a TID (D, π): a probability distribution on $D' \subseteq D$:
  - Each fact $F \in D$ is either present or absent with probability $\pi(F)$
  - Assume independence across facts

$\rightarrow$ For $D' \subseteq D$, $\Pr(D') = (\prod_{F \in D'} \pi(F)) \times (\prod_{F \in D \setminus D'} (1 - \pi(F)))$

Example: TID

$D =$

Contains	
tiramisu	sugar
tiramisu	eggs

Example: TID

$D =$

C	
t	s
t	e

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

Example: TID

$(D, \pi) =$

C		
t	*s*	.5
t	*e*	.2

$.5 \times .2$

C	
t	*s*
t	*e*

Example: TID

$(D, \pi) =$

C		
t	*s*	.5
t	*e*	.2

$.5 \times .2$

C	
t	*s*
t	*e*

$.5 \times (1 - .2)$

C	
t	*s*

Example: TID

$(D, \pi) =$

C		
t	s	$.5$
t	e	$.2$

$.5 \times .2$

C	
t	s
t	e

$.5 \times (1 - .2)$

C	
t	s

$(1 - .5) \times .2$

C	
t	e

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$.5 \times .2$

C	
t	s
t	e

$.5 \times (1 - .2)$

C	
t	s

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

Example: TID

$(D, \pi) =$

C		
t	s	$.5$
t	e	$.2$

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

$.5 \times (1 - .2)$

C	
t	s

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

$\Pr(D \models q) =$

Example: TID

$(D, \pi) =$

C		
t	s	$.5$
t	e	$.2$

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

$.5 \times (1 - .2)$

C	
t	s

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

$\Pr(D \models q) =$

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

✓

$.5 \times (1 - .2)$

C	
t	s

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

$\Pr(D \models q) = .5 \times .2$

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

✓

$.5 \times (1 - .2)$

C	
t	s

✓

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

$\Pr(D \models q) = .5 \times .2$

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

✓

$.5 \times (1 - .2)$

C	
t	s

✓

$(1 - .5) \times .2$

C	
t	e

$(1 - .5) \times (1 - .2)$

C

$$\Pr(D \models q) = .5 \times .2 + .5 \times (1 - .2)$$

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	s
t	e

✔

$.5 \times (1 - .2)$

C	
t	s

✔

$(1 - .5) \times .2$

C	
t	e

✔

$(1 - .5) \times (1 - .2)$

C

$\Pr(D \models q) = .5 \times .2 + .5 \times (1 - .2)$

Example: TID

$$(D, \pi) = $$

C		
t	s	$.5$
t	e	$.2$

$$q = \exists x\, y\; C(x, y)$$

$.5 \times .2$		$.5 \times (1 - .2)$		$(1 - .5) \times .2$		$(1 - .5) \times (1 - .2)$
C		C		C		C
t	s	t	s			
t	e			t	e	
✔		✔		✔		

$$\Pr(D \models q) = .5 \times .2 + .5 \times (1 - .2) + (1 - .5) \times .2$$

Example: TID

$(D, \pi) =$

C		
t	s	.5
t	e	.2

$q = \exists x\, y\, C(x, y)$

$.5 \times .2$	
C	
t	s
t	e
✓	

$.5 \times (1 - .2)$	
C	
t	s
✓	

$(1 - .5) \times .2$	
C	
t	e
✓	

$(1 - .5) \times (1 - .2)$
C
✗

$\Pr(D \models q) = .5 \times .2 + .5 \times (1 - .2) + (1 - .5) \times .2$

Example: TID

$(D, \pi) =$

C		
t	*s*	.5
t	*e*	.2

$q = \exists x\, y\; C(x, y)$

$.5 \times .2$

C	
t	*s*
t	*e*

✓

$.5 \times (1 - .2)$

C	
t	*s*

✓

$(1 - .5) \times .2$

C	
t	*e*

✓

$(1 - .5) \times (1 - .2)$

C

✗

$$\Pr(D \models q) = .5 \times .2 + .5 \times (1 - .2) + (1 - .5) \times .2$$
$$= 1 - [(1 - .5) \times (1 - .2)]$$

Probabilistic query evaluation (PQE)

Let us fix:

- Class $\mathcal{D}$ of relational databases (e.g., acyclic, trees)
- Class $\mathcal{Q}$ of Boolean queries (e.g., paths, trees)

Probabilistic query evaluation (PQE)

Let us fix:

- Class $\mathcal{D}$ of relational databases (e.g., acyclic, trees)
- Class $\mathcal{Q}$ of Boolean queries (e.g., paths, trees)

Probabilistic query evaluation (PQE) problem for $\mathcal{Q}$ and $\mathcal{D}$:

- Given a query $q \in \mathcal{Q}$
- Given a database $D \in \mathcal{D}$ and a probability valuation π
- Compute the probability that (D, π) satisfies q

Probabilistic query evaluation (PQE)

Let us fix:

- Class $\mathcal{D}$ of relational databases (e.g., acyclic, trees)
- Class $\mathcal{Q}$ of Boolean queries (e.g., paths, trees)

Probabilistic query evaluation (PQE) problem for $\mathcal{Q}$ and $\mathcal{D}$:

- Given a query $q \in \mathcal{Q}$
- Given a database $D \in \mathcal{D}$ and a probability valuation π
- Compute the probability that (D, π) satisfies q

$\rightarrow$ $\Pr((D, \pi) \models q) = \sum_{D' \subseteq D,\ D' \models q} \Pr(D')$

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) **complexity** of PQE depending on the class $\mathcal{D}$ of databases and class $\mathcal{Q}$ of queries?

Complexity of probabilistic query evaluation (PQE)

Question: what is the (data, combined) **complexity** of PQE depending on the class $\mathcal{D}$ of databases and class $\mathcal{Q}$ of queries?

Wish list:

- PQE tractable in combined complexity
- **or** PQE tractable in the data, reasonable in the query

Data complexity results: related work (1/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)

Data complexity results: related work (1/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)

→ q is 'safe' $\implies$ PQE is PTIME

Data complexity results: related work (1/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ ($\approx$ SQL with **select**, **where**, **union**)

→ q is 'safe' $\implies$ PQE is PTIME

→ q is not 'safe' $\implies$ PQE is **#P-hard**

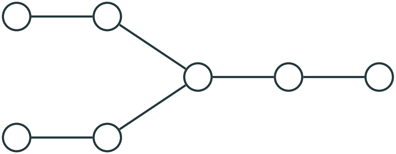
Data complexity results: related work (1/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)

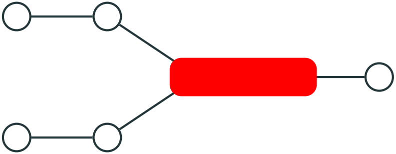
  → q is 'safe' $\Longrightarrow$ PQE is PTIME

  → q is not 'safe' $\Longrightarrow$ PQE is **#P-hard**
- Existing data dichotomy result on data

Treewidth by example:

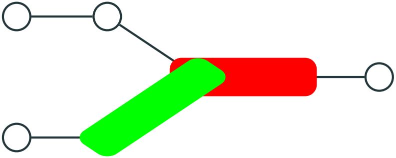


Treewidth by example:



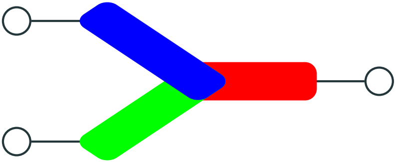
Treewidth

Treewidth by example:

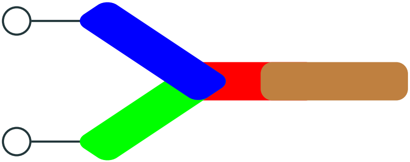


Treewidth

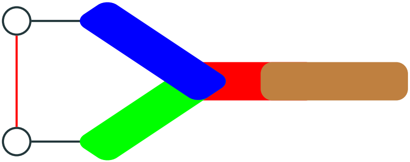
Treewidth by example:



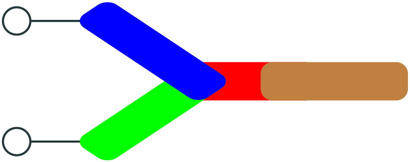
Treewidth by example:

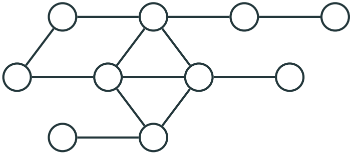


Treewidth by example:

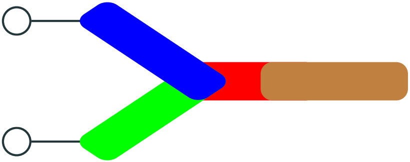


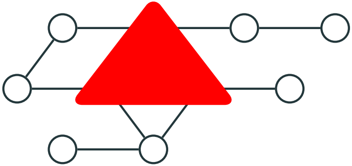
Treewidth by example:





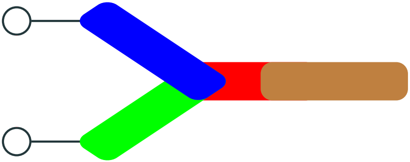
Treewidth by example:

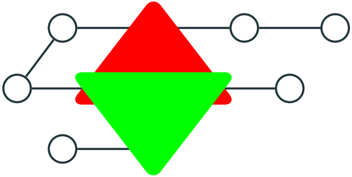




Treewidth

Treewidth by example:



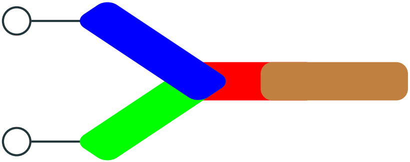


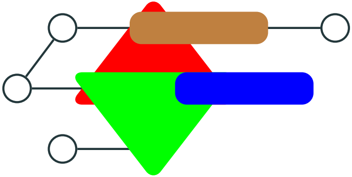
Treewidth

Treewidth by example:

Treewidth

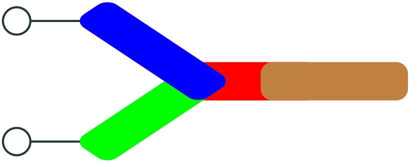
Treewidth by example:

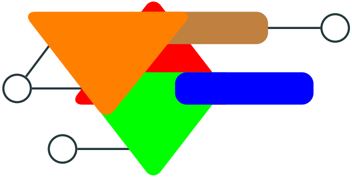




Treewidth

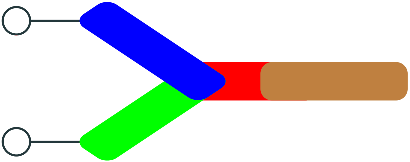
Treewidth by example:

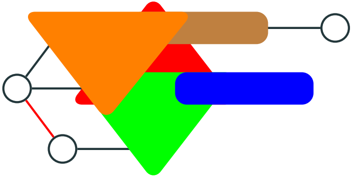




Treewidth

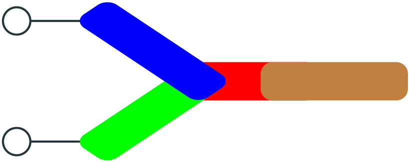
Treewidth by example:

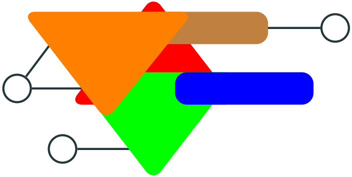




Treewidth

Treewidth by example:





Treewidth

Treewidth by example:

- Trees have treewidth 1
- Cycles have treewidth 2
- k-cliques and $(k-1)$-grids have treewidth $k-1$

Treewidth

Treewidth by example:

- Trees have treewidth 1
- Cycles have treewidth 2
- k-cliques and $(k-1)$-grids have treewidth $k-1$

$\rightarrow$ **Treelike**: the treewidth is **bounded by a constant**

Data complexity results: related work (2/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)
  - → q is 'safe' $\implies$ PQE is PTIME
  - → q is not 'safe' $\implies$ PQE is **#P-hard**
- Existing data dichotomy result on data

Data complexity results: related work (2/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)
  - → q is 'safe' $\implies$ PQE is PTIME
  - → q is not 'safe' $\implies$ PQE is **#P-hard**

- Existing data dichotomy result on data
  - $\mathcal{D}_k$ = all databases whose treewidth is bounded by $k \in \mathbb{N}$

Data complexity results: related work (2/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)

→ q is 'safe' $\implies$ PQE is PTIME

→ q is not 'safe' $\implies$ PQE is **#P-hard**

- Existing data dichotomy result on data
  - $\mathcal{D}_k$ = all databases whose treewidth is bounded by $k \in \mathbb{N}$
  - $\mathcal{Q} = \{q\}$, for q a MSO query

Data complexity results: related work (2/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ ($\approx$ SQL with **select**, **where**, **union**)

  $\rightarrow$ q is 'safe' $\implies$ PQE is PTIME

  $\rightarrow$ q is not 'safe' $\implies$ PQE is **#P-hard**

- Existing data dichotomy result on data
  - $\mathcal{D}_k$ = all databases whose treewidth is bounded by $k \in \mathbb{N}$
  - $\mathcal{Q} = \{q\}$, for q a MSO query

  $\rightarrow$ PQE has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]

Data complexity results: related work (2/2)

- Existing data dichotomy result on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)
  - → q is 'safe' $\implies$ PQE is PTIME
  - → q is not 'safe' $\implies$ PQE is **#P-hard**

- Existing data dichotomy result on data
  - $\mathcal{D}_k$ = all databases whose treewidth is bounded by $k \in \mathbb{N}$
  - $\mathcal{Q} = \{q\}$, for q a MSO query
  - → PQE has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - → There is an FO query for which PQE is **#P-hard** on any unbounded-treewidth database class $\mathcal{D}$ (under some assumptions) [Amarilli, Bourhis, & Senellart, 2016]

Data complexity results: related work (2/2)

- Existing **data dichotomy result** on queries [Dalvi & Suciu, 2012]
  - $\mathcal{D}_{\text{all}}$ = all possible databases
  - $\mathcal{Q} = \{q\}$, for q a UCQ (≈ SQL with **select**, **where**, **union**)
  - → q is 'safe' $\implies$ PQE is PTIME
  - → q is not 'safe' $\implies$ PQE is **#P-hard**

- Existing **data dichotomy result** on data
  - $\mathcal{D}_k$ = all databases whose treewidth is bounded by $k \in \mathbb{N}$
  - $\mathcal{Q} = \{q\}$, for q a MSO query
  - → PQE has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - → There is an FO query for which PQE is **#P-hard** on any unbounded-treewidth database class $\mathcal{D}$ (under some assumptions) [Amarilli, Bourhis, & Senellart, 2016]

What about combined complexity?

Plan

During my thesis I have investigated:

1. Combined complexity of PQE for conjunctive queries on binary signatures (graph databases)
   $\rightarrow$ **PODS'2017** (with A. Amarilli and P. Senellart)

Plan

During my thesis I have investigated:

1. Combined complexity of PQE for conjunctive queries on binary signatures (graph databases)
   - $\rightarrow$ **PODS'2017** (with A. Amarilli and P. Senellart)
2. Combined complexity of non-probabilistic query evaluation on treelike databases
   - $\rightarrow$ **ICDT'2017** (with A. Amarilli, P. Bourhis, and P. Senellart)

Plan

During my thesis I have investigated:

1. Combined complexity of PQE for conjunctive queries on binary signatures (graph databases)
   - → **PODS'2017** (with A. Amarilli and P. Senellart)
2. Combined complexity of non-probabilistic query evaluation on treelike databases
   - → **ICDT'2017** (with A. Amarilli, P. Bourhis, and P. Senellart)
3. Connecting width and semantics in knowledge compilation, and applications to PQE
   - → **ICDT'2018** (with A. Amarilli and P. Senellart)

Plan

During my thesis I have investigated:

1. Combined complexity of PQE for conjunctive queries on binary signatures (graph databases)
   → **PODS'2017** (with A. Amarilli and P. Senellart)
2. Combined complexity of non-probabilistic query evaluation on treelike databases
   → **ICDT'2017** (with A. Amarilli, P. Bourhis, and P. Senellart)
3. Connecting width and semantics in knowledge compilation, and applications to PQE
   → **ICDT'2018** (with A. Amarilli and P. Senellart)

- Connections between safe queries and circuit classes from knowledge compilation
  → **AMW'2018** (with D. Olteanu)

PQE of conjunctive queries on binary signatures

Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x, y) \wedge S(y, z) \wedge S(t, z)$

R		
b	c	.8
c	a	.1
c	d	.1

S		
a	b	1.
d	b	.05

Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x,y) \wedge S(y,z) \wedge S(t,z) \quad \rightarrow \quad x \xrightarrow{R} y \xrightarrow{S} z \xleftarrow{S} t$

R		
b	*c*	.8
c	*a*	.1
c	*d*	.1

S		
a	*b*	1.
d	*b*	.05

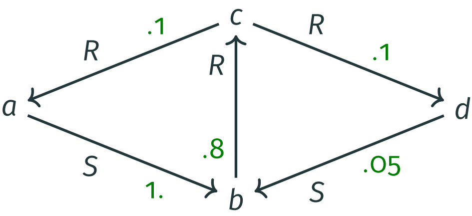
Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x, y) \wedge S(y, z) \wedge S(t, z)$ $\rightarrow$

R		
b	*c*	.8
c	*a*	.1
c	*d*	.1

S		
a	*b*	1.
d	*b*	.05

$\rightarrow$



Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x, y) \land S(y, z) \land S(t, z)$ $\rightarrow$

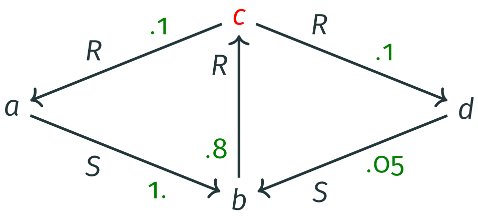
R

b	*c*	.8
c	*a*	.1
c	*d*	.1

S

a	*b*	1.
d	*b*	.05

$\rightarrow$



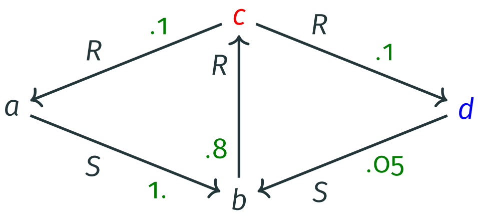
Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x, y) \wedge S(y, z) \wedge S(t, z)$ $\rightarrow$

R		
b	c	.8
c	a	.1
c	d	.1

S		
a	b	1.
d	b	.05

$\rightarrow$



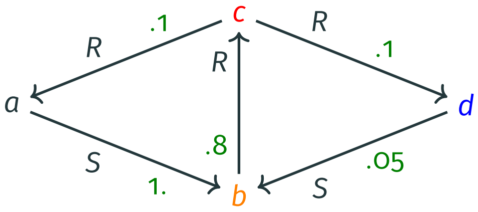
Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x,y) \wedge S(y,z) \wedge S(t,z)$ $\rightarrow$

R		
b	c	.8
c	a	.1
c	d	.1

S		
a	b	1.
d	b	.05

$\rightarrow$



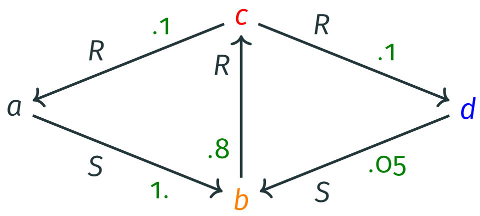
Restrict to CQs on binary signatures

$\exists x\, y\, z\, t\; R(x,y) \wedge S(y,z) \wedge S(t,z)$ $\rightarrow$

R		
b	*c*	.8
c	*a*	.1
c	*d*	.1

S		
a	*b*	1.
d	*b*	.05

$\rightarrow$



Restrict instances to trees

$\mathcal{Q} =$ one-way paths (1WP), $\mathcal{D} =$ polytrees (PT)

Restrict instances to trees

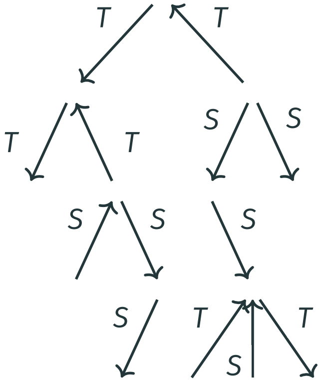
$\mathcal{Q}$ = one-way paths (1WP), $\mathcal{D}$ = polytrees (PT)

Q: $\xrightarrow{T} \xrightarrow{S} \xrightarrow{S} \xrightarrow{S} \xrightarrow{T}$

Restrict instances to trees

$\mathcal{Q}$ = one-way paths (1WP), $\mathcal{D}$ = polytrees (PT)

I:



Q:

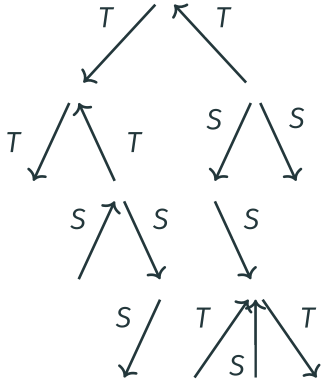
+ prob. for each edge

Restrict instances to trees

$\mathcal{Q}$ = one-way paths (1WP), $\mathcal{D}$ = polytrees (PT)

Q:

I:



+ prob. for each edge

Proposition

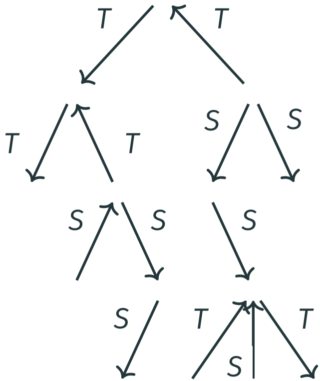
PQE of 1WP *on* PT *is* ***#P-hard***

$\mathcal{Q}$ = *one-way paths*, $\mathcal{D}$ = *polytrees*, without labels

- What if we do not have labels?

Q: $\xrightarrow{T}\ \xrightarrow{S}\ \xrightarrow{S}\ \xrightarrow{S}\ \xrightarrow{T}$

I:

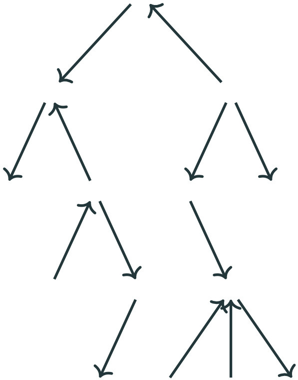


$\mathcal{Q} =$ *one-way paths*, $\mathcal{D} =$ *polytrees*, without labels

- What if we do not have labels?

Q:

I:

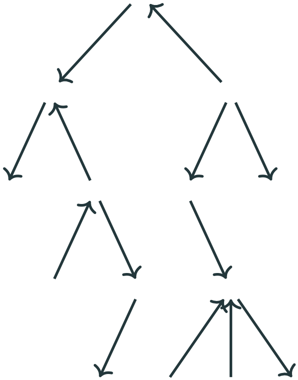


$\mathcal{Q}$ = *one-way paths*, $\mathcal{D}$ = *polytrees*, without labels

- What if we do not have labels?
- Probability that the data graph has a path of length $|Q|$

Q:

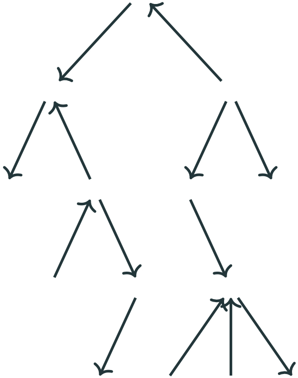
I:



$\mathcal{Q}$ = *one-way paths*, $\mathcal{D}$ = *polytrees*, without labels

- What if we do not have labels?
- Probability that the data graph has a path of length $|Q|$
- Computed bottom-up, e.g., tree automaton

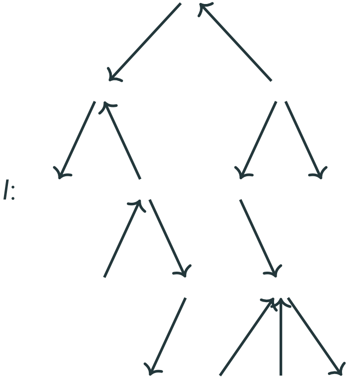
I:



Q:

$\mathcal{Q}$ = *one-way paths*, $\mathcal{D}$ = *polytrees*, without labels

- What if we do not have labels?
- Probability that the data graph has a path of length $|Q|$
- Computed bottom-up, e.g., tree automaton



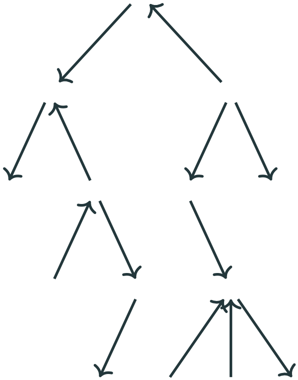
Proposition

PQE of unlabeled 1WP *on* PT *is PTIME*

$\mathcal{Q}$ = *one-way paths*, $\mathcal{D}$ = *polytrees*, without labels

- What if we do not have labels?
- Probability that the data graph has a path of length $|Q|$
- Computed bottom-up, e.g., tree automaton
- Labels have an impact!

I:



Q:

Proposition

PQE of unlabeled 1WP *on* PT *is PTIME*

Our graph classes

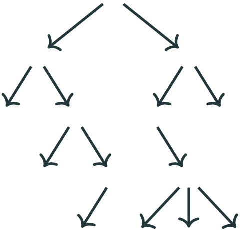
1WP

$\xrightarrow{R} \xrightarrow{S} \xrightarrow{S} \xrightarrow{T}$

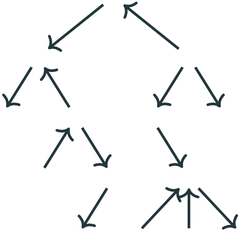
2WP

$\xrightarrow{R} \xleftarrow{S} \xleftarrow{S} \xrightarrow{T} \xleftarrow{R}$

DWT



PT



$$\text{1WP} \subseteq \text{2WP} \subseteq \text{PT}, \quad \text{1WP} \subseteq \text{DWT} \subseteq \text{PT}$$

$$\text{PT} \subseteq \text{Connected} \subseteq \text{All}$$

Results

$\downarrow Q$ $\quad D\rightarrow$	1WP	2WP	DWT	PT	Connected
1WP					
2WP					
DWT		PTIME			
PT					#P-hard
Connected					

$\geqslant 2$ labels

Results

$\downarrow Q$ $D \rightarrow$	1WP	2WP	DWT	PT	Connected
1WP					
2WP					
DWT		PTIME			
PT					#P-hard
Connected					

$\geqslant 2$ labels

$\downarrow Q$ $D \rightarrow$	1WP	2WP	DWT	PT	Connected
1WP					
2WP					
DWT		PTIME			
PT					#P-hard
Connected					

No labels

Summary

- Detailed study of the combined complexity of PQE
- Showed the importance of various features on the problem: labels, global orientation, branching, connectedness
- Established the complexity for all combinations of the graph classes we considered
- Essentially, all tractable cases involve paths

Summary

- Detailed study of the combined complexity of PQE
- Showed the importance of various features on the problem: labels, global orientation, branching, connectedness
- Established the complexity for all combinations of the graph classes we considered
- Essentially, all tractable cases involve paths

Drawbacks:

- Our graph classes may seem "arbitrary"
- Not yet a dichotomy, just starting to understand the problem
- Tractable cases very restricted

Lowering our expectations

What if we want the complexity to be:

- Tractable in the data
- **Not *too* horrible** in the **query**

Can we then support a **more expressive query language**?
(e.g., disjunctions, negations, recursion)

Non-probabilistic query evaluation on treelike databases

Starting point

- Existing data dichotomy result on data

Starting point

- Existing data dichotomy result on data
  - $\rightarrow$ PQE for MSO on bounded-treewidth databases has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]

Starting point

- Existing data dichotomy result on data
  - → PQE for MSO on bounded-treewidth databases has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - Problem: nonelementary in the query $\left.2^{2^{\cdot^{\cdot^{\cdot^{|Q|}}}}}\right\}|Q|$

Starting point

- Existing data dichotomy result on data
  - → PQE for MSO on bounded-treewidth databases has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - Problem: nonelementary in the query $\left.2^{2^{\cdot^{\cdot^{\cdot^{|Q|}}}}}\right\}|Q|$

The database class is parameterized

Starting point

- Existing data dichotomy result on data
  - → PQE for MSO on bounded-treewidth databases has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - Problem: nonelementary in the query $\left.2^{2^{\cdot^{\cdot^{\cdot^{|Q|}}}}}\right\}|Q|$

The database class is parameterized

Idea:

- one parameter for the databases

Starting point

- Existing data dichotomy result on data
  → PQE for MSO on bounded-treewidth databases has linear data complexity [Amarilli, Bourhis, & Senellart, 2015]
  - Problem: nonelementary in the query $\left.2^{2^{\cdot^{\cdot^{\cdot^{|Q|}}}}}\right\}|Q|$

The database class is parameterized

Idea:

- one parameter for the databases
- and one parameter for the queries

Parameterized Complexity

Idea: one parameter k_D for the database (treewidth) AND one parameter k_Q for the query

Parameterized Complexity

Idea: one parameter k_D for the database (treewidth) AND one parameter k_Q for the query

- **Database** classes $\mathcal{D}_1, \mathcal{D}_2, \cdots$

Parameterized Complexity

Idea: one parameter k_D for the database (treewidth) AND one parameter k_Q for the query

- **Database** classes $\mathcal{D}_1, \mathcal{D}_2, \cdots$
- **Query** classes $\mathcal{Q}_1, \mathcal{Q}_2, \cdots$

Parameterized Complexity

Idea: one parameter k_D for the database (treewidth) AND one parameter k_Q for the query

- **Database** classes $\mathcal{D}_1, \mathcal{D}_2, \cdots$
- **Query** classes $\mathcal{Q}_1, \mathcal{Q}_2, \cdots$

Definition

The problem is *fixed-parameter tractable (FPT) linear* if there exists a computable function f such that it can be solved in time $f(k_D, k_Q) \times |Q| \times |D|$

Results

1) A new language...

- We introduce the language of ***clique-frontier-guarded Datalog*** (CFG-Datalog), parameterized by *body-size* k_P

Results

1) A new language...

- We introduce the language of ***clique-frontier-guarded Datalog*** (CFG-Datalog), parameterized by *body-size* k_P

2) ... with **FPT-linear** (combined) evaluation...

- Given a CFG-Datalog program P with body-size k_P and a relational database D of treewidth k_D, checking if $D \models P$ can be done in time $f(k_P, k_D) \times |P| \times |D|$

Results

1) A new language...

- We introduce the language of ***clique-frontier-guarded Datalog*** (CFG-Datalog), parameterized by *body-size* k_P

2) ... with **FPT-linear** (combined) evaluation...

- Given a CFG-Datalog program P with body-size k_P and a relational database D of treewidth k_D, checking if $D \models P$ can be done in time $f(k_P, k_D) \times |P| \times |D|$

3) ... and also **FPT-linear** (combined) computation of provenance

- We design a new concise provenance representation based on cyclic Boolean circuits: ***cycluits***

Proof Sketch

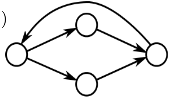
CFG-Datalog program P
of body-size ≤ k_P

1 C(x): Subway('Corvisart', x)
C(x): C(y) AND Subway(y, x)

2 Goal(): NOT C("Châtelet")

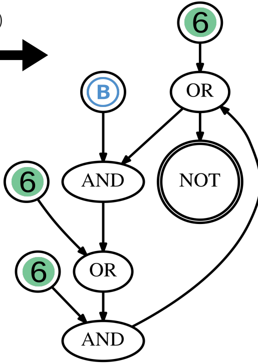
O(g′(k_P,k_D) |P|)

Two-way Alternating
Tree Automaton A



O(|A| |E|)

Provenance Cycluit

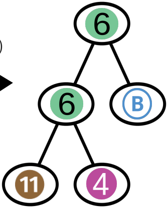


Database D
of treewidth ≤ k_D

(Paris Metro map)

O(g(k_D) |D|)

Tree encoding E



"Under which conditions is it impossible to go from station Corvisart to station Châtelet with the subway?"

Summary

Theorem

Given a CFG-Datalog program P with body-size k_P and a relational database D of treewidth k_D, we can compute a cycluit representing the ***provenance*** *of P on D in time $f(k_P, k_D) \times |P| \times |D|$*

Summary

Theorem

Given a CFG-Datalog program P with body-size k_P and a relational database D of treewidth k_D, we can compute a cycluit representing the ***provenance*** *of P on D in time $f(k_P, k_D) \times |P| \times |D|$*

- Capture interesting query languages such as 2RPQs (graph query language), conjunctive queries (of bounded simplicial treewidth), guarded negation logic fragments, etc.

Summary

Theorem

Given a CFG-Datalog program P with body-size k_P and a relational database D of treewidth k_D, we can compute a cycluit representing the ***provenance*** *of P on D in time $f(k_P, k_D) \times |P| \times |D|$*

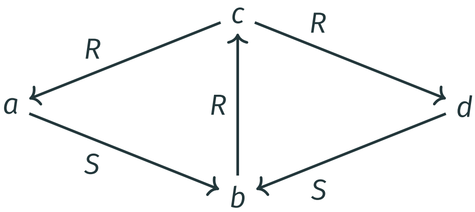
- Capture interesting query languages such as 2RPQs (graph query language), conjunctive queries (of bounded simplicial treewidth), guarded negation logic fragments, etc.

Can we lift this result to **probabilistic** evaluation?

Boolean cycluits to d-SDNNFs and lower bounds

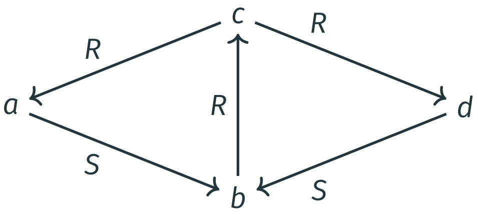
Example: Provenance

$\exists x\, y\, z\; (R(x, y) \wedge S(y, z)) \vee (S(x, y) \wedge R(y, z))$



Example: Provenance

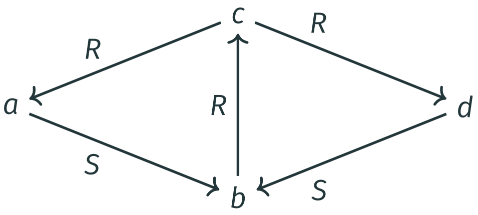
$\exists x\, y\, z\ (R(x,y) \land S(y,z)) \lor (S(x,y) \land R(y,z))$

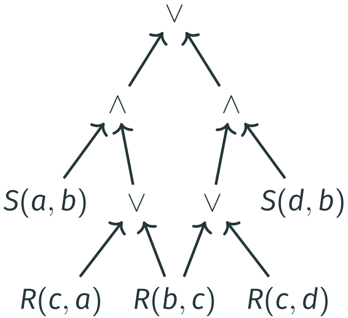


$$\text{Prov}(q,D) = [S(a,b) \land (R(b,c) \lor R(c,a))]$$
$$\lor [S(d,b) \land (R(b,c) \lor R(c,d))]$$

Example: Provenance

$\exists x\, y\, z\ (R(x, y) \wedge S(y, z)) \vee (S(x, y) \wedge R(y, z))$





$$\begin{aligned}\mathrm{Prov}(q, D) = {} & [S(a, b) \wedge (R(b, c) \vee R(c, a))] \\ & \vee [S(d, b) \wedge (R(b, c) \vee R(c, d))]\end{aligned}$$

Hardness of probability computation

→ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

Hardness of probability computation

→ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

→ Which restrictions on φ or C make this possible?

Hardness of probability computation

$\rightarrow$ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

$\rightarrow$ Which restrictions on φ or C make this possible?

- Knowledge compilation studies which classes of circuits ensure tractability of...
  - **SAT**

Hardness of probability computation

→ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

→ Which restrictions on φ or C make this possible?

- Knowledge compilation studies which classes of circuits ensure tractability of...
    - **SAT**
    - **#SAT**

Hardness of probability computation

→ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

→ Which restrictions on φ or C make this possible?

- Knowledge compilation studies which classes of circuits ensure tractability of...
  - **SAT**
  - **#SAT**
  - **probability computation**

Hardness of probability computation

→ Computing the probability of a Boolean formula φ (or circuit C) is **#P-hard**! (since **#SAT** is **#P-hard**)

→ Which restrictions on φ or C make this possible?

- Knowledge compilation studies which classes of circuits ensure tractability of...
  - **SAT**
  - **#SAT**
  - **probability computation**
  - ...

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  $\rightarrow$ message passing algorithm for #SAT and probabilistic evaluation

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  $\rightarrow$ message passing algorithm for #SAT and probabilistic evaluation
  - Links with **Bayesian networks**

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  - $\rightarrow$ message passing algorithm for #SAT and probabilistic evaluation
    - Links with **Bayesian networks**

Semantics-based:

- **Ordered Binary Decision Diagrams** (OBDDs)/ **Deterministic Structured Decomposable Negation Normal Forms** (d-SDNNFs)

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  - → message passing algorithm for #SAT and probabilistic evaluation
    - Links with **Bayesian networks**

Semantics-based:

- **Ordered Binary Decision Diagrams** (OBDDs)/ **Deterministic Structured Decomposable Negation Normal Forms** (d-SDNNFs)
  - → #SAT and probabilistic evaluation are easy because these classes have strong semantic constraints

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  - → message passing algorithm for #SAT and probabilistic evaluation
    - Links with **Bayesian networks**

Semantics-based:

- **Ordered Binary Decision Diagrams** (OBDDs)/ **Deterministic Structured Decomposable Negation Normal Forms** (d-SDNNFs)
  - → #SAT and probabilistic evaluation are easy because these classes have strong semantic constraints
    - Used to understand **#SAT solvers**

Target classes in knowledge compilation

For #SAT and probabilistic evaluation, two main restrictions on circuits:

Width-based:

- Bounded **pathwidth**/**treewidth** Boolean circuits, CNFs, DNFs, etc.
  → message passing algorithm for #SAT and probabilistic evaluation
  - Links with **Bayesian networks**

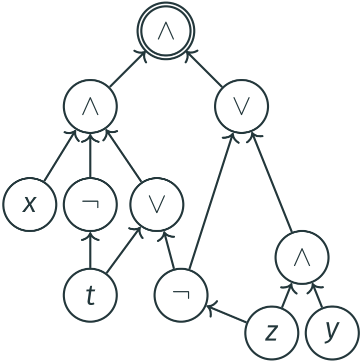
Semantics-based:

- **Ordered Binary Decision Diagrams** (OBDDs)/ **Deterministic Structured Decomposable Negation Normal Forms** (d-SDNNFs)
  → #SAT and probabilistic evaluation are easy because these classes have strong semantic constraints
  - Used to understand **#SAT solvers**

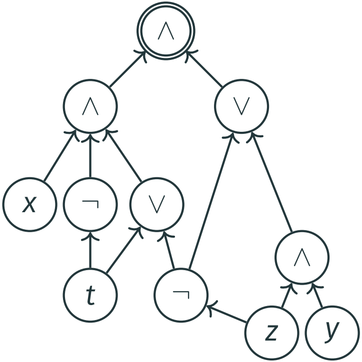
Question: what are the links between the two?

Treewidth and **d-SDNNFs**

Bounded treewidth Boolean circuits

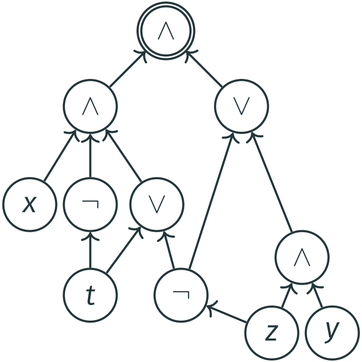


Bounded treewidth Boolean circuits



Treewidth of C = that of the underlying graph

Bounded treewidth Boolean circuits



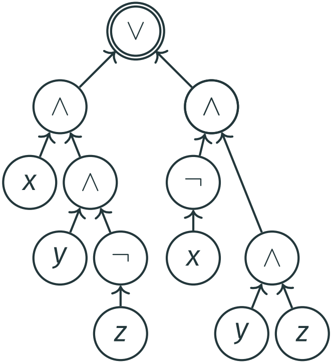
Treewidth of C = that of the underlying graph

We can do message passing:

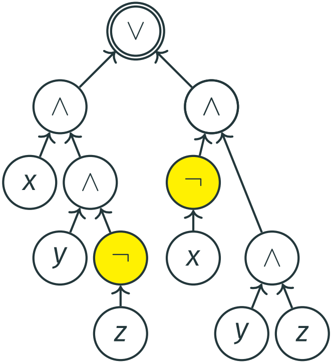
Theorem (Lauritzen & Spielgelhalter, 1988)

Fix $k \in \mathbb{N}$. Given a Boolean circuit C of treewidth $\leqslant k$, we can compute its probability in time $O(f(k) \times |C|)$, where f is singly exponential

d-SDNNF

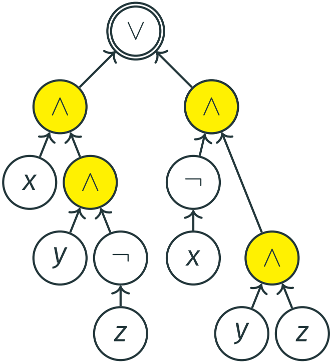


d-SDNNF



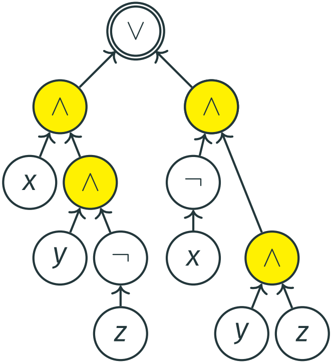
- Negation Normal Form: negations only applied to the leaves

d-SDNNF



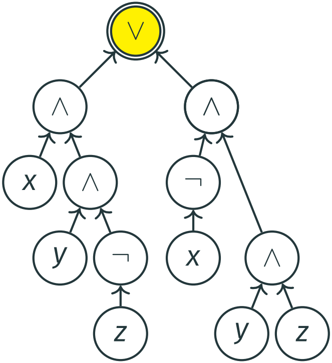
- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)

d-SDNNF



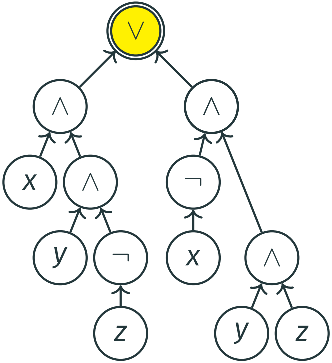
- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of $\wedge$-gates are **independent** (no variable has a path to two different inputs of the same $\wedge$-gate)
  - $\rightarrow$ **SAT** can be solved efficiently

d-SDNNF



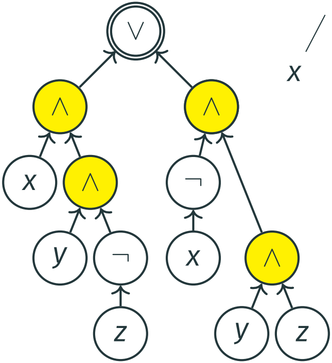
- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  → **SAT** can be solved efficiently
- Deterministic: inputs of ∨-gates are **mutually exclusive**

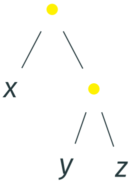
d-SDNNF



- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  - → **SAT** can be solved efficiently
- Deterministic: inputs of ∨-gates are **mutually exclusive**
  - → **#SAT** and **probability evaluation**

d-SDNNF





- Negation Normal Form: negations only applied to the leaves
- Decomposable: inputs of ∧-gates are **independent** (no variable has a path to two different inputs of the same ∧-gate)
  - → **SAT** can be solved efficiently
- Deterministic: inputs of ∨-gates are **mutually exclusive**
  - → **#SAT** and **probability evaluation**
- Structured: there is a **v-tree** that structures the ∧-gates

Treewidth and d-SDNNFs: Upper bound

Theorem

*Let C be a Boolean circuit of **treewidth** $\leqslant k$.*
***We can compute** a **d-SDNNF** equivalent to C in time $O(|C| \times f(k))$,*
*where f is **singly** exponential*

Treewidth and d-SDNNFs: Upper bound

Theorem

*Let C be a Boolean circuit of **treewidth** $\leqslant k$.*
***We can compute** a **d-SDNNF** equivalent to C in time $O(|C| \times f(k))$,*
*where f is **singly** exponential*

→ Recaptures message passing through the use of knowledge compilation

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*
- Arity: size of the largest clause

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*
- Arity: size of the largest clause
- Degree: maximal number of clauses to which a variable belongs

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*
- Arity: size of the largest clause
- Degree: maximal number of clauses to which a variable belongs

Theorem

*Let φ be a **monotone DNF** of **treewidth** k, let $a := \mathrm{arity}(\varphi)$ and $d := \mathrm{degree}(\varphi)$. Then any **d-SDNNF** for φ has size* $\geqslant 2^{\left\lfloor \frac{k}{3 \times a^3 \times d^2} \right\rfloor} - 1$

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*
- Arity: size of the largest clause
- Degree: maximal number of clauses to which a variable belongs

Theorem

*Let φ be a **monotone DNF** of **treewidth** k, let $a := \mathrm{arity}(\varphi)$ and $d := \mathrm{degree}(\varphi)$. Then any **d-SDNNF** for φ has size* $\geqslant 2^{\left\lfloor \frac{k}{3 \times a^3 \times d^2} \right\rfloor} - 1$

- For **CNFs**, the bound even works for (non-deterministic) **SDNNF**

Treewidth and d-SDNNFs: Lower bound

- Already applies to very restricted Boolean circuits: **monotone DNFs and CNFs**
- Treewidth of a DNF/CNF: that of its *Gaifman graph*
- Arity: size of the largest clause
- Degree: maximal number of clauses to which a variable belongs

Theorem

*Let φ be a **monotone DNF** of **treewidth** k, let $a := \mathrm{arity}(\varphi)$ and $d := \mathrm{degree}(\varphi)$. Then any **d-SDNNF** for φ has size* $\geqslant 2^{\left\lfloor \frac{k}{3 \times a^3 \times d^2} \right\rfloor} - 1$

- For **CNFs**, the bound even works for (non-deterministic) **SDNNF**
- The bound is generic: it applies to *any* monotone DNF/CNF

Proof Sketch for CNFs (1/2)

Use the connection made in [Bova, Capelli & Mengel, 2016] between the notion of combinatorial rectangle in **communication complexity** and SDNNFs.

Definition

A (X, Y)-rectangle is a Boolean function $R : 2^{X \cup Y} \to \{0, 1\}$ that can be written as $R_X \wedge R_Y$, for some Boolean functions $R_X : 2^X \to \{0, 1\}$ and $R_Y : 2^Y \to \{0, 1\}$. A (X, Y)-rectangle cover of a function $f : 2^{X \cup Y} \to \{0, 1\}$ is a set $\{R_1, \cdots, R_n\}$ of (X, Y)-rectangles such that $f \equiv \bigvee_{i=1}^{n} R_i$.

Proof Sketch for CNFs (1/2)

Use the connection made in [Bova, Capelli & Mengel, 2016] between the notion of combinatorial rectangle in **communication complexity** and SDNNFs.

Definition

A (X, Y)-rectangle is a Boolean function $R : 2^{X \cup Y} \to \{0, 1\}$ that can be written as $R_X \wedge R_Y$, for some Boolean functions $R_X : 2^X \to \{0, 1\}$ and $R_Y : 2^Y \to \{0, 1\}$. A (X, Y)-rectangle cover of a function $f : 2^{X \cup Y} \to \{0, 1\}$ is a set $\{R_1, \cdots, R_n\}$ of (X, Y)-rectangles such that $f \equiv \bigvee_{i=1}^{n} R_i$.

Theorem (Bova, Capelli & Mengel, 2016)

Let C be an SDNNF computing a function φ on variables V, structured by a v-tree T. Let $n \in T$, and let (X, Y) be the partition of V that n induces. Then φ has a (X, Y)-rectangle cover of size at most $|C|$.

Proof Sketch for CNFs (2/2)

A CNF having no small rectangle cover:

Theorem (Sherstov, 2014)
Let $X = \{x_1, \ldots, x_n\}$ *and* $Y = \{y_1, \ldots, y_n\}$ *be two disjoint sets of variables. Then any* (X, Y)*-rectangle cover of the Boolean function* $\mathrm{SCOV}_n(X, Y) := \bigwedge_{i=1}^{n} x_i \vee y_i$ *has size* $\geqslant 2^n$.

Proof Sketch for CNFs (2/2)

A CNF having no small rectangle cover:

Theorem (Sherstov, 2014)
Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ be two disjoint sets of variables. Then any (X, Y)-rectangle cover of the Boolean function $\mathrm{SCOV}_n(X, Y) := \bigwedge_{i=1}^{n} x_i \vee y_i$ has size $\geqslant 2^n$.

We show that we can find $\mathrm{SCOV}_{\frac{k}{3\times a^3\times d^2}}(X, Y)$ within any CNF of treewidth $\geqslant k$.

Proof Sketch for CNFs (2/2)

A CNF having no small rectangle cover:

Theorem (Sherstov, 2014)
Let $X = \{x_1, \ldots, x_n\}$ *and* $Y = \{y_1, \ldots, y_n\}$ *be two disjoint sets of variables. Then any* (X, Y)*-rectangle cover of the Boolean function* $\mathrm{SCOV}_n(X, Y) := \bigwedge_{i=1}^{n} x_i \vee y_i$ *has size* $\geqslant 2^n$.

We show that we can find $\mathrm{SCOV}_{\frac{k}{3 \times a^3 \times d^2}}(X, Y)$ within any CNF of treewidth $\geqslant k$.

→ Rephrase treewidth as treesplitwidth, a new measure capturing the 'performance' of a v-tree

Application to PQE

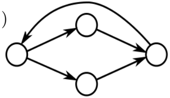
CFG-Datalog program P of body-size ≤ k_P

1 C(x): Subway('Corvisart', x)
C(x): C(y) AND Subway(y, x)

2 Goal(): NOT C("Châtelet")

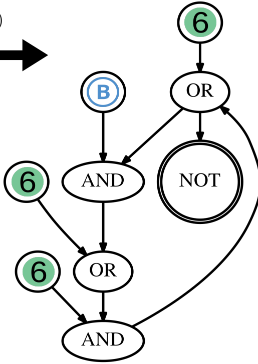
$O(g'(k_P,k_D)|P|)$

Two-way Alternating Tree Automaton A



$O(|A||E|)$

Provenance Cycluit

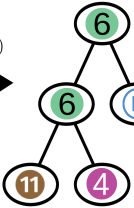


Database D of treewidth ≤ k_D

(Paris Metro map)

$O(g(k_D)|D|)$

Tree encoding E



"Under which conditions is it impossible to go from station Corvisart to station Châtelet with the subway?"

Application to PQE

Cycluit
size $O(|P| \times |D|)$
treewidth $O(|P|)$

Application to PQE

Cycluit		Circuit
size $O(\|P\| \times \|D\|)$	▸	size $O(2^{\|P\|^\alpha} \times \|D\|)$
treewidth $O(\|P\|)$		treewidth $O(2^{\|P\|^\alpha})$

Application to PQE

Cycluit
size $O(|P| \times |D|)$
treewidth $O(|P|)$

▸

Circuit
size $O(2^{|P|^\alpha} \times |D|)$
treewidth $O(2^{|P|^\alpha})$

▸

d-SDNNF
size $O(2^{2^{|P|^\alpha}} \times |D|)$

Application to PQE

Cycluit		Circuit		d-SDNNF
size $O(\|P\| \times \|D\|)$ treewidth $O(\|P\|)$	▸	size $O(2^{\|P\|^\alpha} \times \|D\|)$ treewidth $O(2^{\|P\|^\alpha})$	▸	size $O(2^{2^{\|P\|^\alpha}} \times \|D\|)$

Theorem

*Fix k_P and k_I. We can solve PQE of a **CFG-Datalog program P** on a **treelike database D** in time $O(2^{2^{|P|^\alpha}} |D|)$.*

Application to PQE

Cycluit
size $O(|P| \times |D|)$
treewidth $O(|P|)$

▸

Circuit
size $O(2^{|P|^\alpha} \times |D|)$
treewidth $O(2^{|P|^\alpha})$

▸

d-SDNNF
size $O(2^{2^{|P|^\alpha}} \times |D|)$

Theorem
*Fix k_P and k_I. We can solve PQE of a **CFG-Datalog program P** on a **treelike database D** in time $O(2^{2^{|P|^\alpha}} |D|)$.*

- 2EXP, but still better than previous nonelementary bounds

Conclusion (1/2)

Main contributions:

Conclusion (1/2)

Main contributions:

1. Detailed study of the combined complexity of PQE of conjunctive queries on binary signatures

Conclusion (1/2)

Main contributions:

1. Detailed study of the combined complexity of PQE of conjunctive queries on binary signatures
2. Efficient provenance computation for a new expressive query language (CFG-Datalog) on treelike data, introduction of a new provenance representation (cycluits)

Conclusion (1/2)

Main contributions:

1. Detailed study of the combined complexity of PQE of conjunctive queries on binary signatures
2. Efficient provenance computation for a new expressive query language (CFG-Datalog) on treelike data, introduction of a new provenance representation (cycluits)
3. Connections between two classes of Boolean circuits in knowledge compilation: width-based and semantics-based. Application to PQE of CFG-Datalog

Conclusion (2/2)

Ideas for Future work:

Conclusion (2/2)

Ideas for Future work:

- Correlations?

Conclusion (2/2)

Ideas for Future work:

- Correlations?
- More general provenance semirings?

Conclusion (2/2)

Ideas for Future work:

- Correlations?
- More general provenance semirings?
- Approximations?

Conclusion (2/2)

Ideas for Future work:

- Correlations?
- More general provenance semirings?
- Approximations?
- Notion of width for d-SDNNFs?

Conclusion (2/2)

Ideas for Future work:

- Correlations?
- More general provenance semirings?
- Approximations?
- Notion of width for d-SDNNFs?
- Obtain 1EXP for PQE of CFG-Datalog on treelike data?

Conclusion (2/2)

Ideas for Future work:

- Correlations?
- More general provenance semirings?
- Approximations?
- Notion of width for d-SDNNFs?
- Obtain 1EXP for PQE of CFG-Datalog on treelike data?
- Practical implementations?

Thanks for your attention!

CFG-Datalog: Definition by example

- Fragment of Datalog with stratified negation

CFG-Datalog: Definition by example

- Fragment of Datalog with stratified negation
- $\sigma = \sigma^{\mathrm{ext}} \sqcup \sigma^{\mathrm{int}} = \{R_1, R_2, \ldots\} \sqcup \{S_1, S_2, \ldots\}$

CFG-Datalog: Definition by example

- Fragment of Datalog with stratified negation
- $\sigma = \sigma^{\text{ext}} \sqcup \sigma^{\text{int}} = \{R_1, R_2, \ldots\} \sqcup \{S_1, S_2, \ldots\}$
- Boolean programs: special 0-ary intensional predicate Goal()

CFG-Datalog: Definition by example

- Fragment of Datalog with stratified negation
- $\sigma = \sigma^{\text{ext}} \sqcup \sigma^{\text{int}} = \{R_1, R_2, \ldots\} \sqcup \{S_1, S_2, \ldots\}$
- Boolean programs: special 0-ary intensional predicate Goal()

$$\begin{cases} \vdots \\ S_3(x,y,t) \leftarrow R_1(x,y) \land S_3(y,t,y) \land S_2(x,t) \land \neg S_1(x,z) \\ \vdots \\ \text{Goal}() \leftarrow \cdots \end{cases}$$

CFG-Datalog: Definition by example

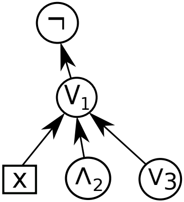
- Fragment of Datalog with stratified negation
- $\sigma = \sigma^{\text{ext}} \sqcup \sigma^{\text{int}} = \{R_1, R_2, \ldots\} \sqcup \{S_1, S_2, \ldots\}$
- Boolean programs: special 0-ary intensional predicate Goal()

$$\begin{cases} \vdots \\ S_3(x,y,t) \leftarrow R_1(x,y) \wedge S_3(y,t,y) \wedge S_2(x,t) \wedge \neg S_1(x,z) \\ \vdots \\ \text{Goal}() \leftarrow \cdots \end{cases}$$

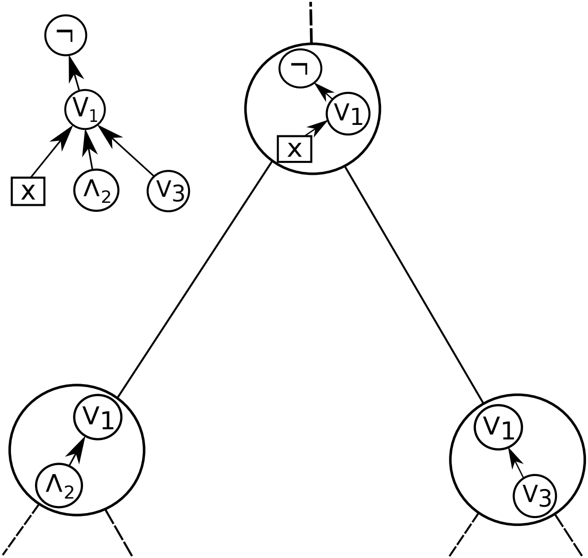
body-size $= \mathrm{MaxArity}(\sigma) \times \max_{\text{rule } r} \mathrm{NbAtoms}(r)$

"*size to write a rule*"

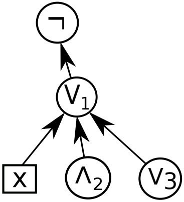
Construction sketch for slide 36

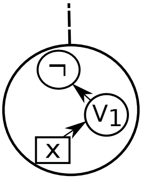


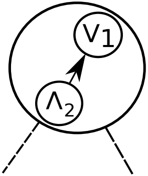
Construction sketch for slide 36

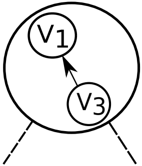


Construction sketch for slide 36

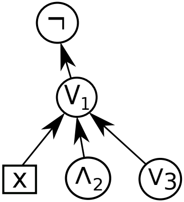


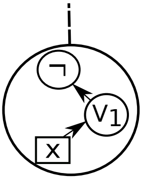


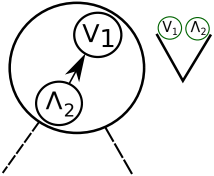


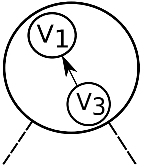


Construction sketch for slide 36

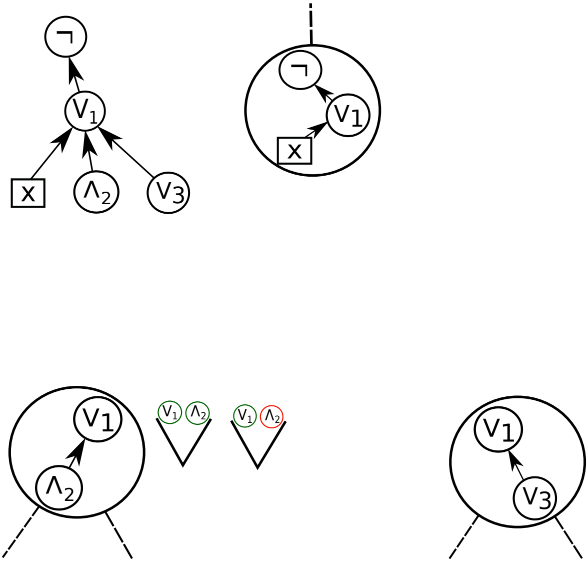




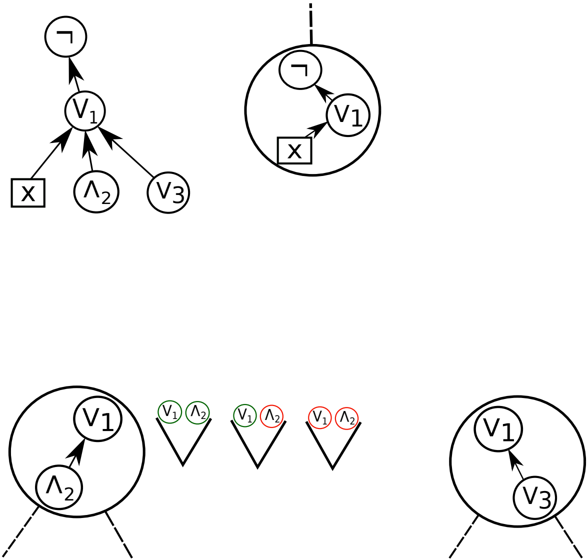




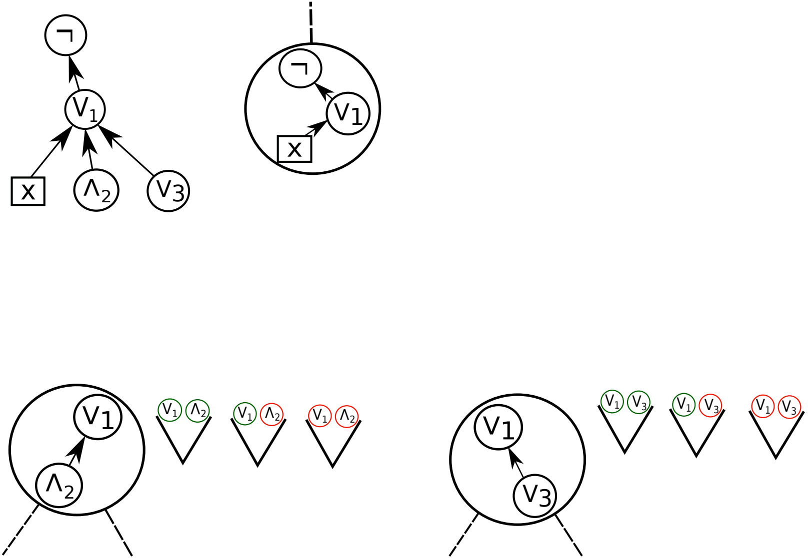
Construction sketch for slide 36



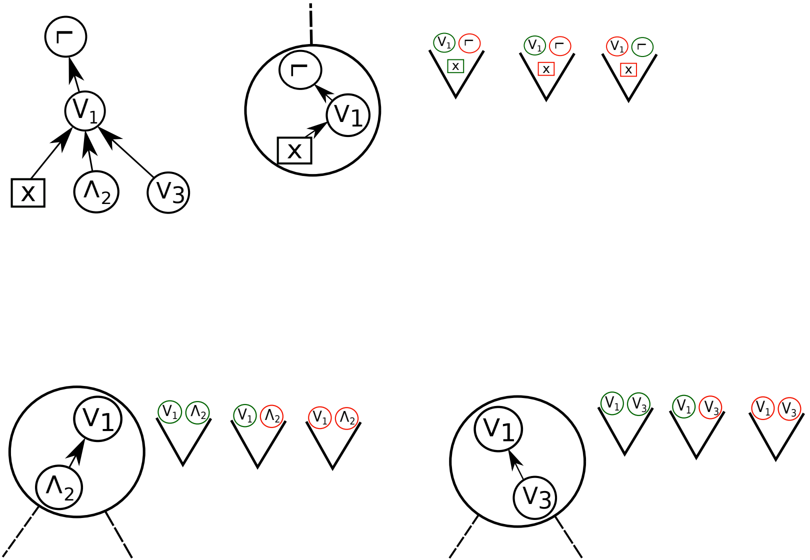
Construction sketch for slide 36



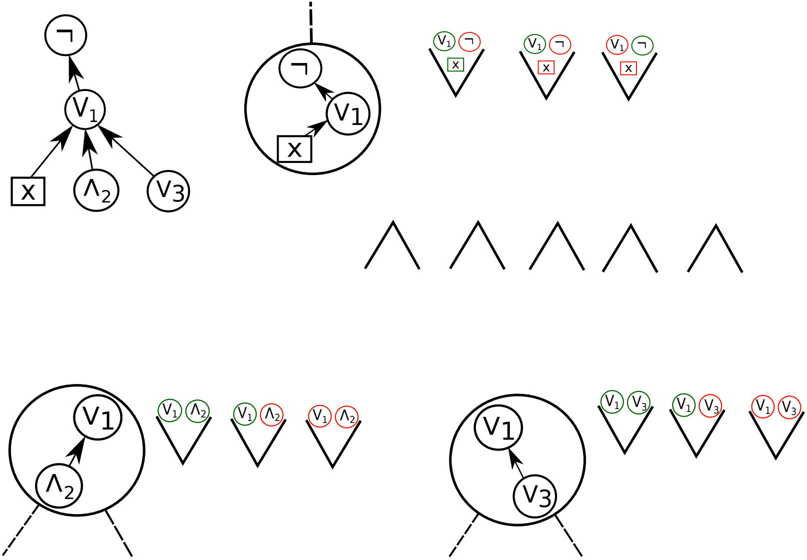
Construction sketch for slide 36



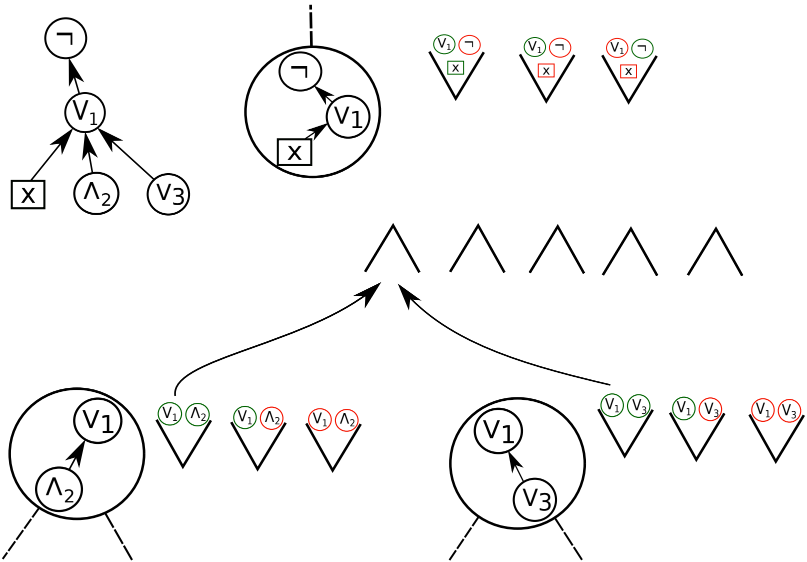
Construction sketch for slide 36



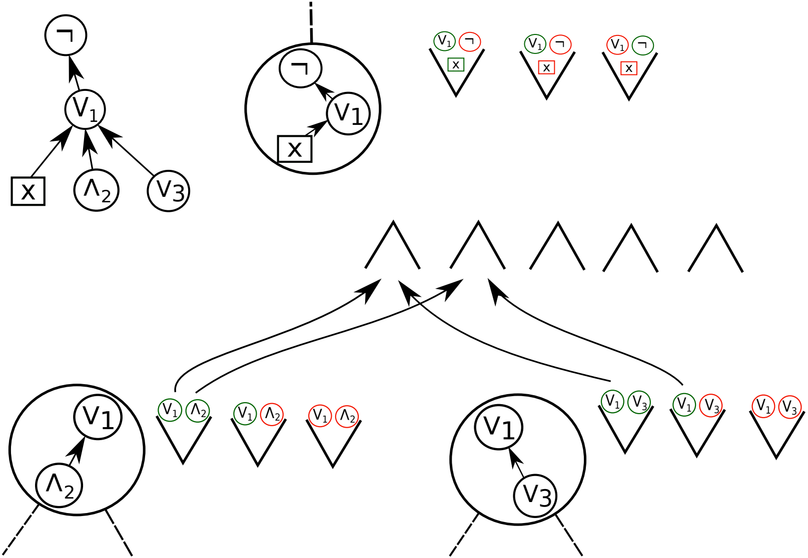
Construction sketch for slide 36



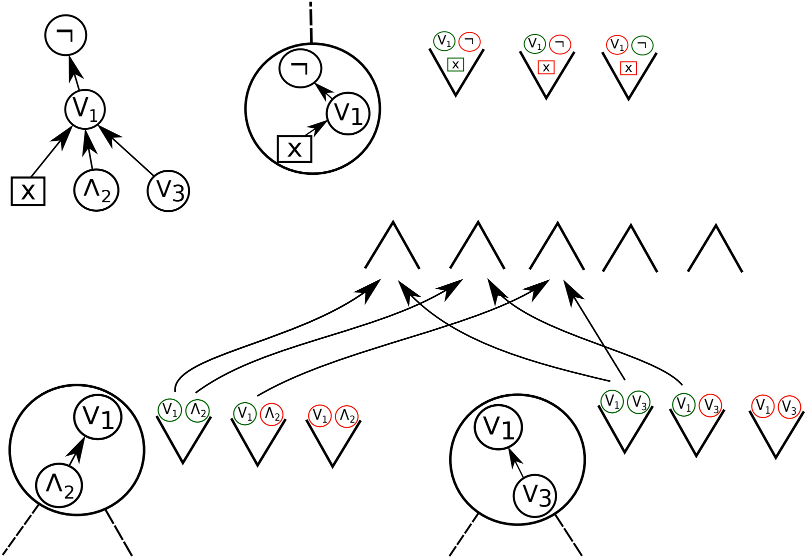
Construction sketch for slide 36



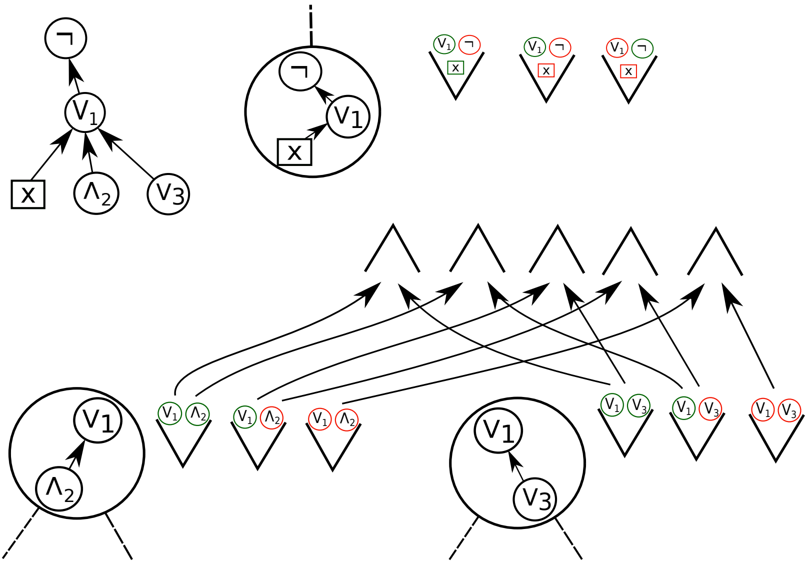
Construction sketch for slide 36



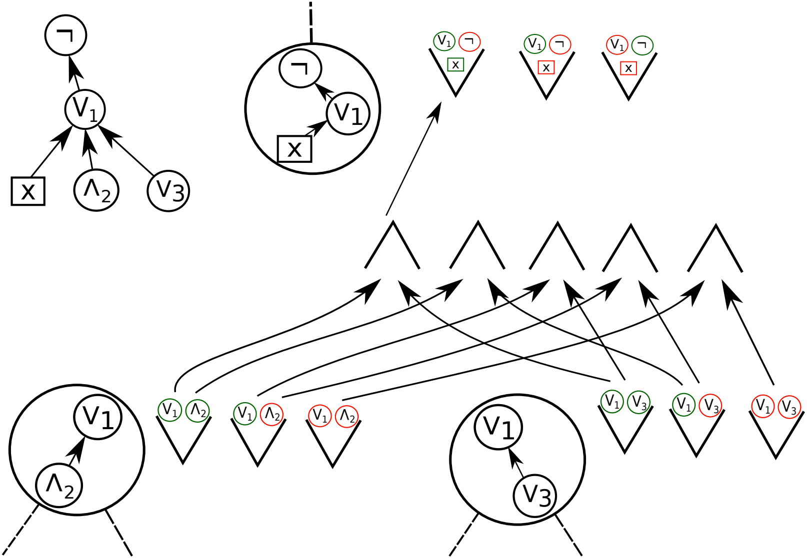
Construction sketch for slide 36



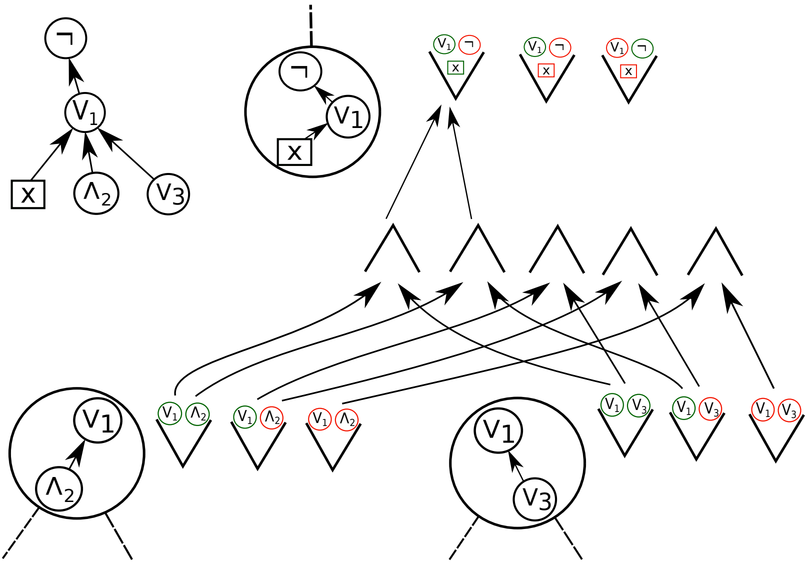
Construction sketch for slide 36



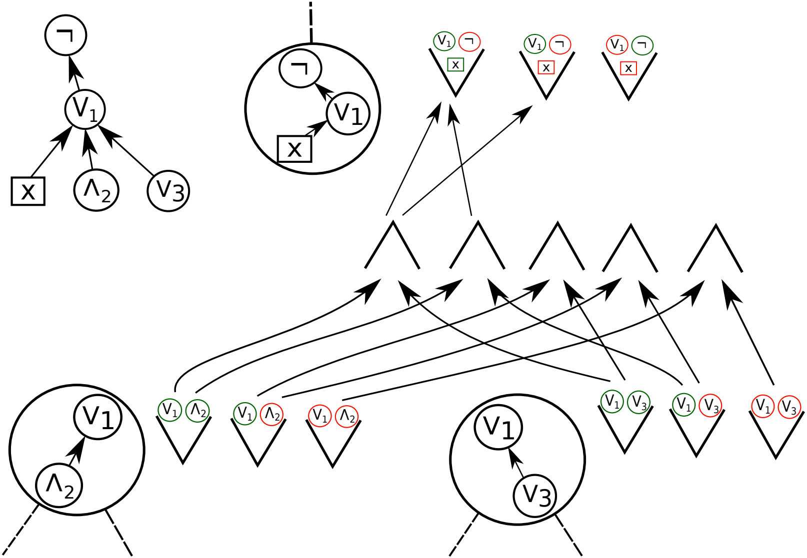
Construction sketch for slide 36



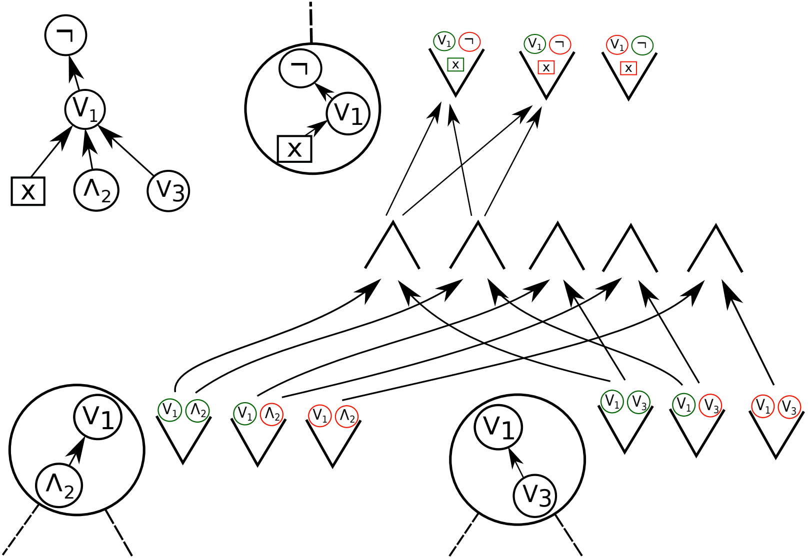
Construction sketch for slide 36



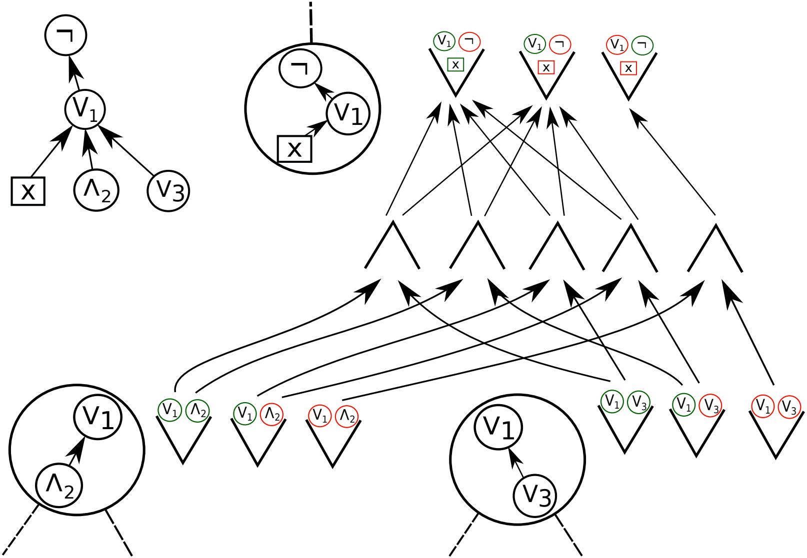
Construction sketch for slide 36



Construction sketch for slide 36



Construction sketch for slide 36



Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$

Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$

I:

Q:

Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$

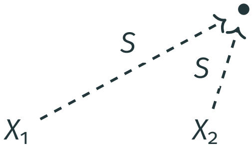
I:

Q:

Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$

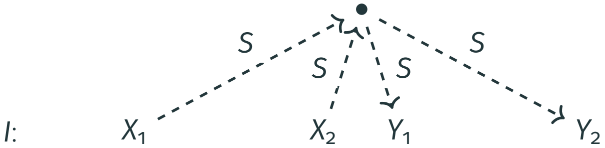
I:



Q:

Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

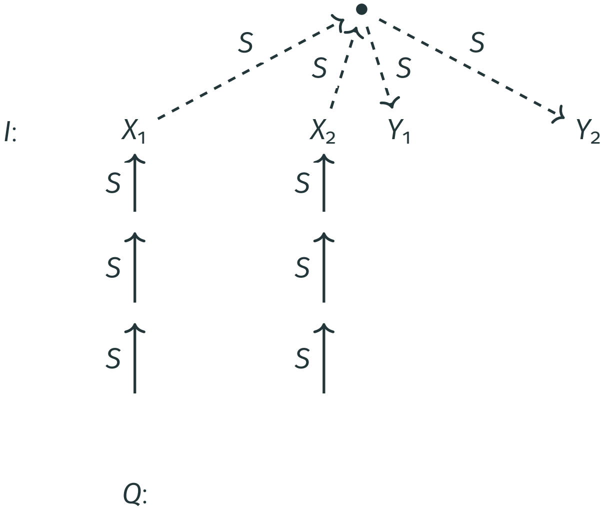
$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



Q:

Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

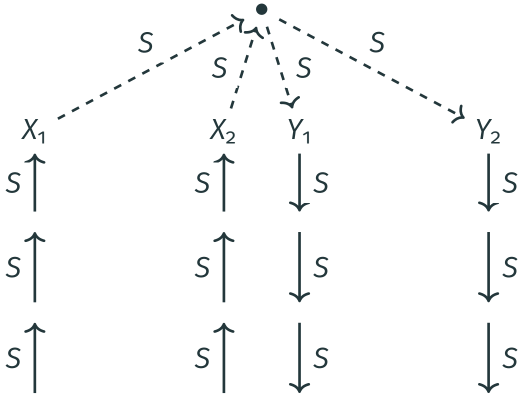
$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$

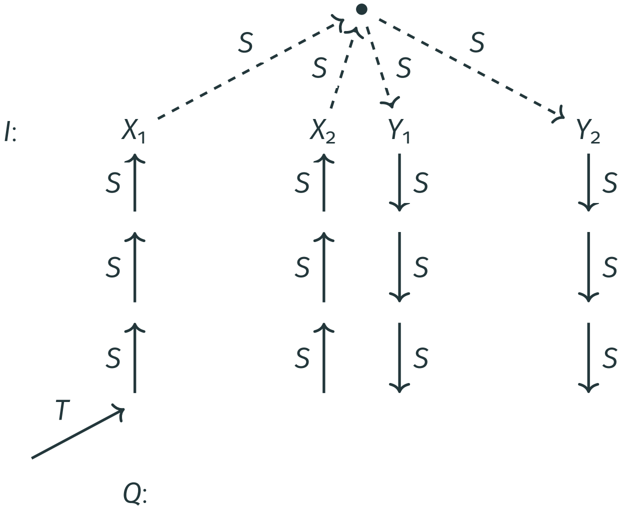
I:



Q:

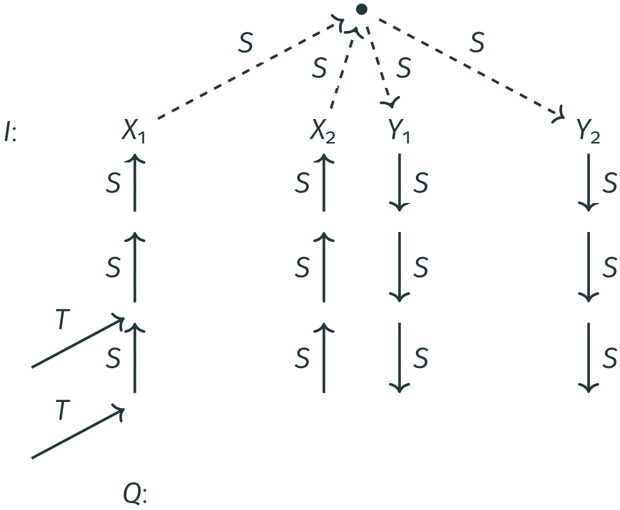
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



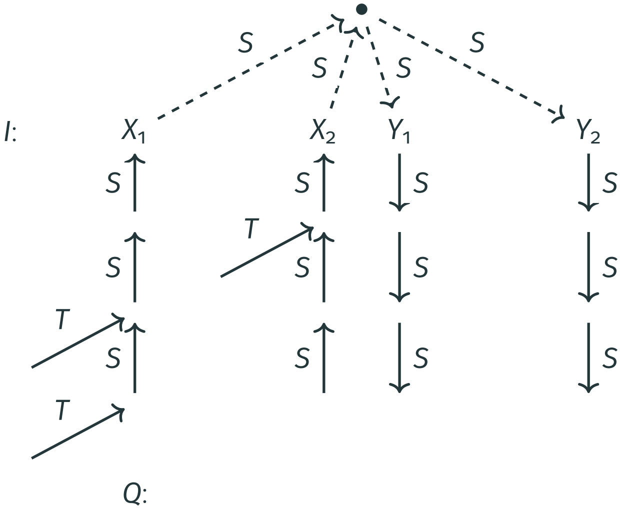
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



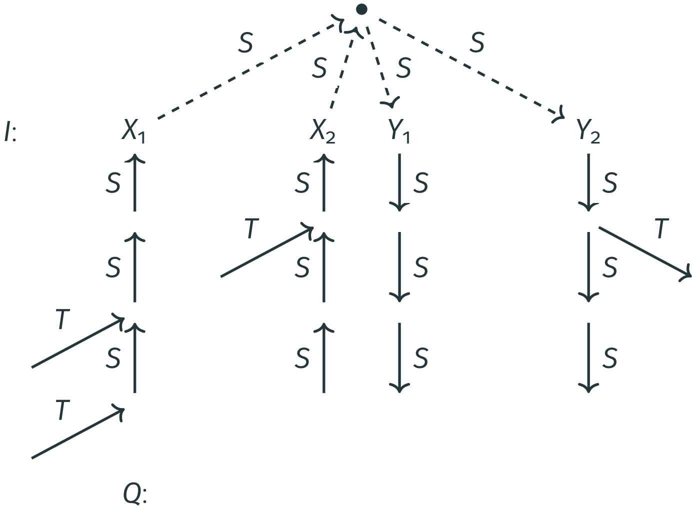
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



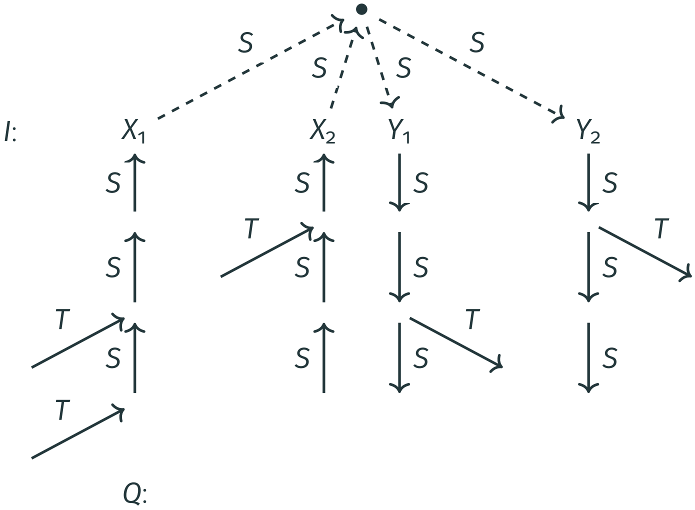
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



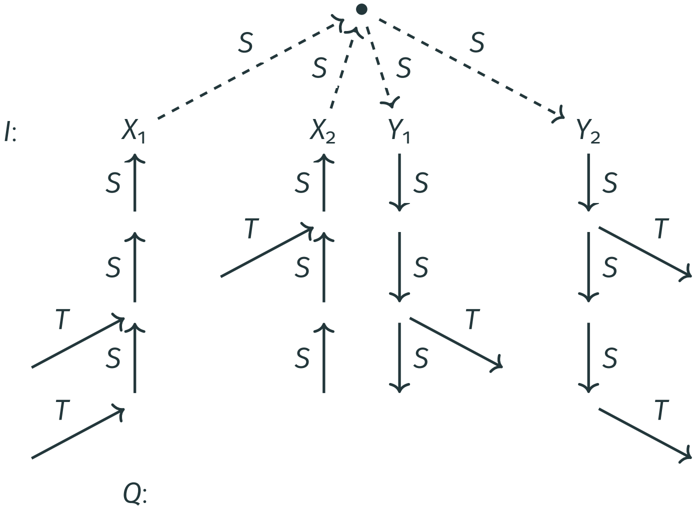
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



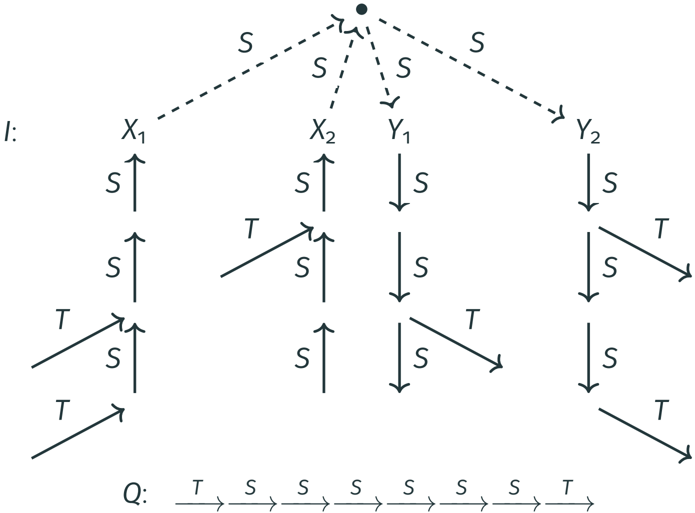
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



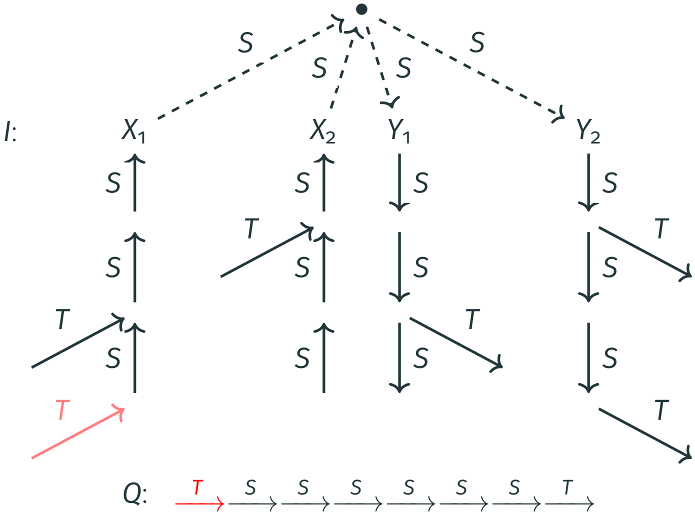
Reduction for $\mathcal{Q} =$ *one-way paths*, $\mathcal{I} =$ *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



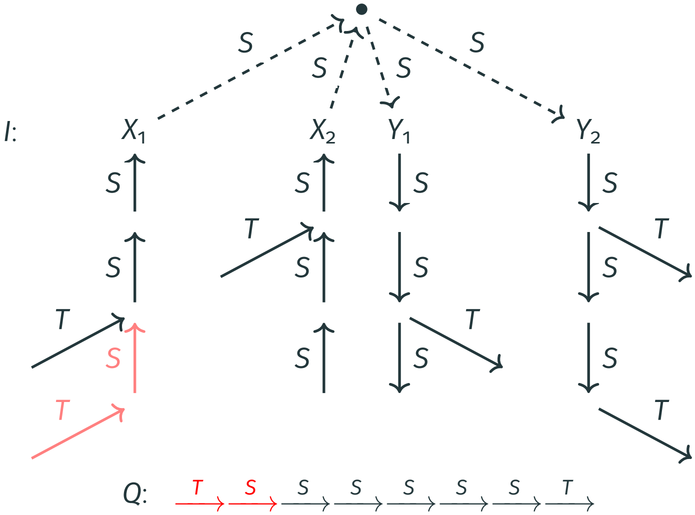
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



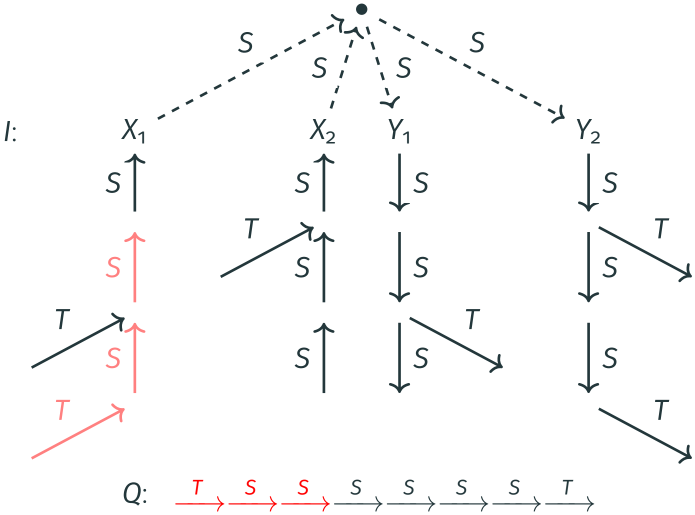
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



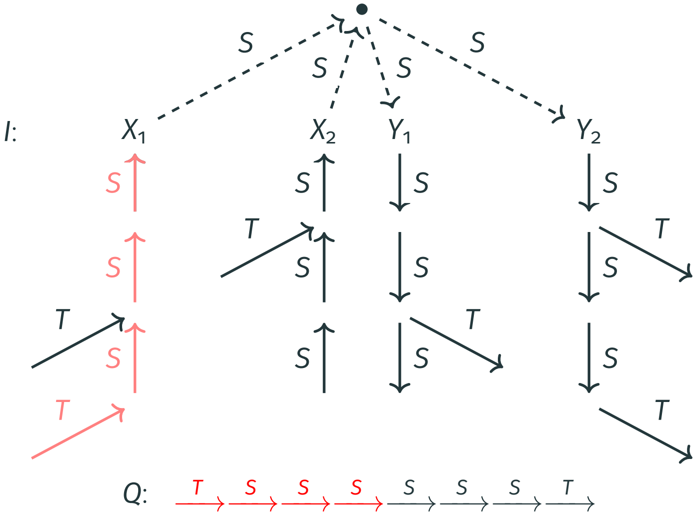
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



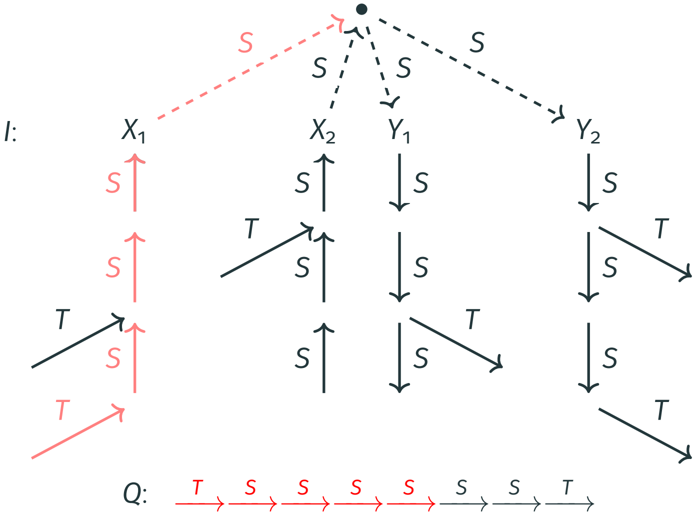
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



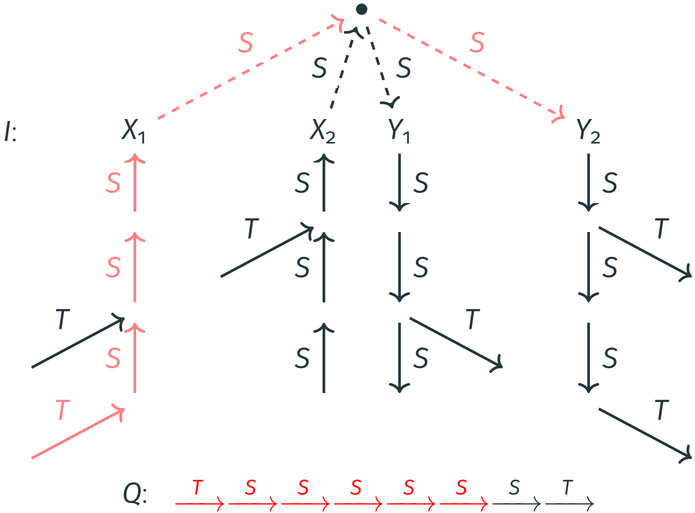
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



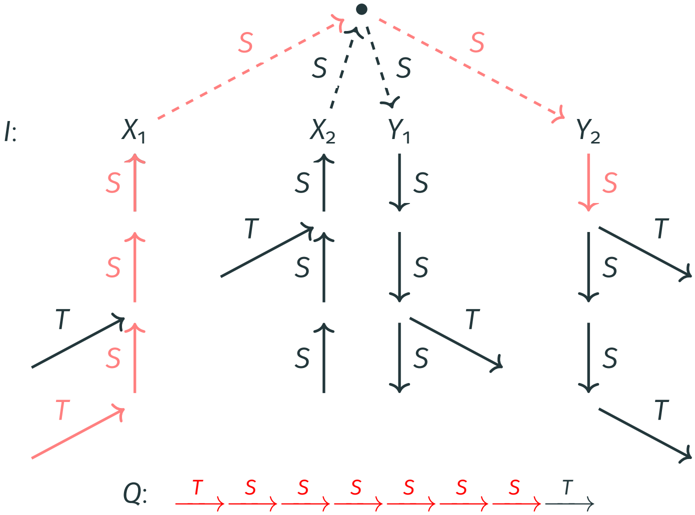
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



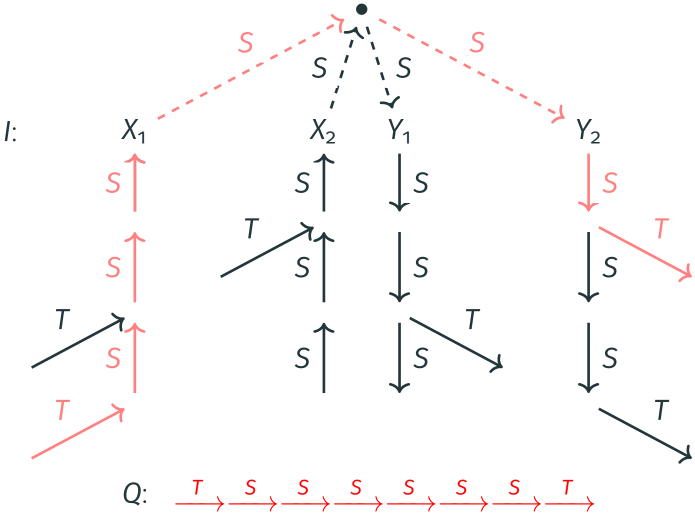
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



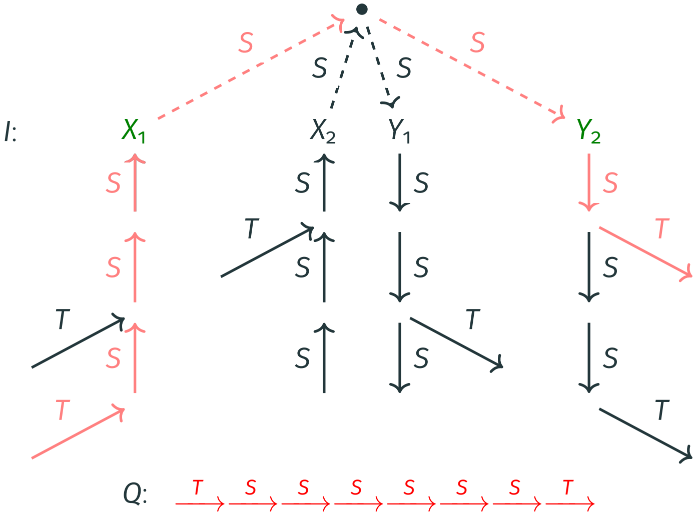
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



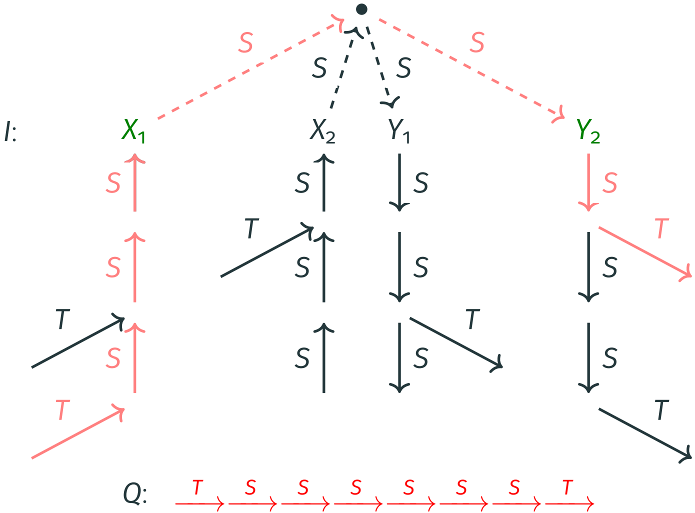
Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \lor X_1 Y_1 \lor X_2 Y_2$



Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$



Reduction for $\mathcal{Q}$ = *one-way paths*, $\mathcal{I}$ = *polytrees*

$\varphi = X_1 Y_2 \vee X_1 Y_1 \vee X_2 Y_2$

$\#\varphi = \Pr((I, \pi) \models Q) \times 2^{|\text{vars}(\varphi)|}$