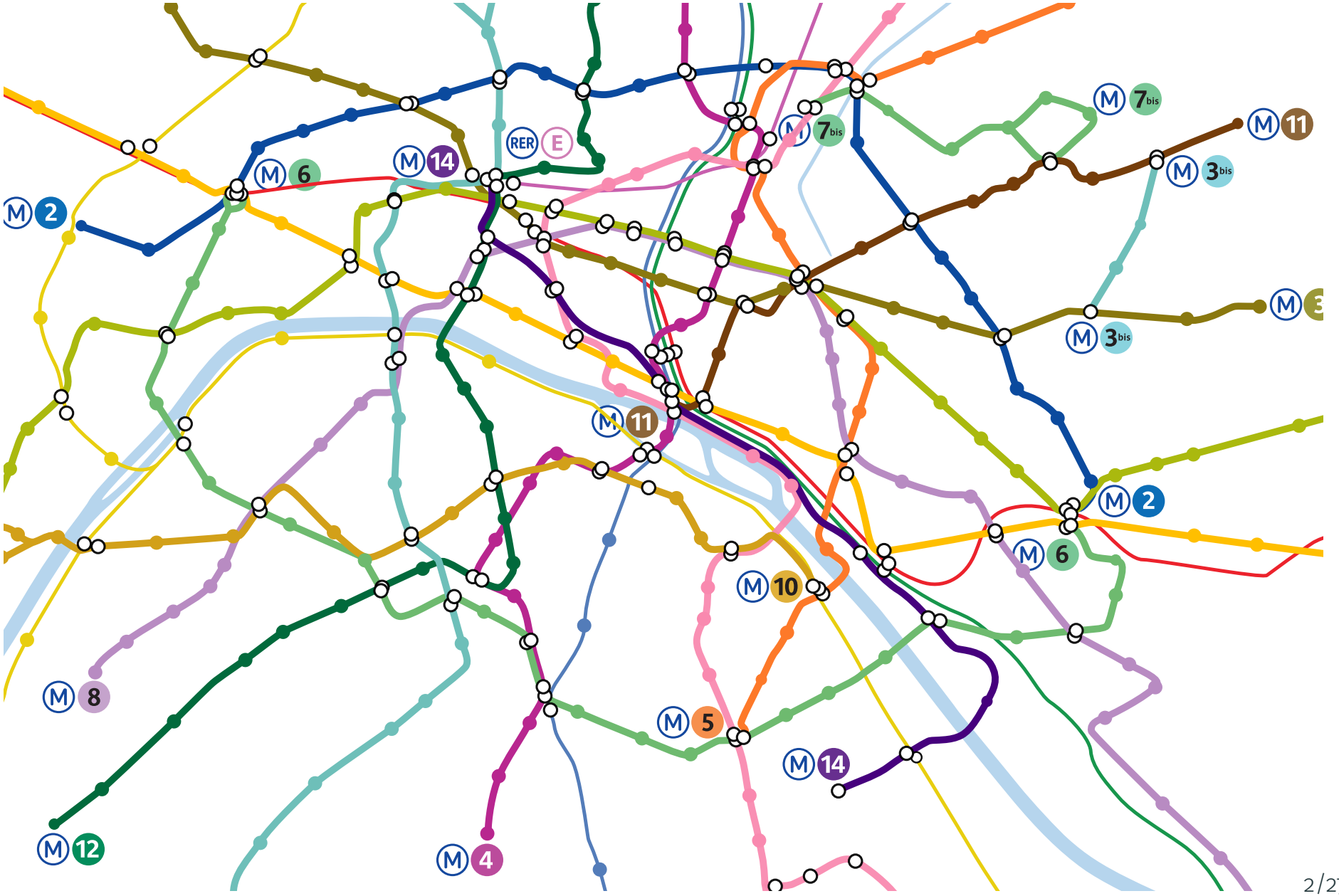
Leveraging the structure of uncertain data

Antoine Amarilli

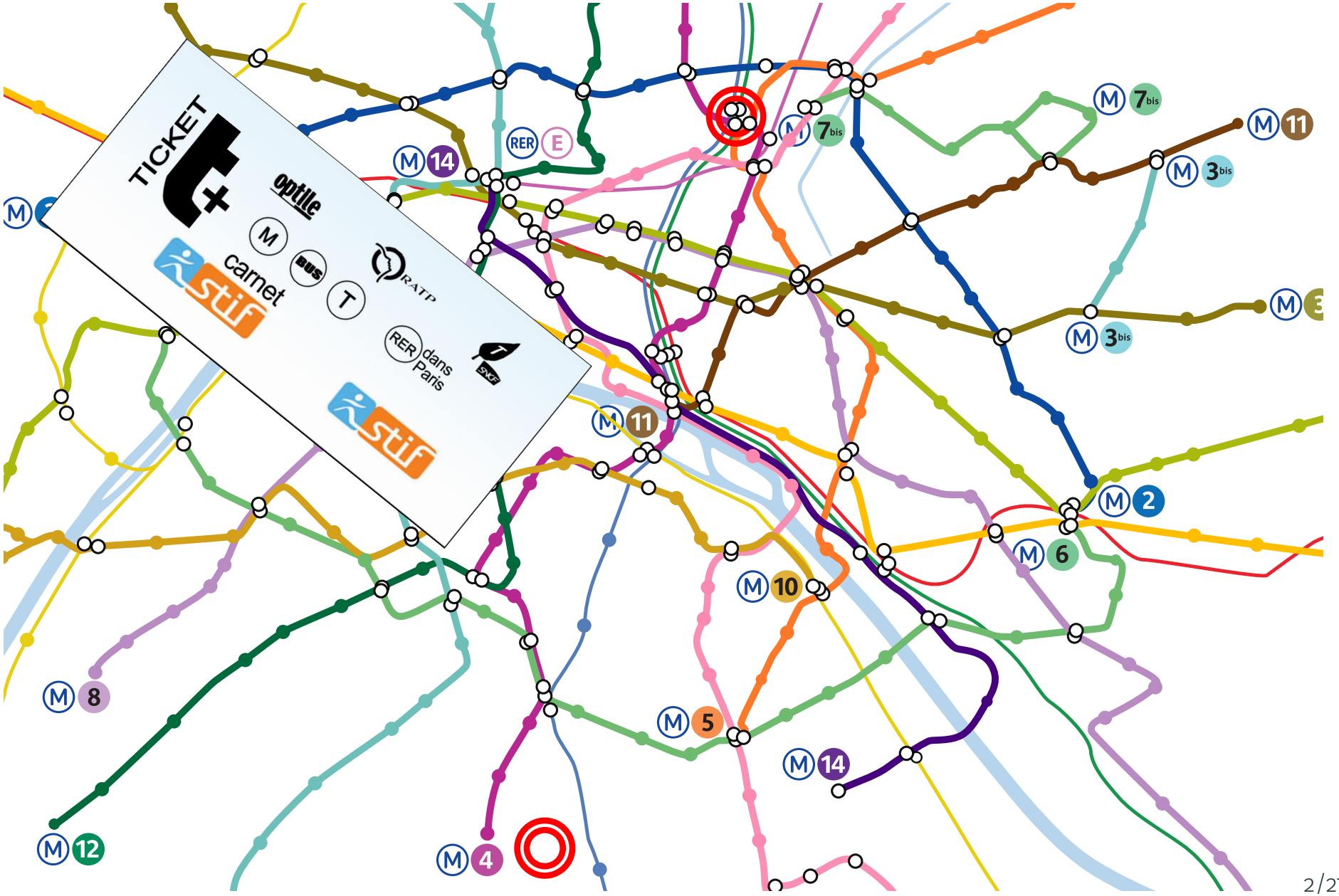
May 26, 2017

Example application: Subway routing

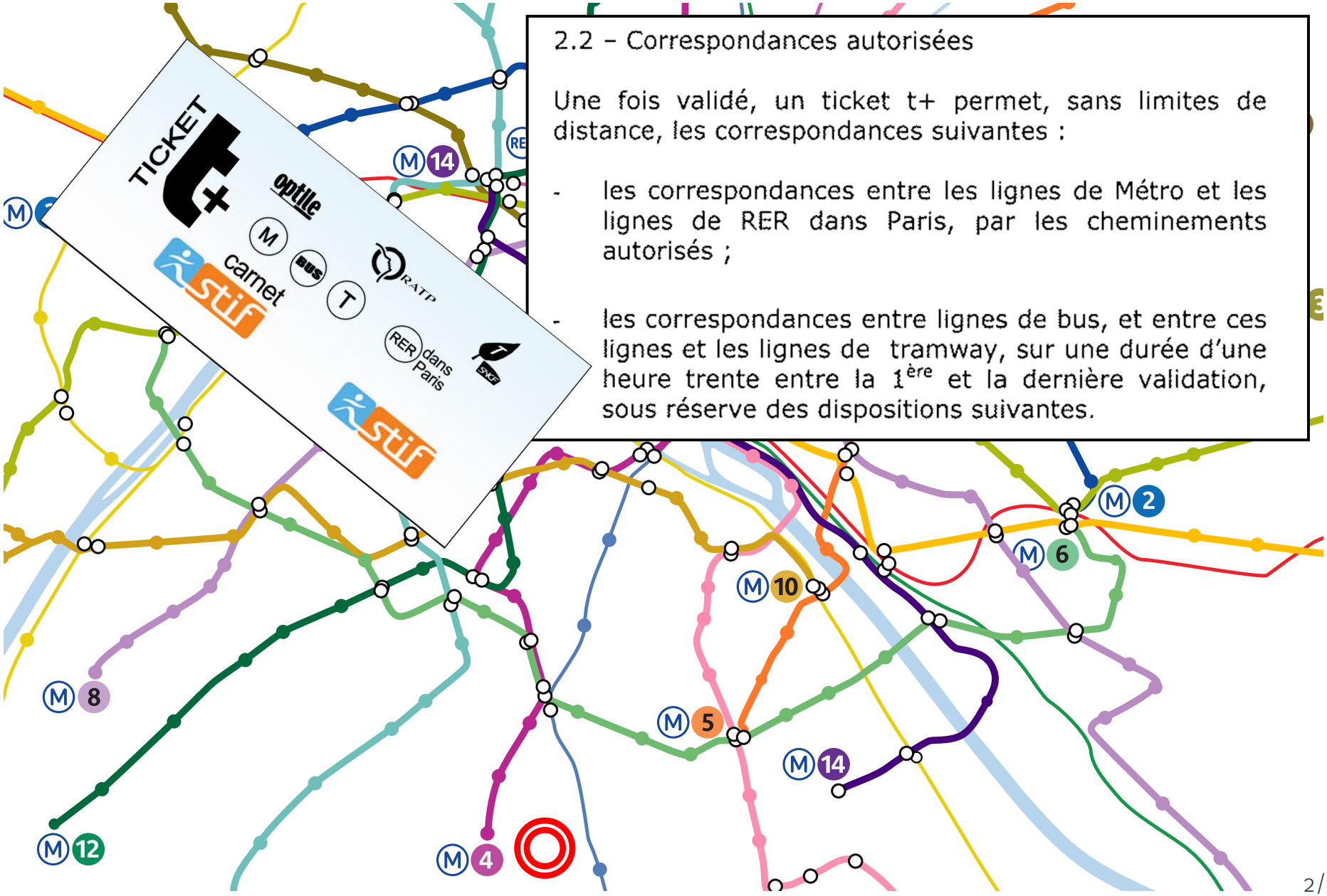


Example application: Subway routing

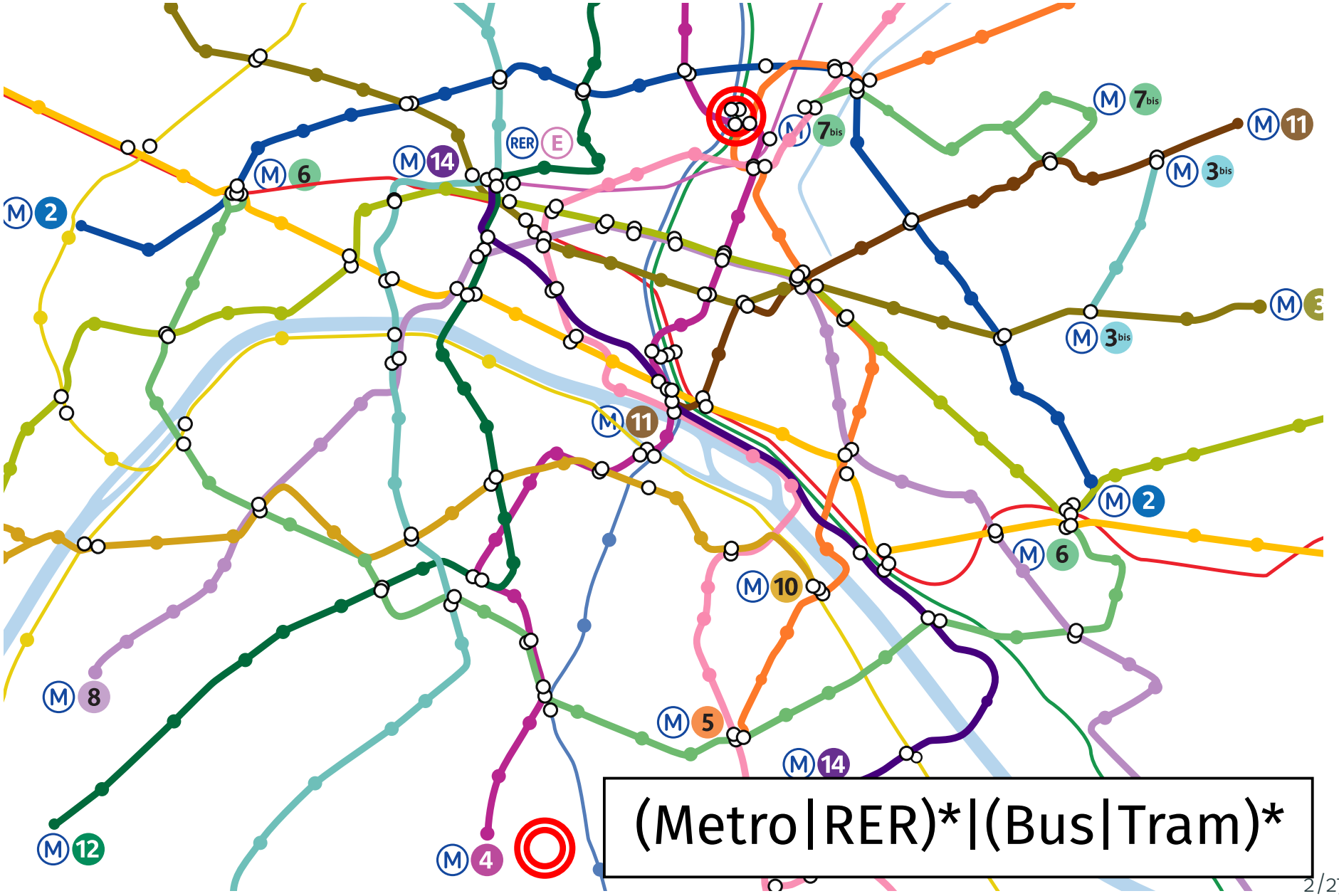
Example application: Subway routing



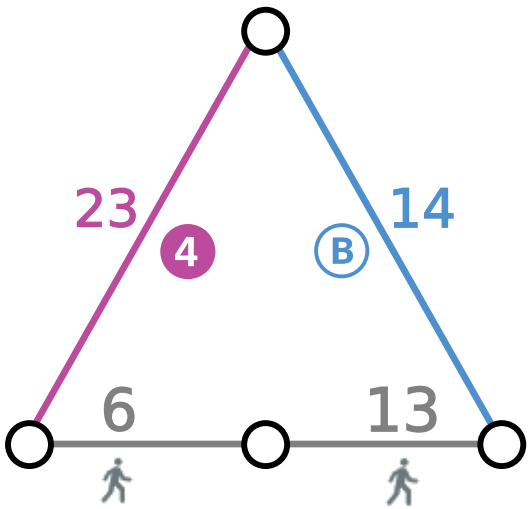
Example application: Subway routing



Example application: Subway routing



Database

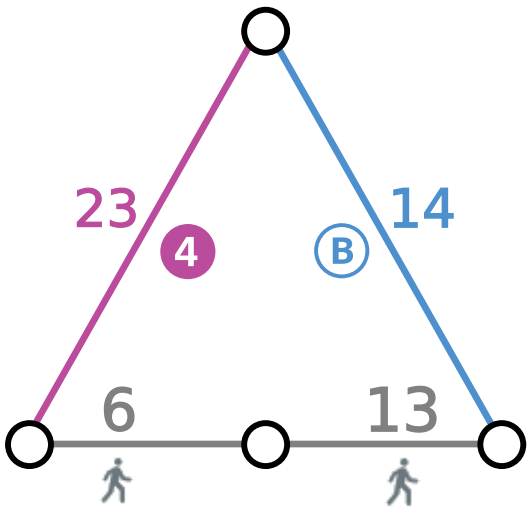


- (Hyper)graph
- Collection of ground facts

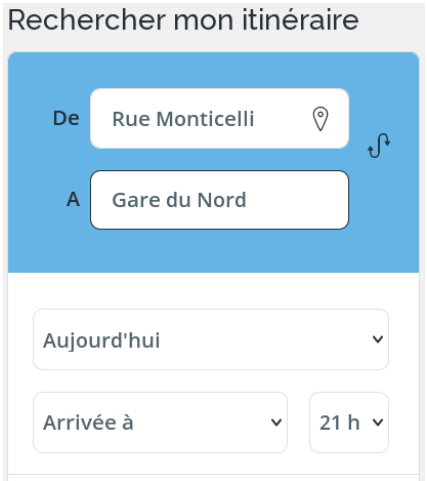
$G(aa_1, ab_2)$, $G(ab_2, ac_3)$,
$S(aa_1, m_4)$, $S(ab_2, r_B)$, ...

Database theory and query evaluation

Database



Query



- (Hyper)graph
- Collection of ground facts

$G(aa_1, ab_2)$, $G(ab_2, ac_3)$,
$S(aa_1, m_4)$, $S(ab_2, r_B)$, ...

- Regular path

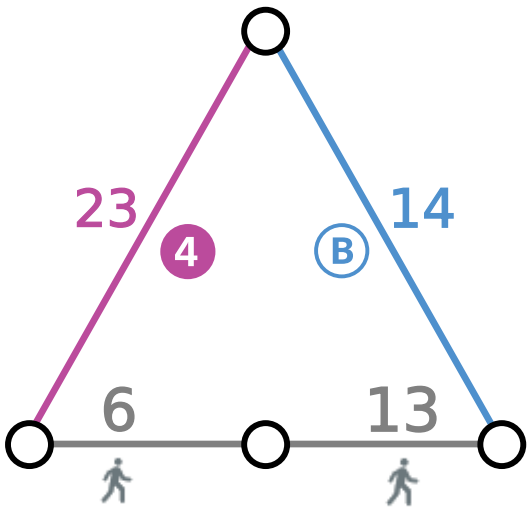
  (Metro|RER)*
  |(Bus|Tram)*

- Logic formula

$$\forall X(rm \in X \wedge \forall xy\ (x \in X \wedge G(x, y) \rightarrow y \in X)) \rightarrow gn \in X$$

Database theory and query evaluation

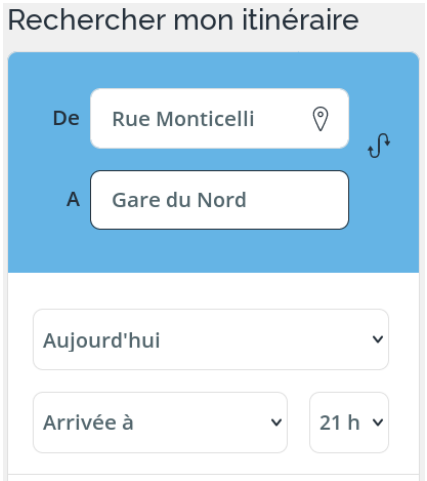
Database



- (Hyper)graph
- Collection of ground facts

$G(aa_1, ab_2)$, $G(ab_2, ac_3)$,
$S(aa_1, m_4)$, $S(ab_2, r_B)$, ...

Query



- Regular path

  (Metro|RER)*
  |(Bus|Tram)*

- Logic formula

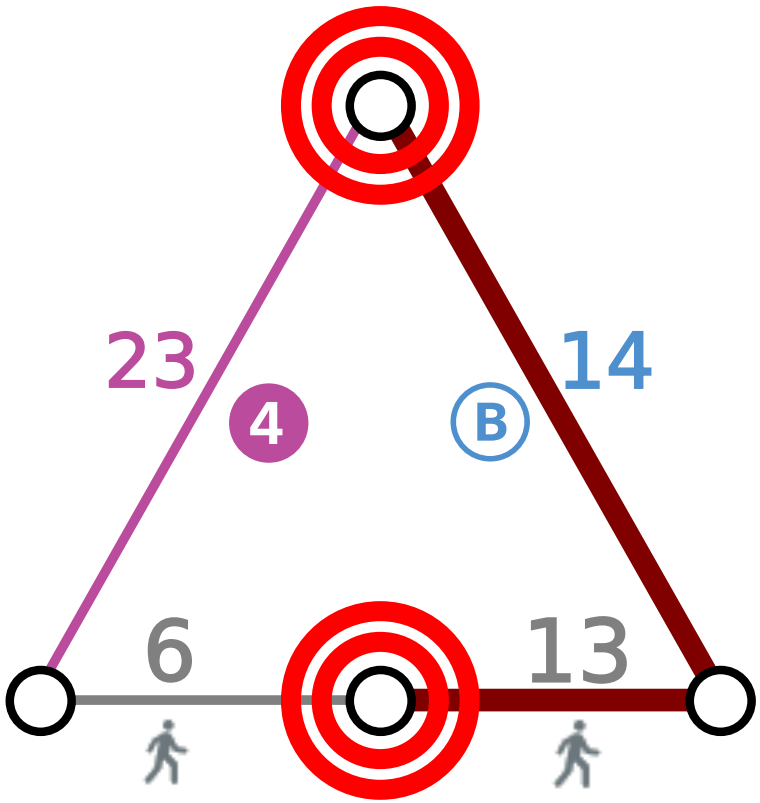
$\forall X(rm \in X \wedge \forall xy$
$(x \in X \wedge G(x, y) \rightarrow$
$y \in X)) \rightarrow gn \in X$

Result

- TRUE/FALSE

↔ Model checking

Probabilistic query evaluation



Probabilistic query evaluation

Panne du RER B : trafic interrompu entre Paris et Roissy, des TGV en renfort

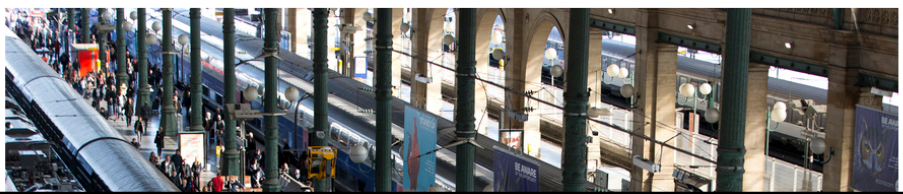
> Transports | 06 décembre 2016, 9h56 | MAJ : 06 décembre 2016, 17h03 |

Probabilistic query evaluation

Panne du RER B : trafic interrompu entre

Paris : pourquoi il y a autant de perturbations sur le RER B et à Gare du Nord

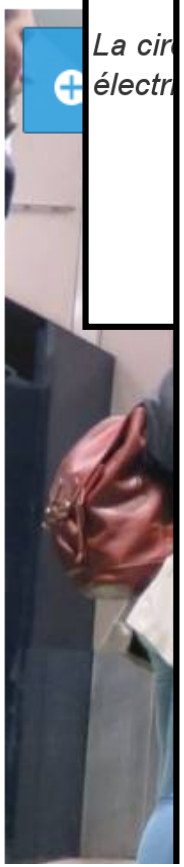
La circulation de l'ensemble des trains au départ de gare du Nord est totalement interrompue à la suite d'une panne électrique.



Probabilistic query evaluation

Panne du RER B : trafic interrompu entre

Paris : pourquoi il y a autant de perturbations sur le



INCIDENT SUR LE RER B : QUE S'EST-IL PASSÉ CE MATIN ?

Malaise voyageur et application des mesures de sécurité : pour quelles raisons le trafic a-t-il été perturbé ce matin sur la ligne B ?

Pour beaucoup, le voyage a été difficile ce matin. Au fil de vos réactions sur Twitter notamment, je constate que les raisons de ces perturbations ne paraissent pas cohérentes. Je tiens donc à vous apporter des premiers

Probabilistic query evaluation

Panne du RER B : trafic interrompu entre

Paris : pourquoi il y a autant de perturbations sur le

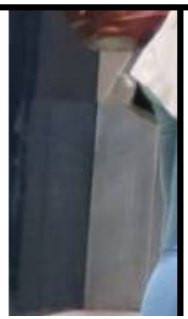
INCIDENT SUR LE RER B · QUE

ACTUALITÉS

Le RER B en panne, les voyageurs n'ont pas eu d'autre choix que de descendre sur les voies

Alors que la circulation alternée a augmenté le nombre de voyageurs dans les transports en commun, le RER B s'est retrouvé à l'arrêt.

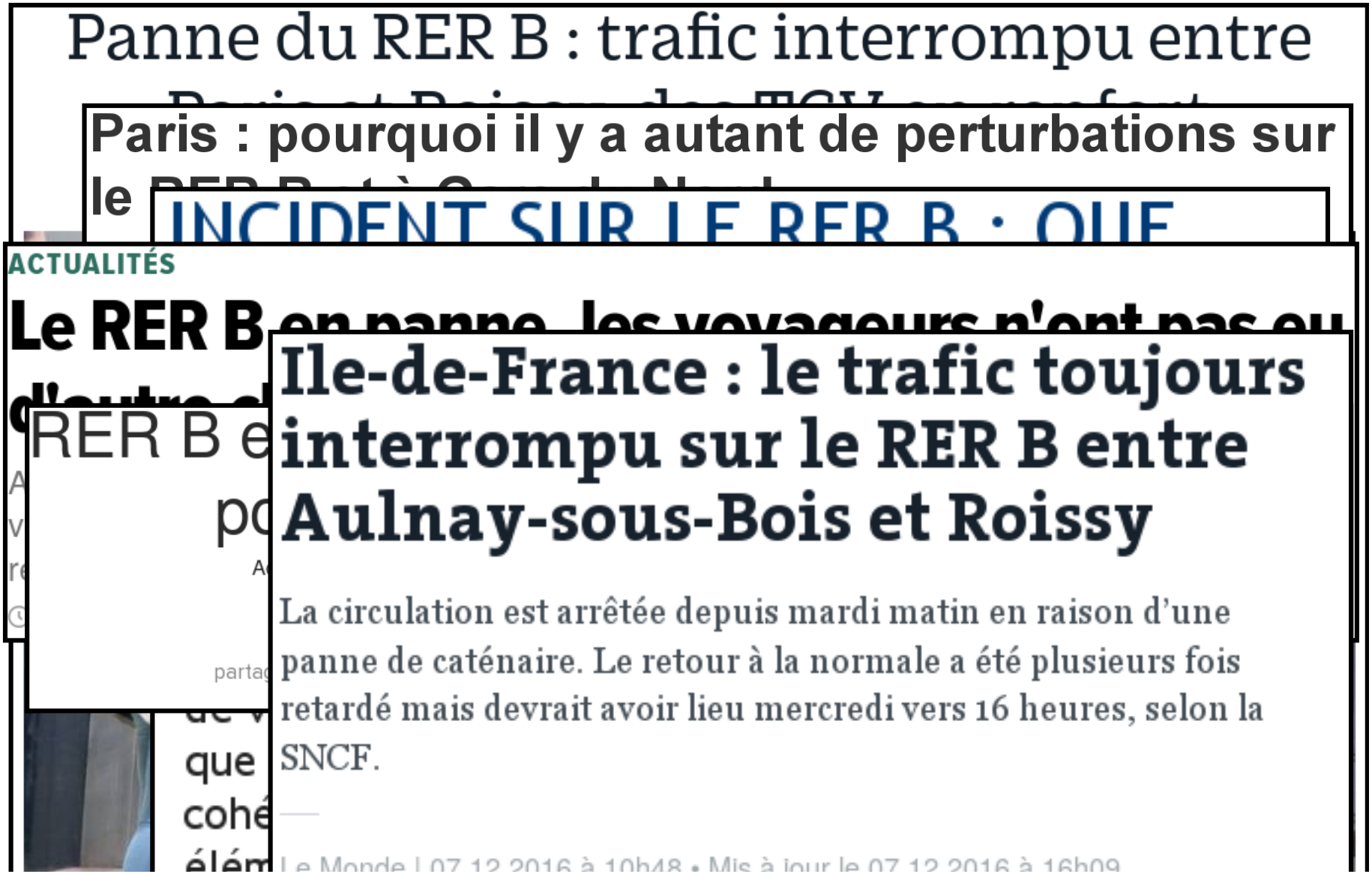
06/12/2016 11:57 CET | **Actualisé** 06/12/2016 20:14 CET



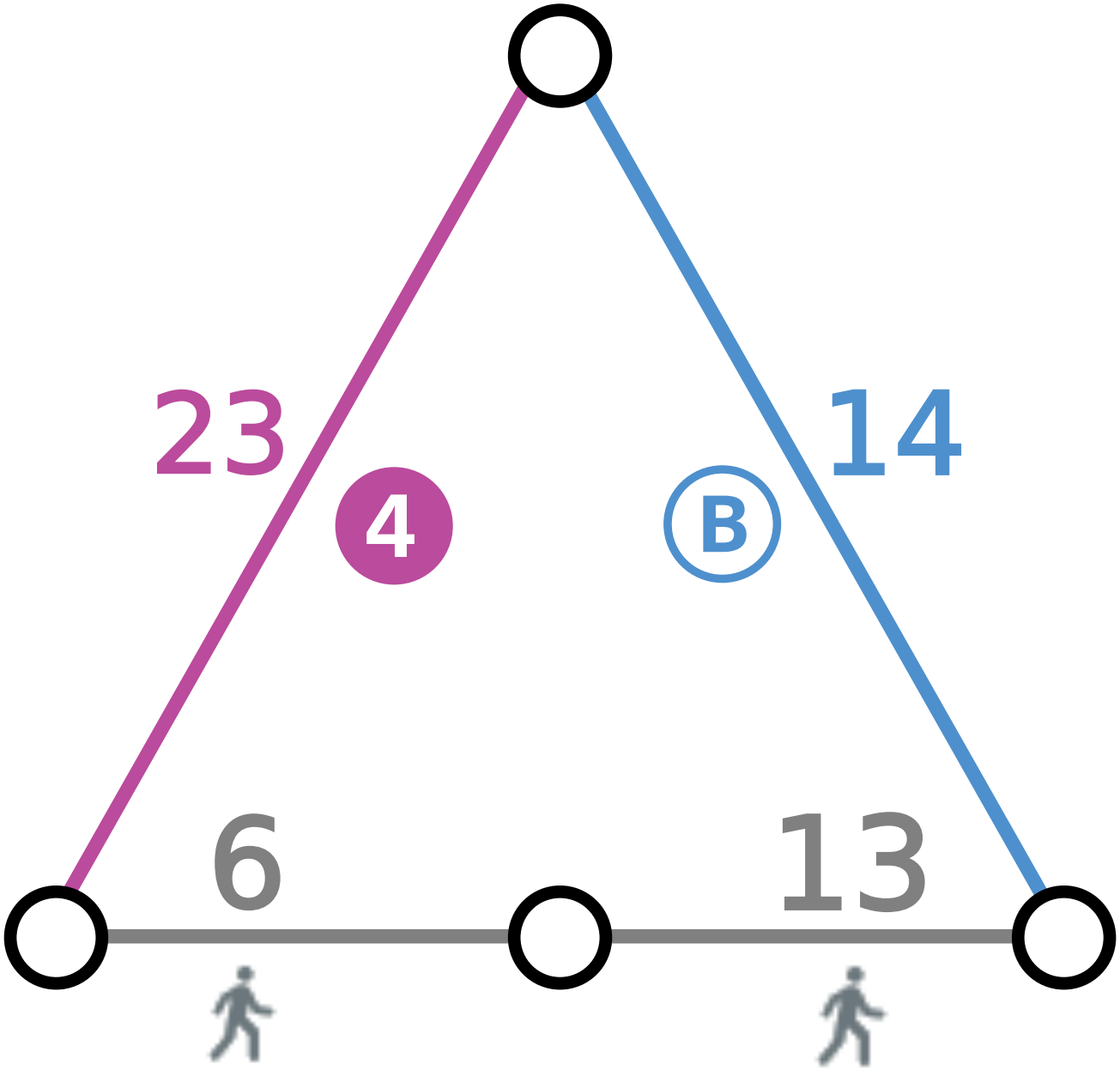
Pour beaucoup, le voyage a été difficile ce matin. Au fil de vos réactions sur Twitter notamment, je constate que les raisons de ces perturbations ne paraissent pas cohérentes. Je tiens donc à vous apporter des premiers éléments d'explication, que nous pourrons développer

Probabilistic query evaluation

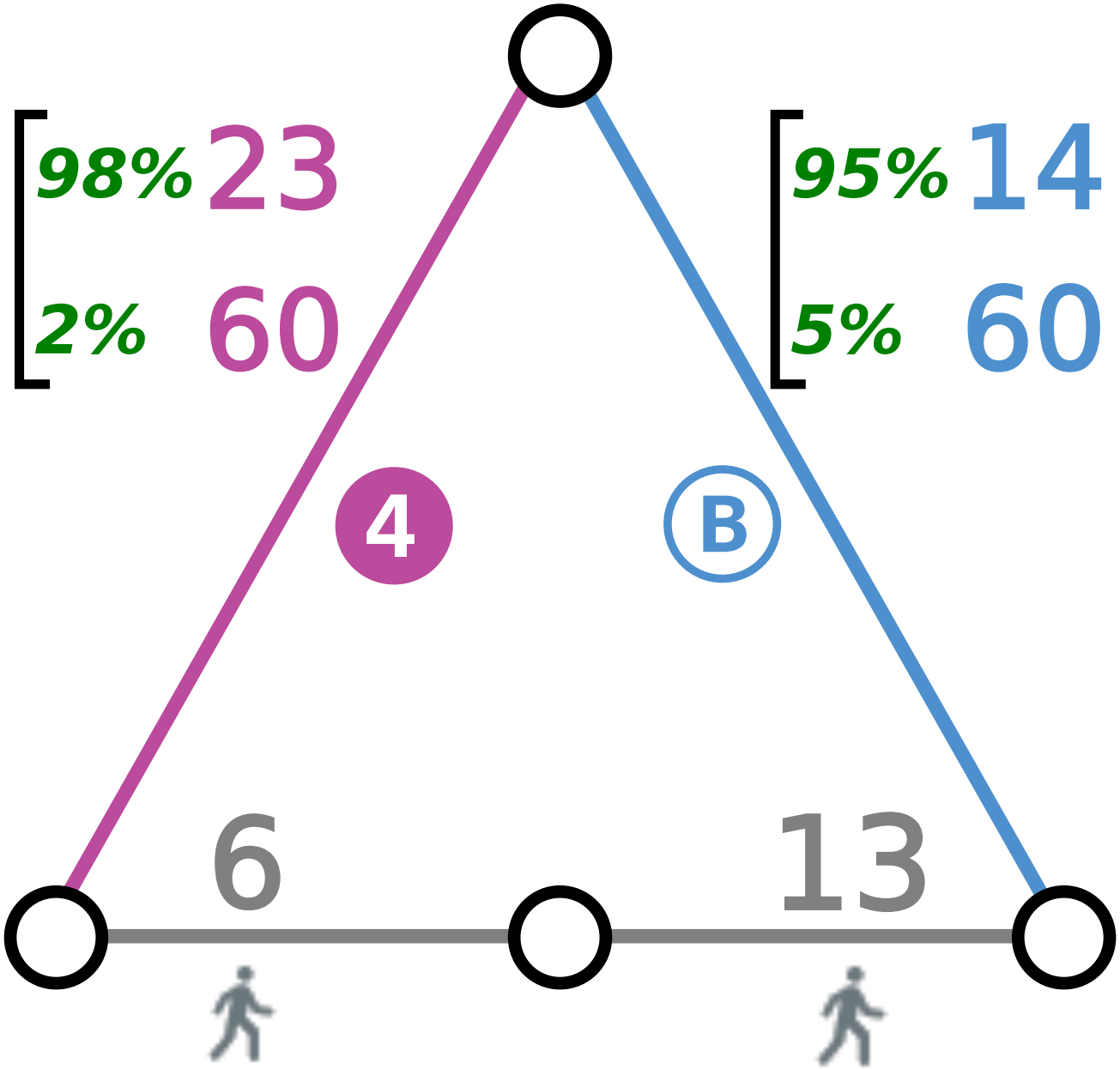
Probabilistic query evaluation



Probabilistic query evaluation

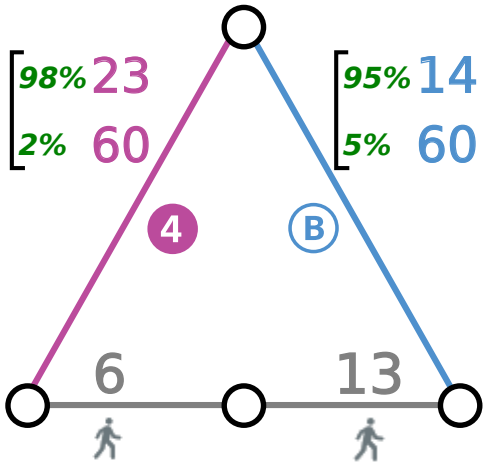


Probabilistic query evaluation



Probabilistic query evaluation

Probabilistic database

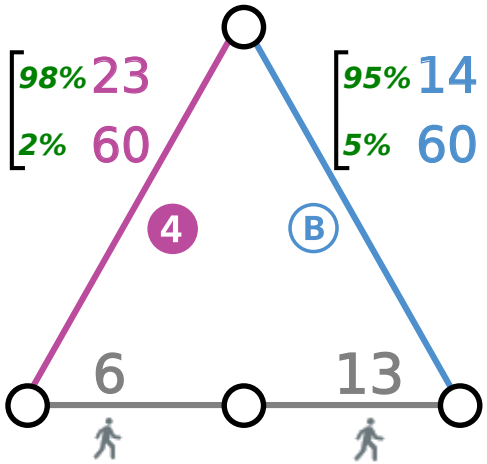


- (Hyper)graph
- Collection of ground facts

+ independent probabilities

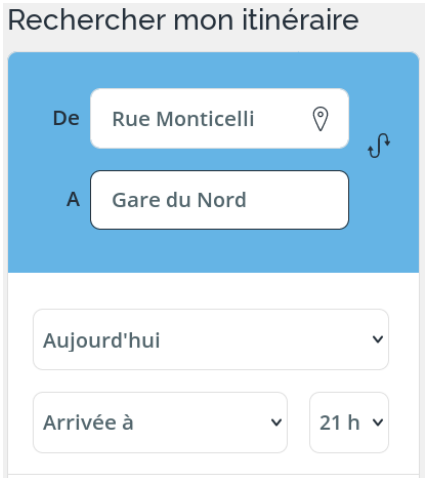
Probabilistic query evaluation

Probabilistic database



+

Query



- (Hyper)graph
- Collection of ground facts

+ independent probabilities

- Regular path

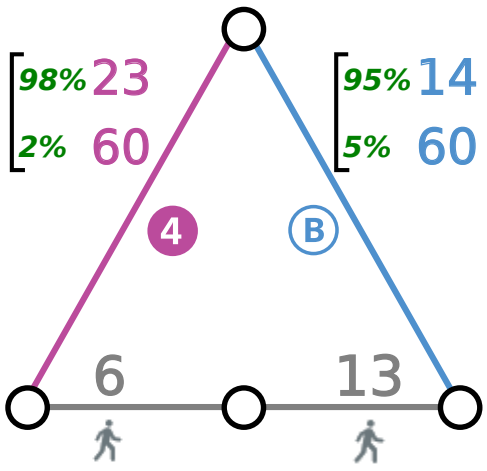
  (Metro|RER)*
  |(Bus|Tram)*

- Logic formula

$$\forall X(rm \in X \wedge \forall xy\ (x \in X \wedge G(x, y) \rightarrow y \in X)) \rightarrow gn \in X$$

Probabilistic query evaluation

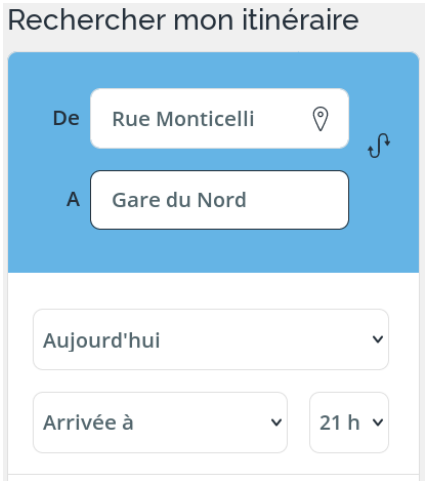
Probabilistic database



- (Hyper)graph
- Collection of ground facts

+ independent probabilities

Query



- Regular path

  (Metro|RER)* |(Bus|Tram)*

- Logic formula

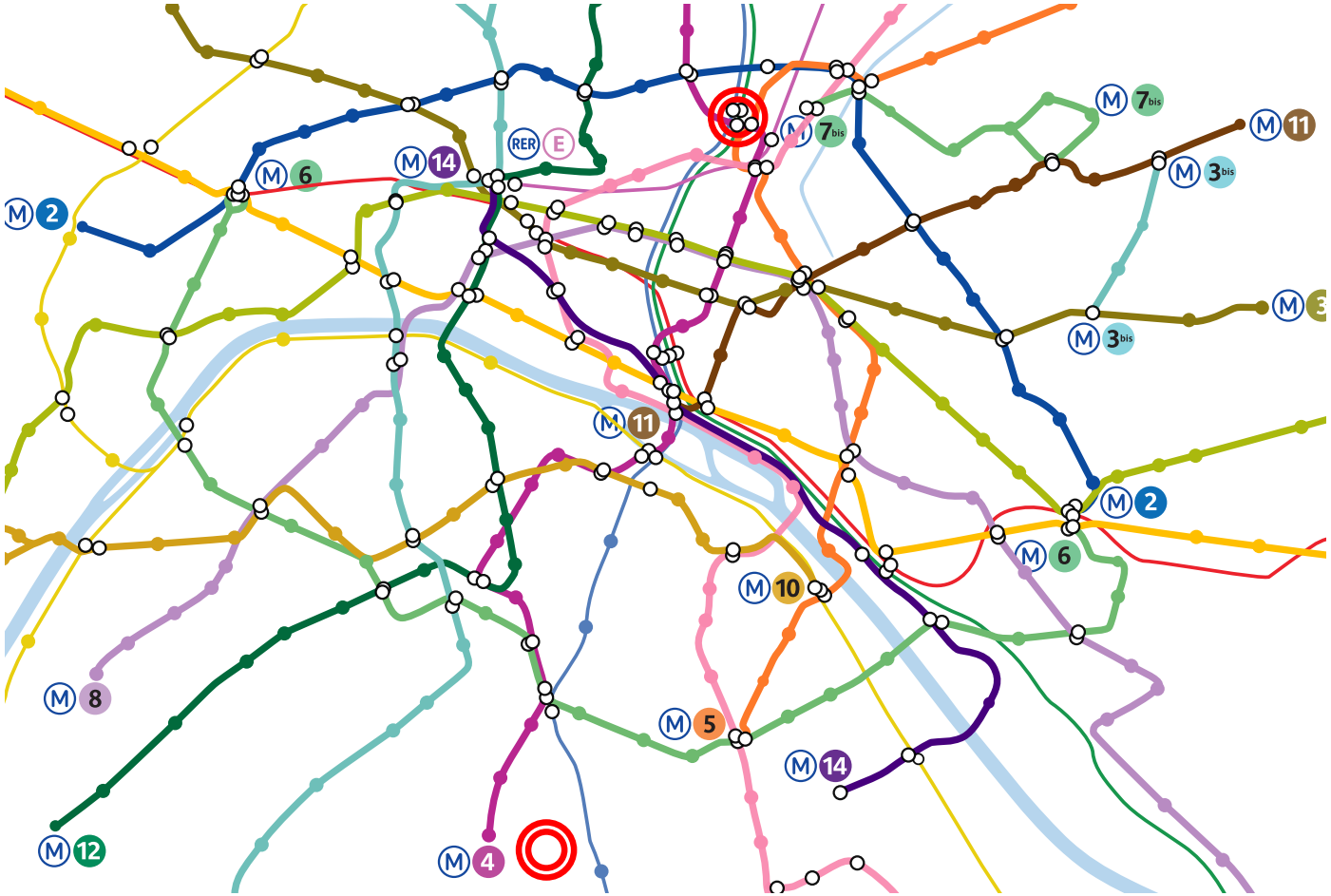
  $\forall X(rm \in X \wedge \forall xy\ (x \in X \wedge G(x, y) \rightarrow y \in X)) \rightarrow gn \in X$

Probabilistic Result

proba to be on time: **98%**

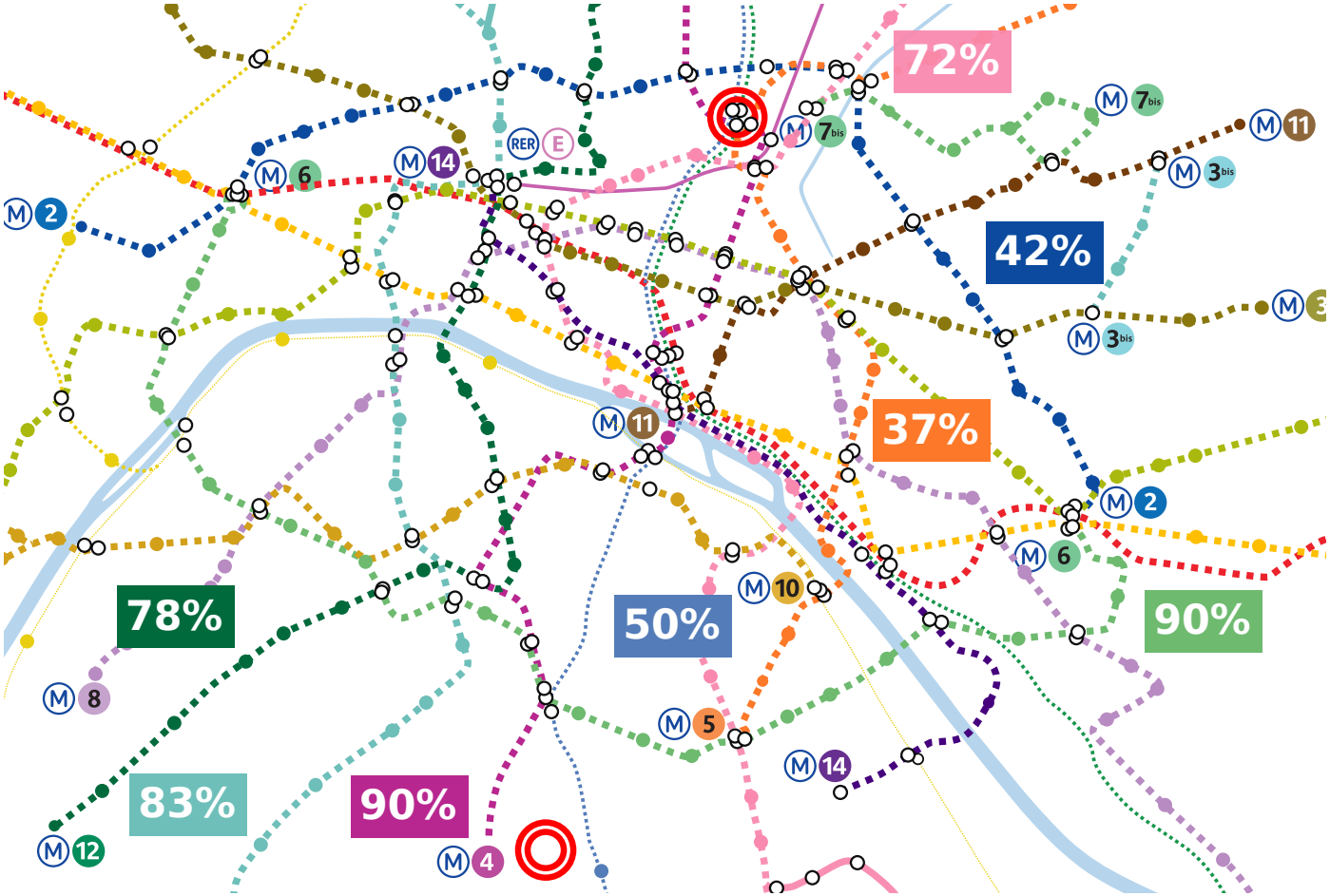
- Probability according to the input distribution

Computational complexity



- Computing paths on a large graph:
  - → Well-studied problem, efficient algorithms

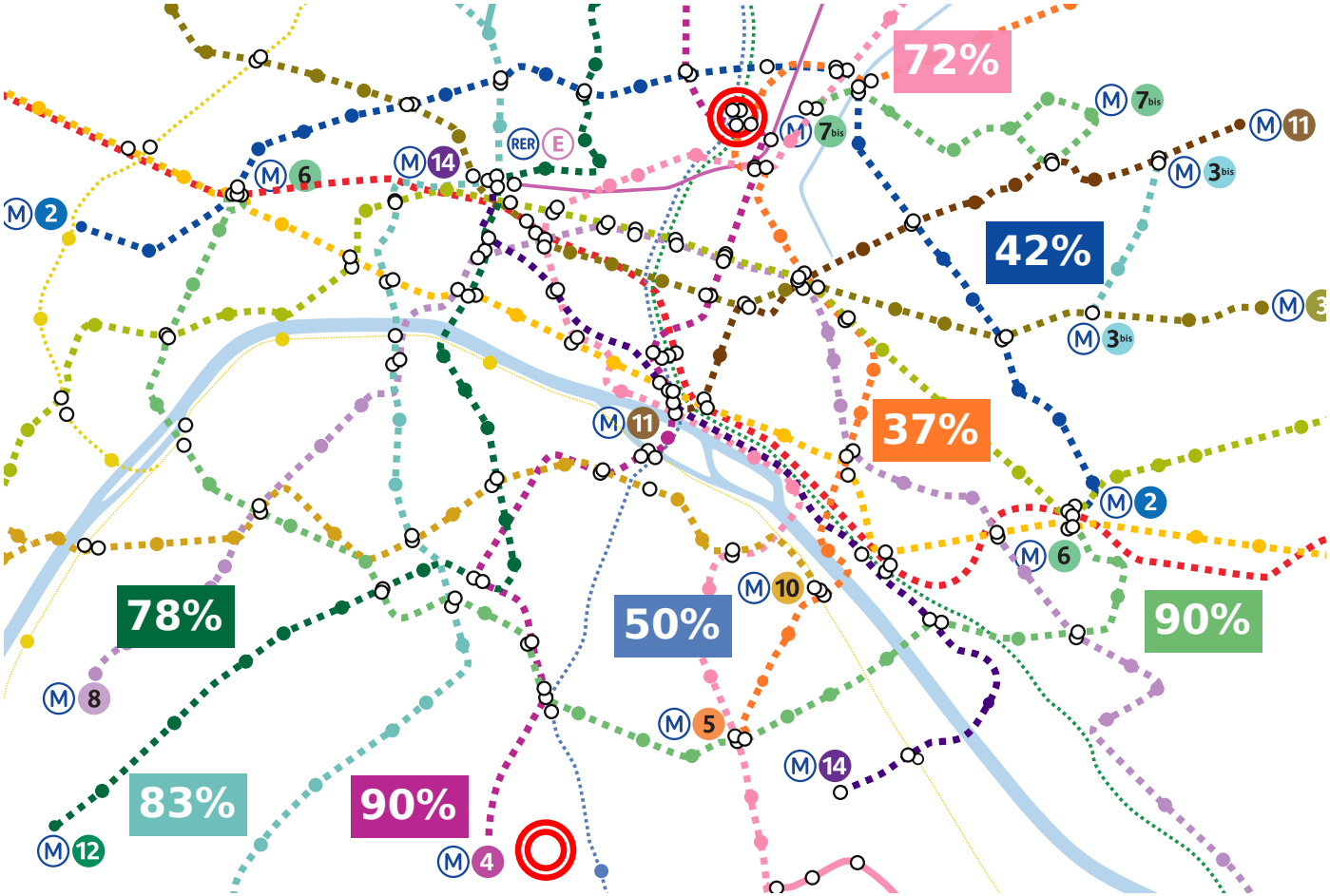
Computational complexity



- Computing paths on a large probabilistic graph:

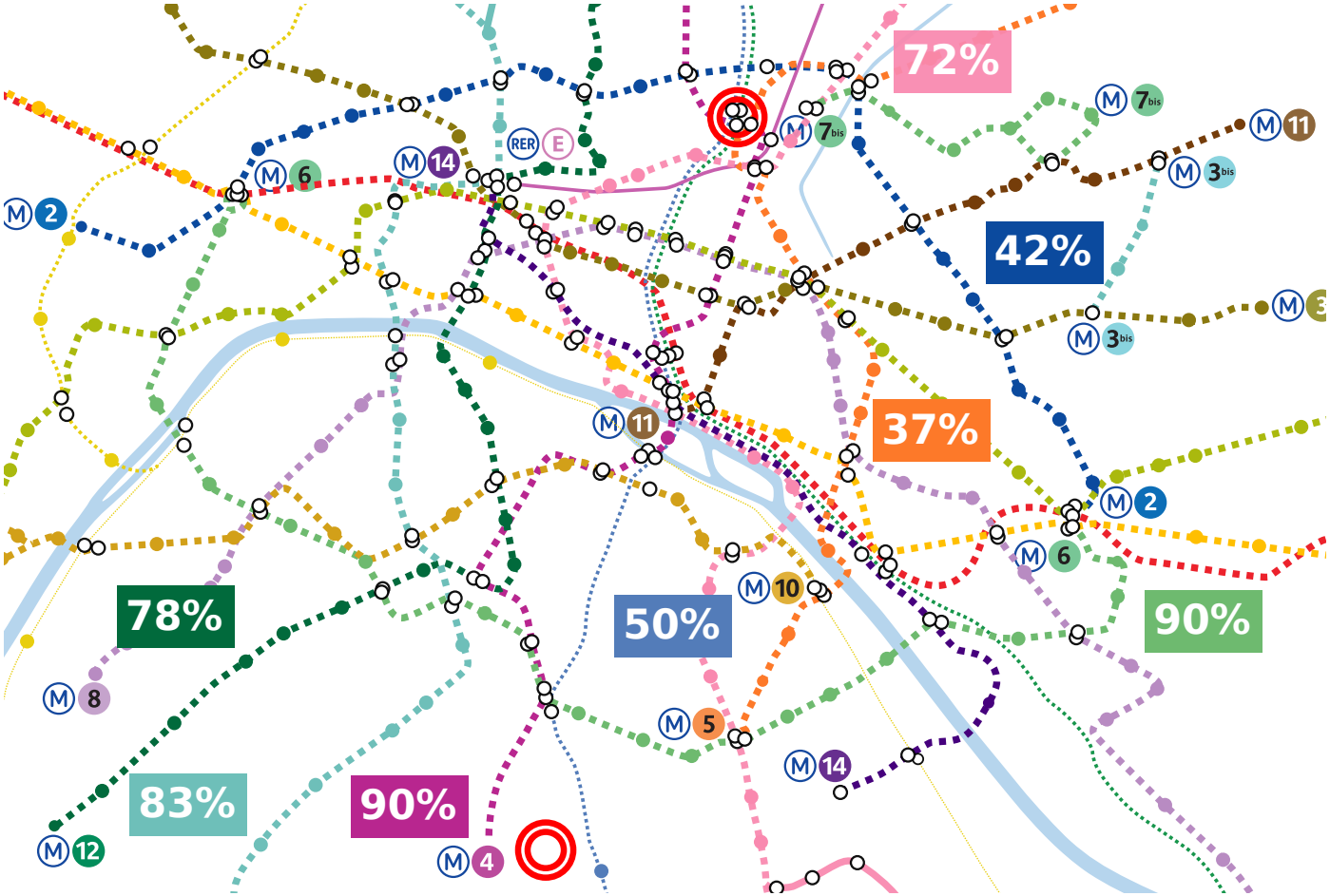
  → ???

Computational complexity



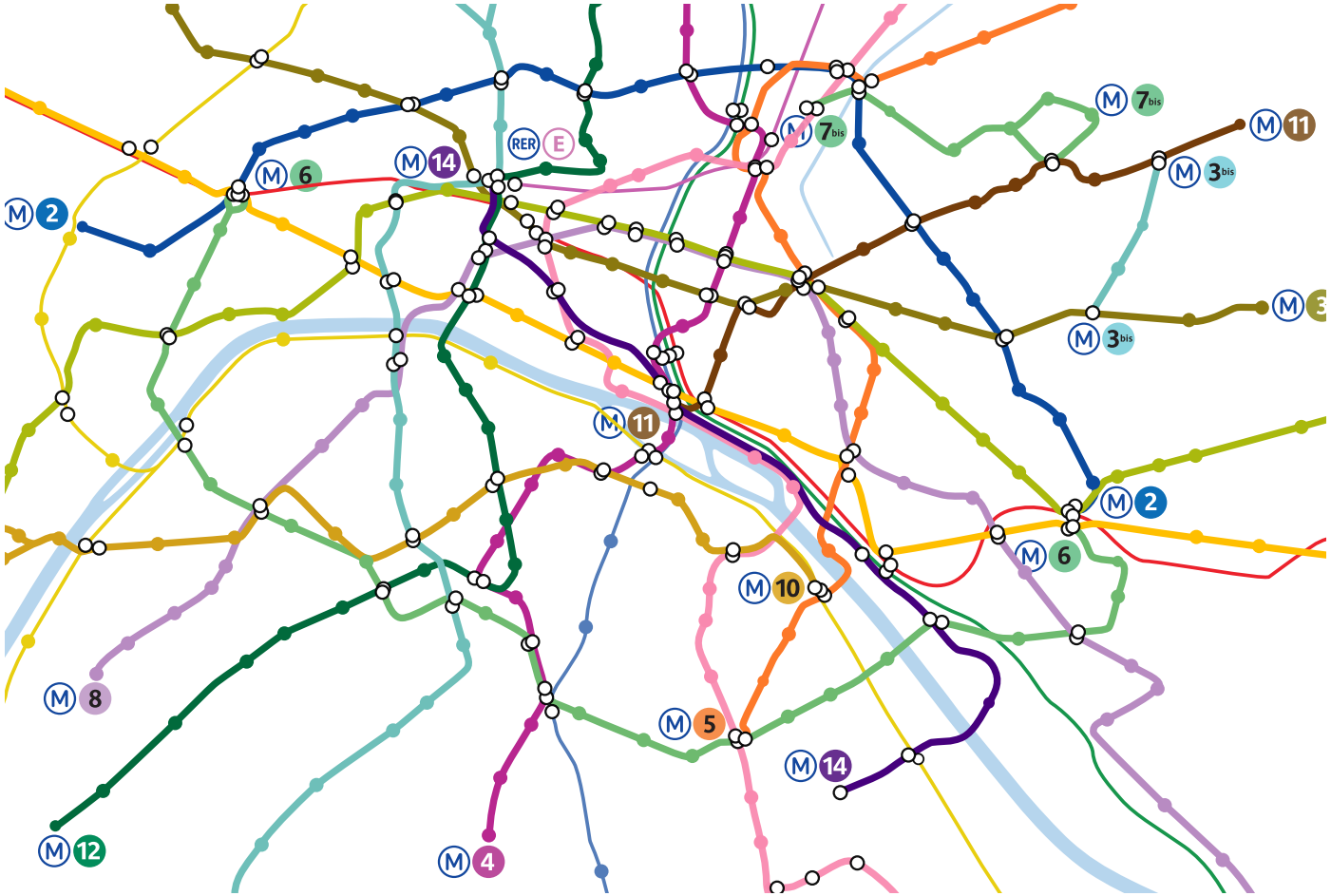
- Computing paths on a large probabilistic graph:
  - → Exponential number of possibilities

Computational complexity

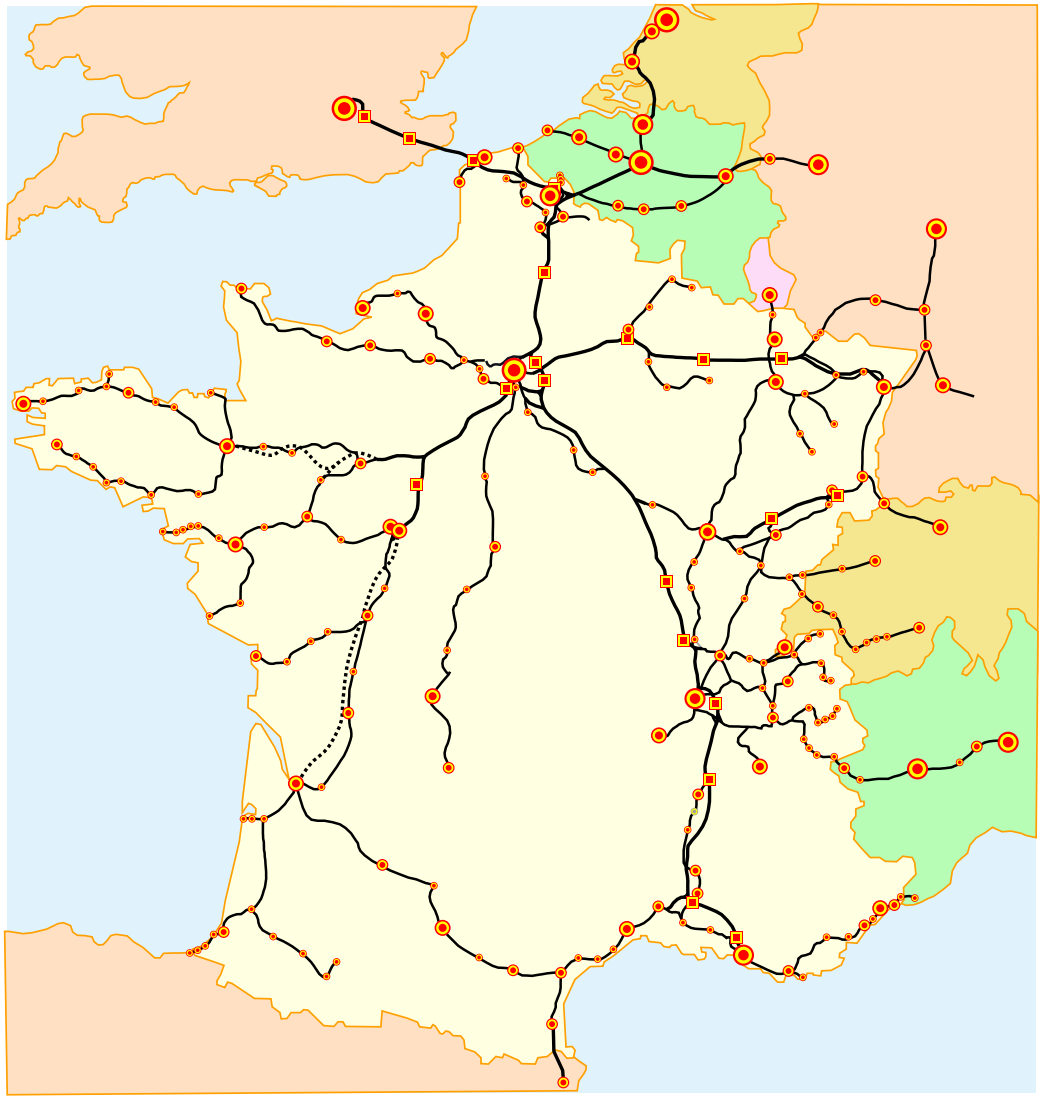


- Computing paths on a large probabilistic graph:
  - → Exponential number of possibilities
  - → #P-hard computational complexity in the database

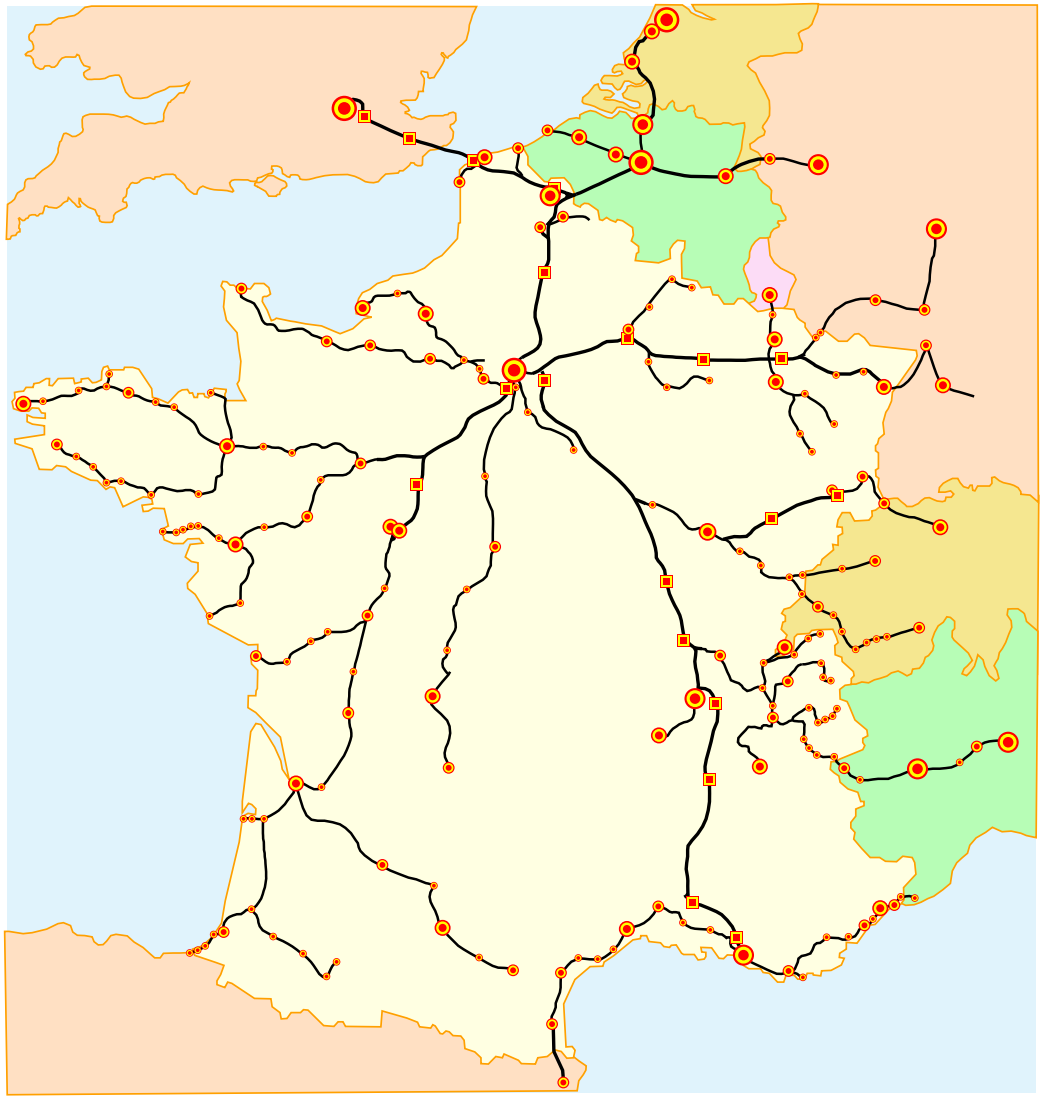
Idea: use the structure of data



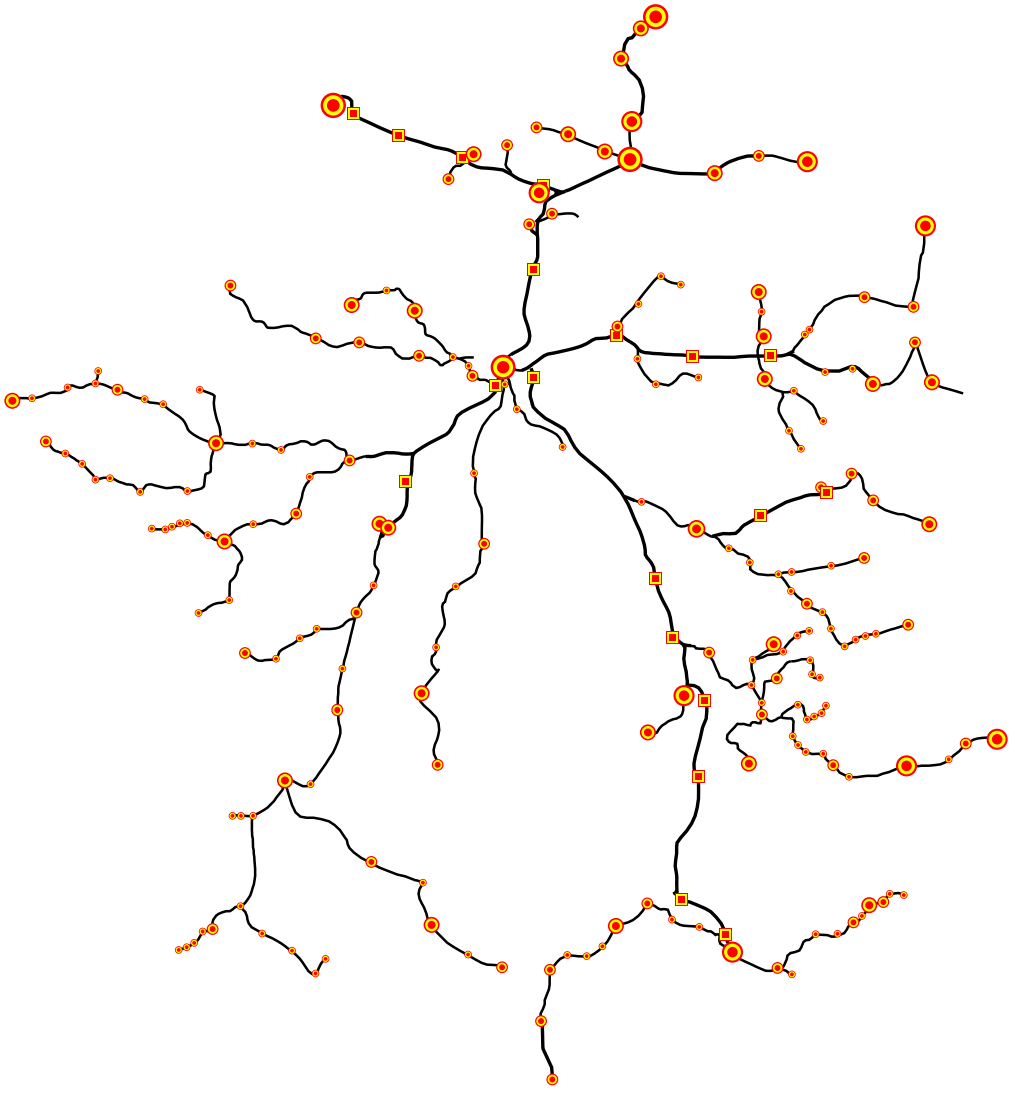
Idea: use the structure of data



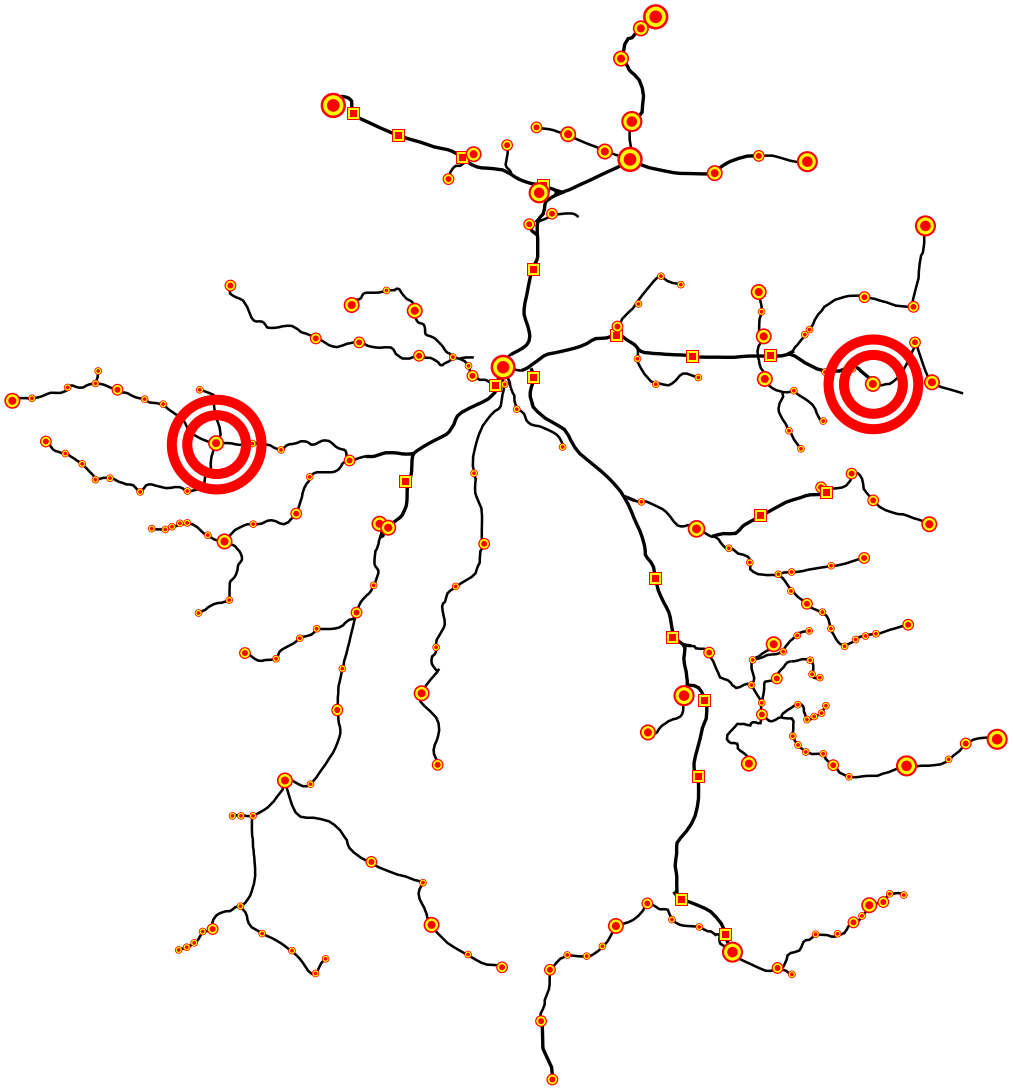
Idea: use the structure of data



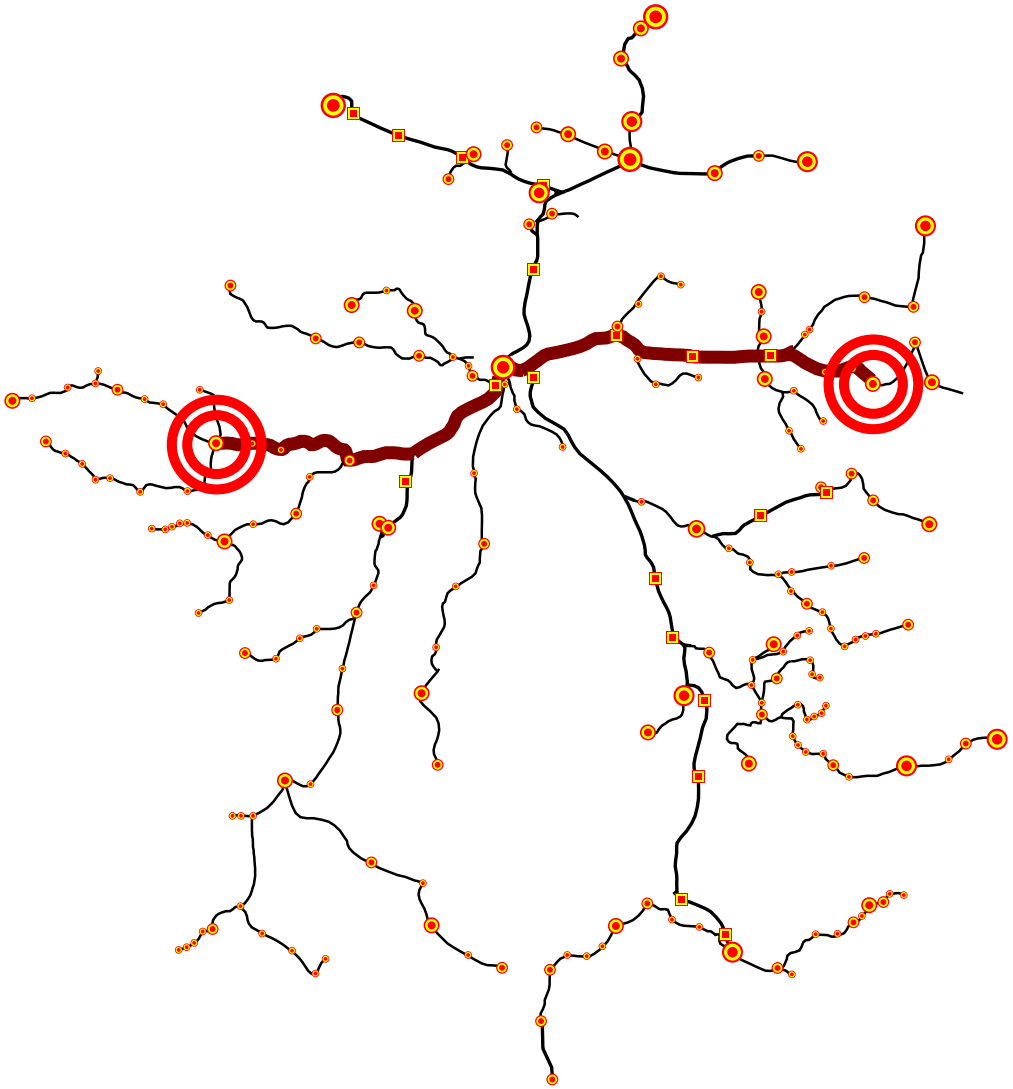
Idea: use the structure of data



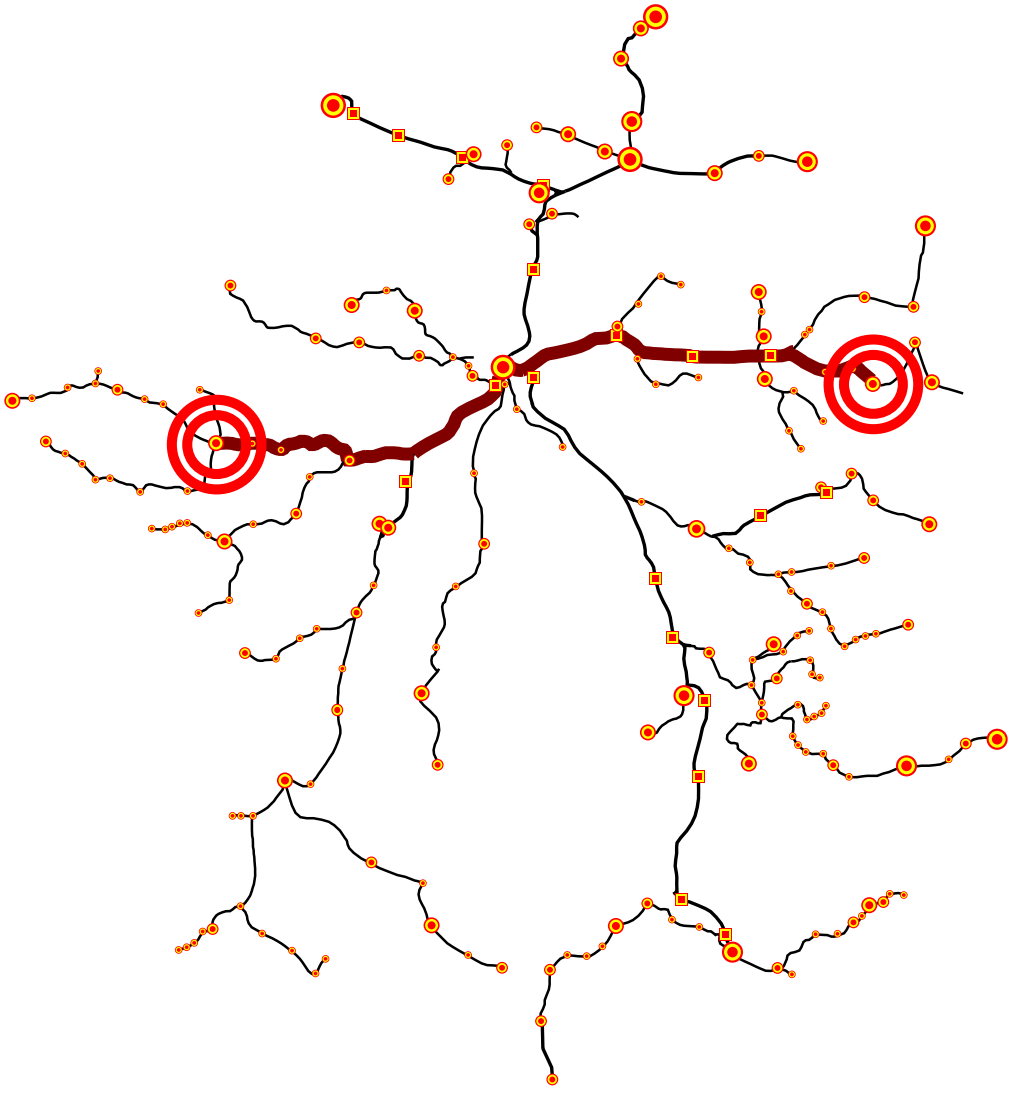
Idea: use the structure of data



Idea: use the structure of data



Idea: use the structure of data



→ Shortest path: very easy on a large tree

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"
  - Courcelle's theorem

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"
  - Courcelle's theorem
- Introduce new tools and results:

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"
  - Courcelle's theorem
- Introduce new tools and results:
  - Provenance circuits of tree automata on uncertain trees

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"
  - Courcelle's theorem
- Introduce new tools and results:
  - Provenance circuits of tree automata on uncertain trees
  - Applications to probabilistic query evaluation

Leveraging the structure of uncertain data

Does query evaluation on probabilistic data have lower complexity when the structure of the data is close to a tree?

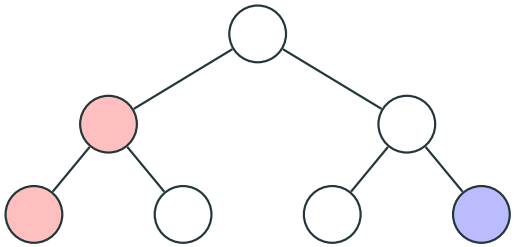
In this talk:

- Existing results on non-probabilistic data:
  - Tree automata, to evaluate queries on trees
  - Treewidth, formalizes the notion of being "close to a tree"
  - Courcelle's theorem
- Introduce new tools and results:
  - Provenance circuits of tree automata on uncertain trees
  - Applications to probabilistic query evaluation
- Other applications: Counting, enumeration, provenance...

Table of contents

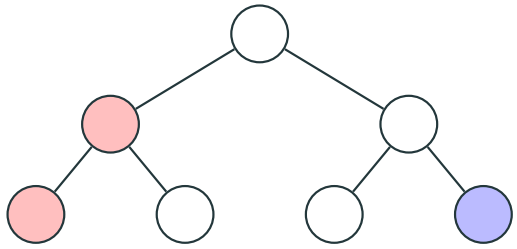
Non-probabilistic query evaluation on trees

Database: a tree T where nodes have a color from an alphabet ○ ● ●



Non-probabilistic query evaluation on trees

Database: a tree T where nodes have a color from an alphabet ○ ● ●



Query Q: a sentence in monadic second-order logic (MSO)

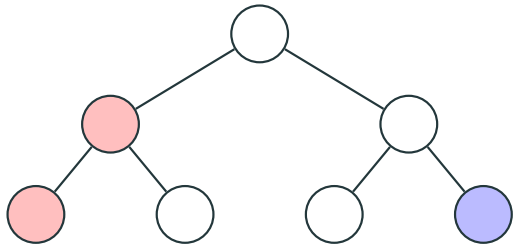
- $P_{\bullet}(x)$ means "x is blue"
- $x \to y$ means "x is the parent of y"

"Is there both a pink and a blue node?"

$\exists x\, y\; P_{\bullet}(x) \wedge P_{\bullet}(y)$

Non-probabilistic query evaluation on trees

Database: a tree T where nodes have a color from an alphabet ○ ● ●



Query Q: a sentence in monadic second-order logic (MSO)

- $P_{●}(x)$ means "x is blue"
- $x \to y$ means "x is the parent of y"

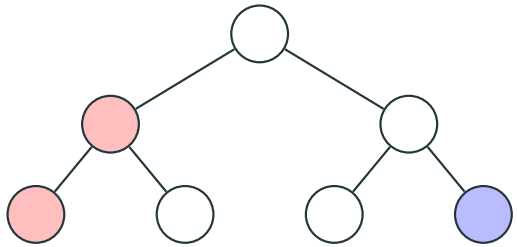
"Is there both a pink and a blue node?"

$\exists x\, y\; P_{●}(x) \wedge P_{●}(y)$

Result: TRUE/FALSE indicating if the tree T satisfies the query Q

Non-probabilistic query evaluation on trees

Database: a tree T where nodes have a color from an alphabet ○ ● ●



Query Q: a sentence in monadic second-order logic (MSO)

- $P_{\bullet}(x)$ means "x is blue"
- $x \to y$ means "x is the parent of y"

"Is there both a pink and a blue node?"

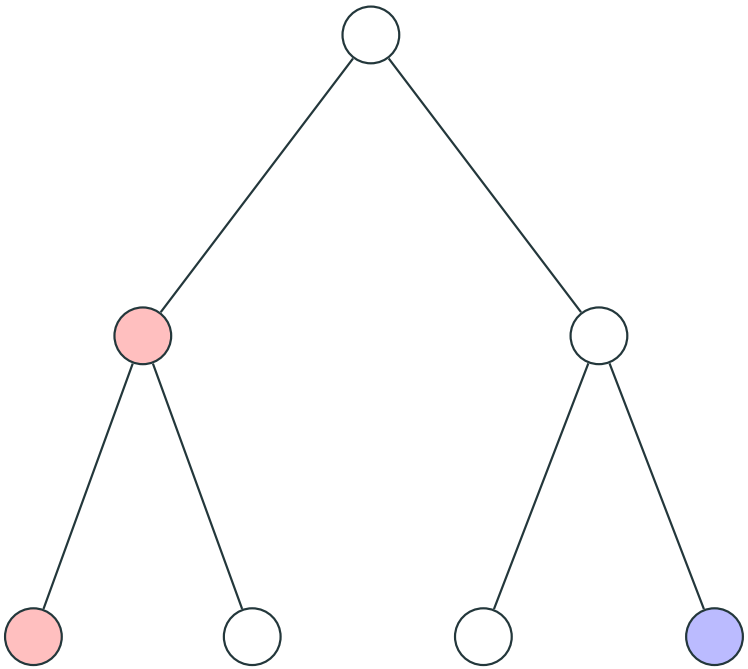
$\exists x\, y\; P_{\bullet}(x) \wedge P_{\bullet}(y)$

Result: TRUE/FALSE indicating if the tree T satisfies the query Q

Computational complexity as a function of T
(the query Q is fixed)

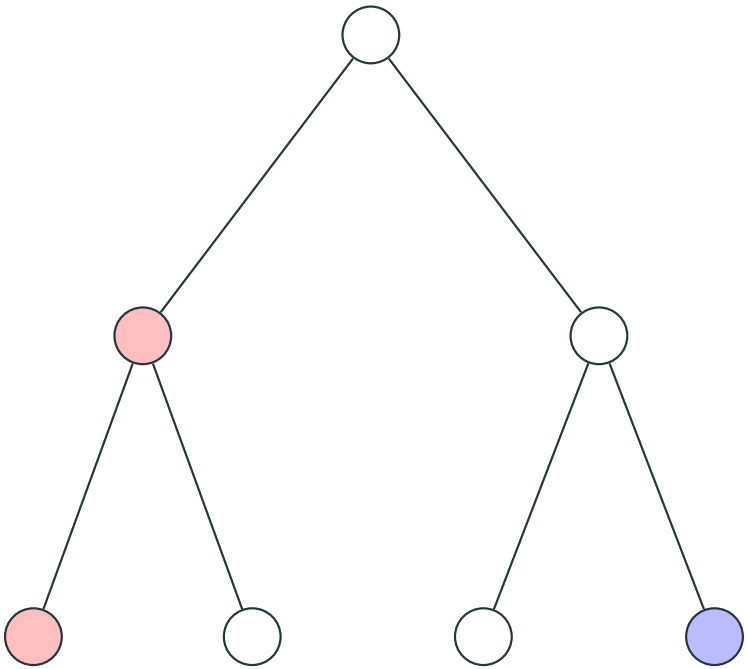
Tree automata

Tree alphabet: 



Tree automata

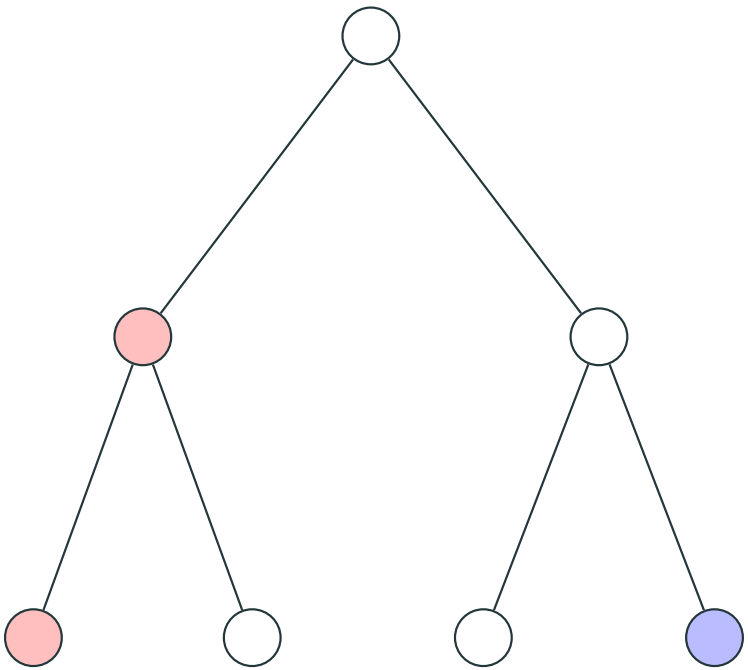
Tree alphabet: 



- Bottom-up deterministic tree automaton
- *“Is there both a pink and a blue node?”*

Tree automata

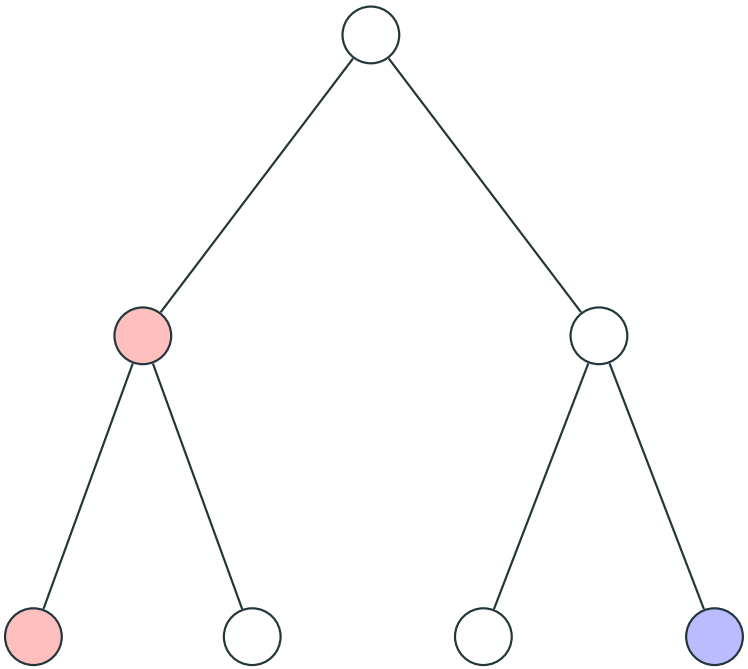
Tree alphabet: ○ ● ●



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$

Tree automata

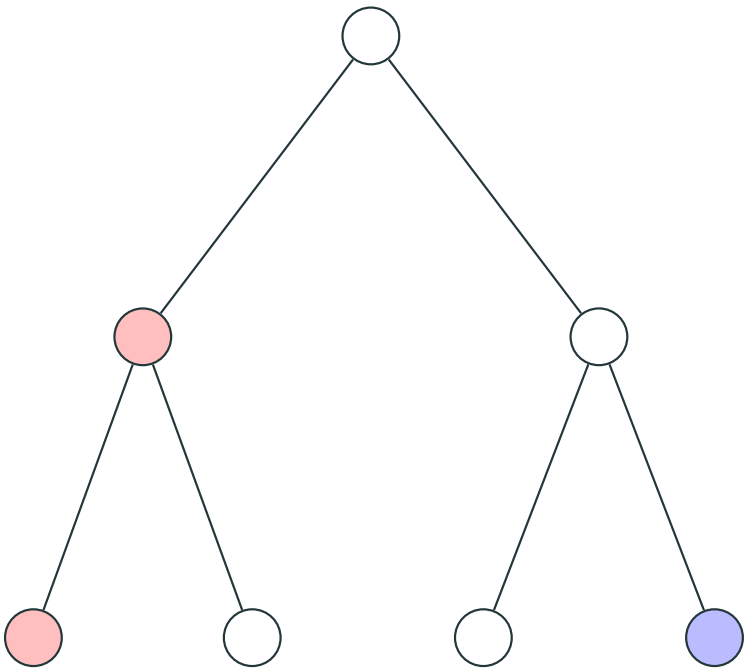
Tree alphabet: 



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$

Tree automata

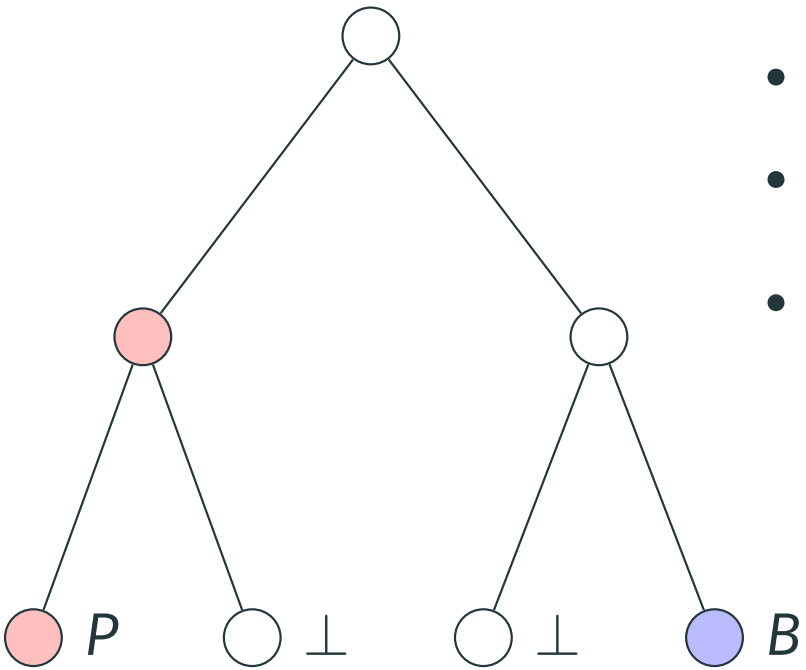
Tree alphabet: ○ ● ●



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B

Tree automata

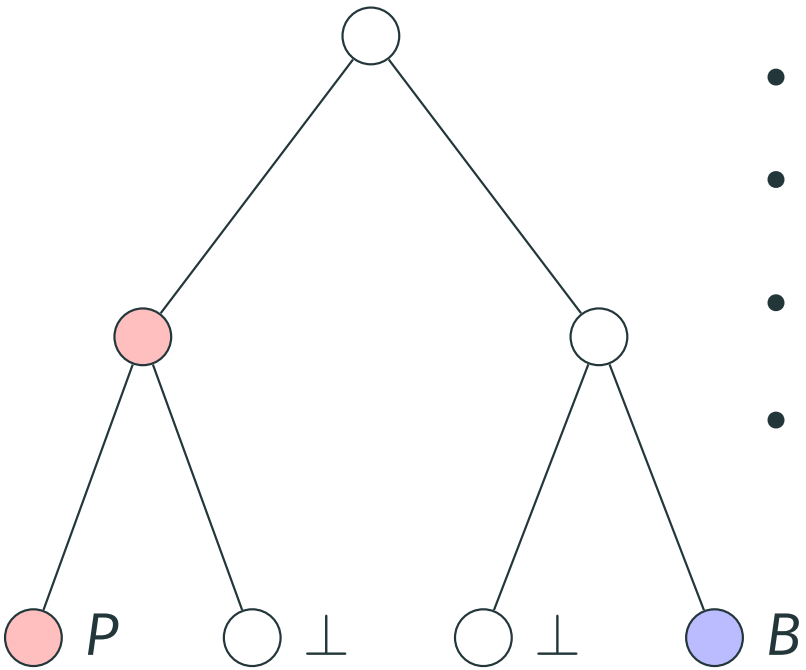
Tree alphabet: ○ ● ●



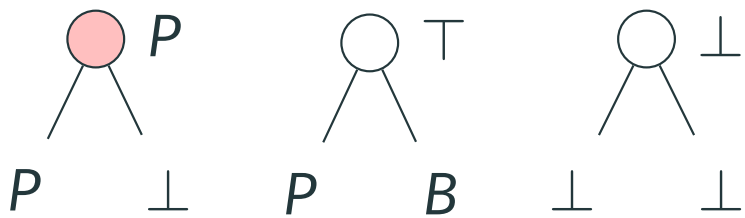
- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B

Tree automata

Tree alphabet: ○ ● ●

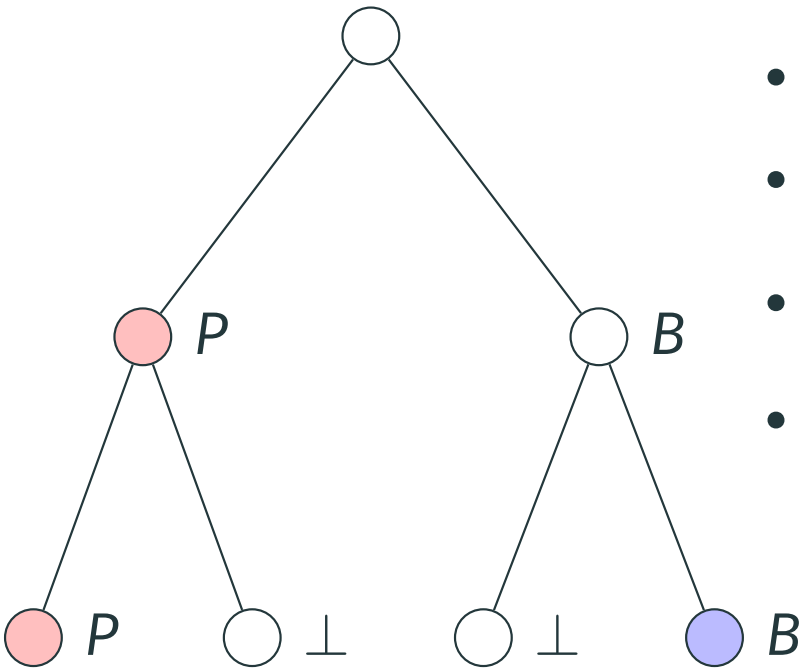


- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B
- Transitions (examples):

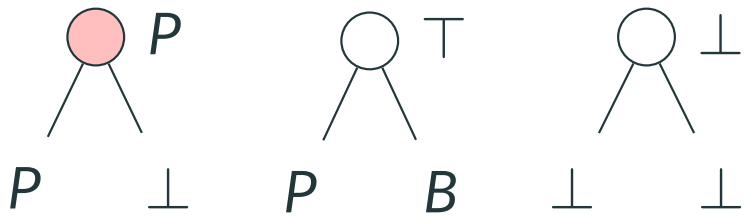


Tree automata

Tree alphabet: ○ ● ●

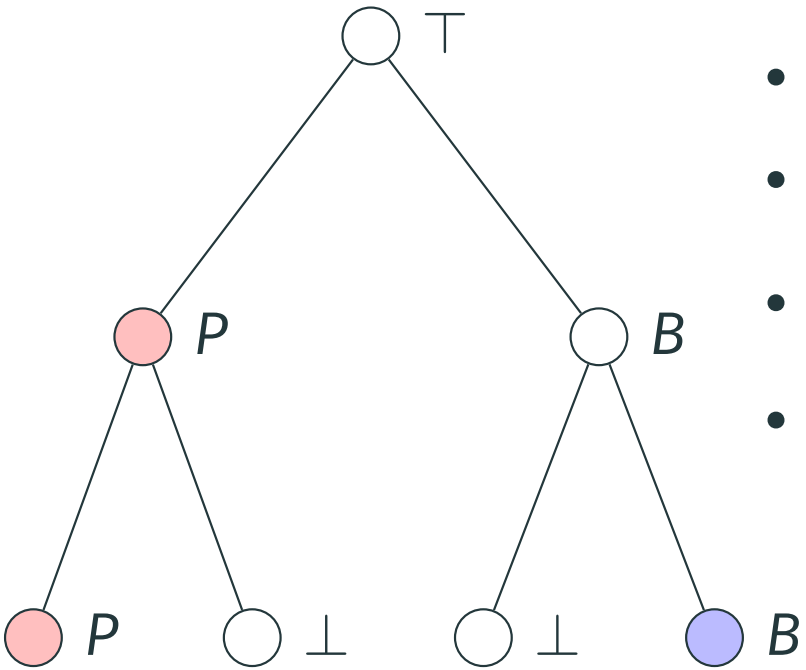


- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B
- Transitions (examples):

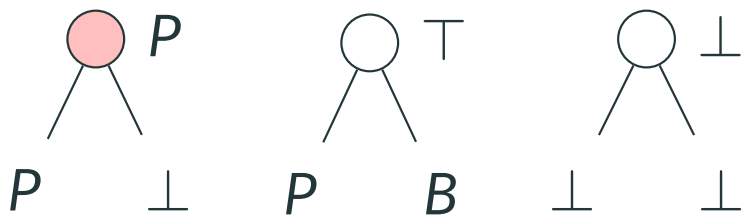


Tree automata

Tree alphabet: ○ ● ●

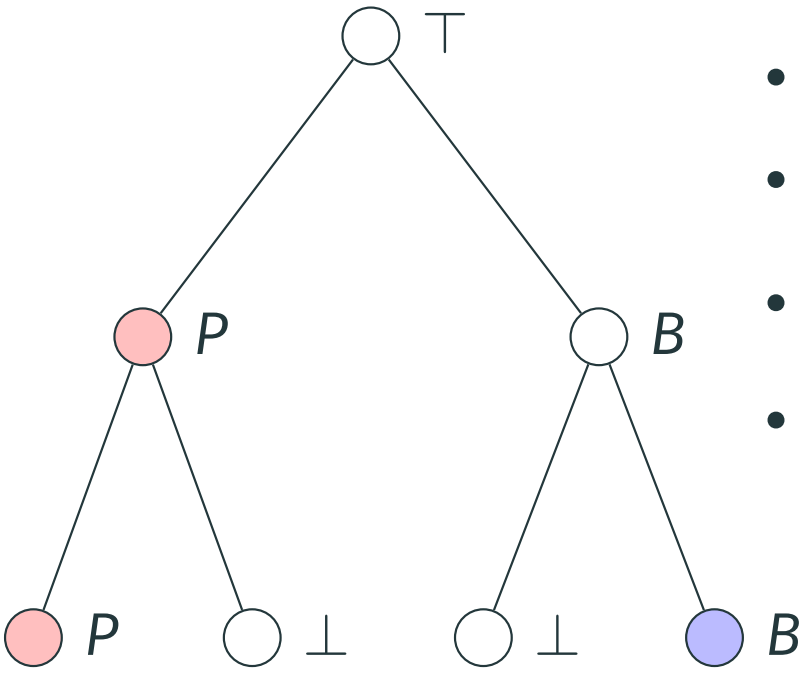


- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B
- Transitions (examples):

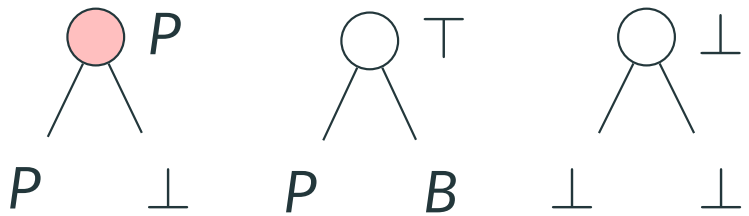


Tree automata

Tree alphabet: ○ ● ●



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B
- Transitions (examples):

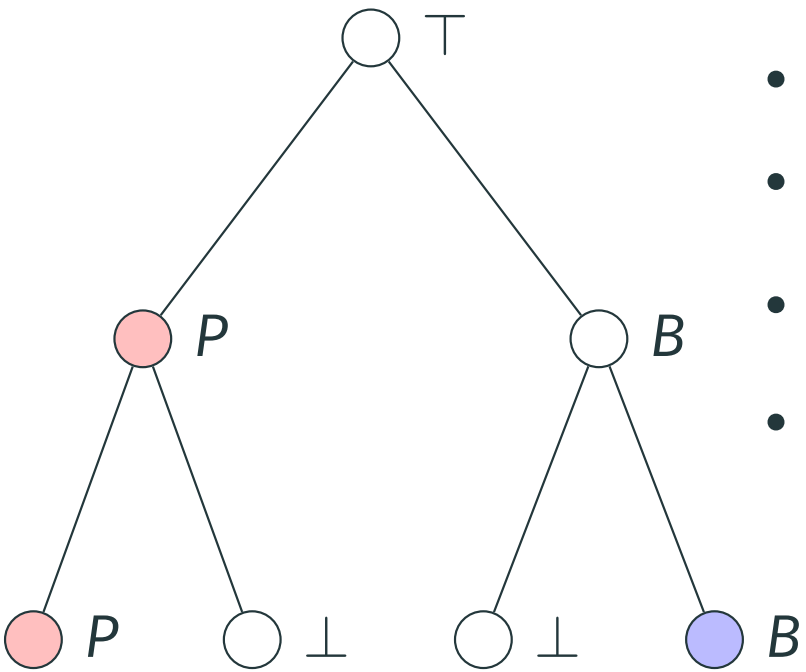


Theorem [Thatcher and Wright, 1968]

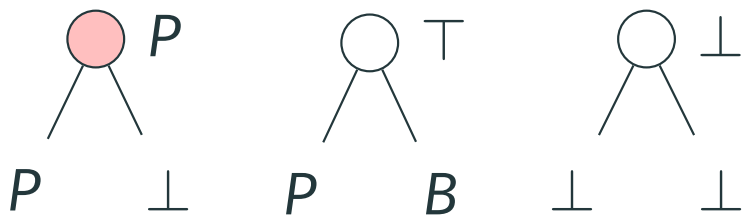
MSO and tree automata have the same expressive power on trees

Tree automata

Tree alphabet: ○ ● ●



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ ● P ● B
- Transitions (examples):



Theorem [Thatcher and Wright, 1968]

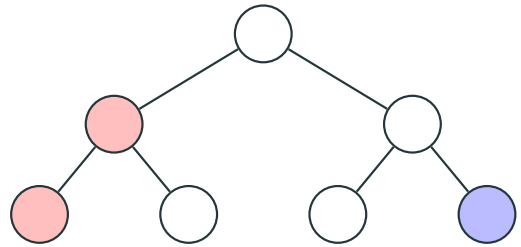
MSO and tree automata have the same expressive power on trees

Corollary

Non-probabilistic query evaluation of MSO on trees is in linear time.

Non-probabilistic query evaluation on trees

Database: a tree T where nodes have a color from an alphabet ○ ● ●



Query Q: a sentence in monadic second-order logic (MSO)

- $P_{\bullet}(x)$ means "x is blue"
- $x \to y$ means "x is the parent of y"

"Is there both a pink and a blue node?"

$\exists x\, y\; P_{\bullet}(x) \wedge P_{\bullet}(y)$

Result: TRUE/FALSE indicating if T satisfies the query Q

Computational complexity as a function of the tree T
(the query Q is fixed)

Non-probabilistic query evaluation on treelike data

Database: a treelike database T

???

Query Q: a sentence in monadic second-order logic (MSO)

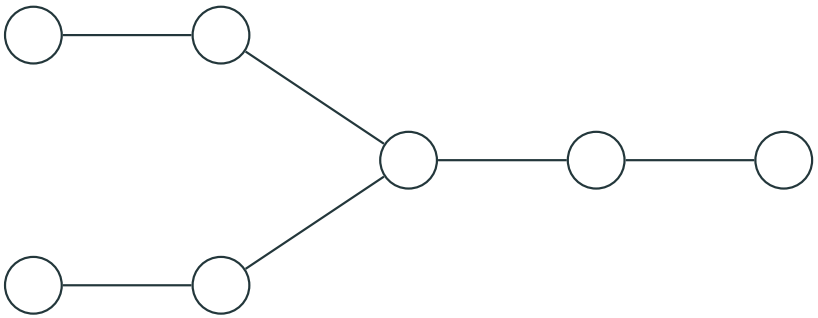
(Metro|RER)*
| (Bus|Tram)*

Result: TRUE/FALSE indicating if T satisfies the query Q

Computational complexity as a function of the tree T
(the query Q is fixed)

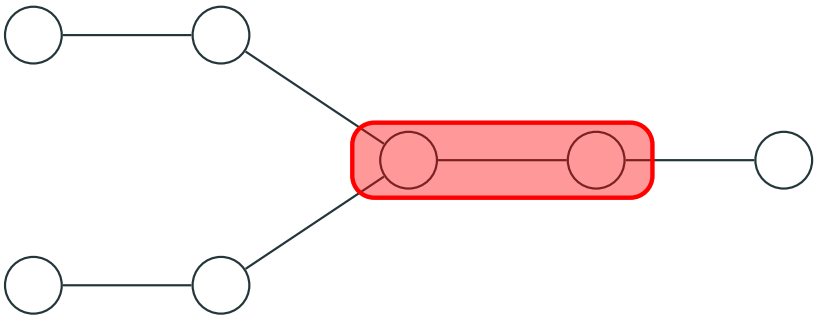
Treewidth

Treewidth by example:



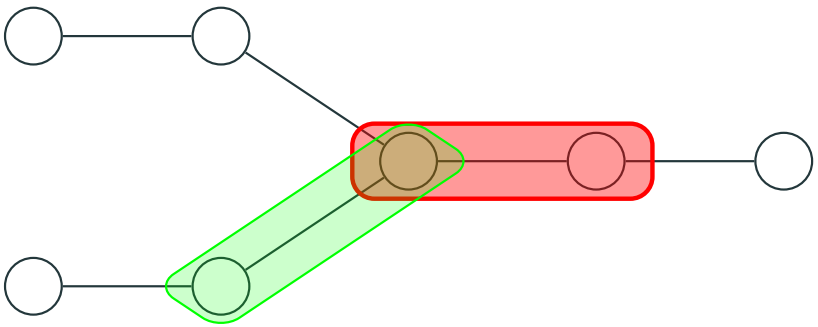
Treewidth

Treewidth by example:



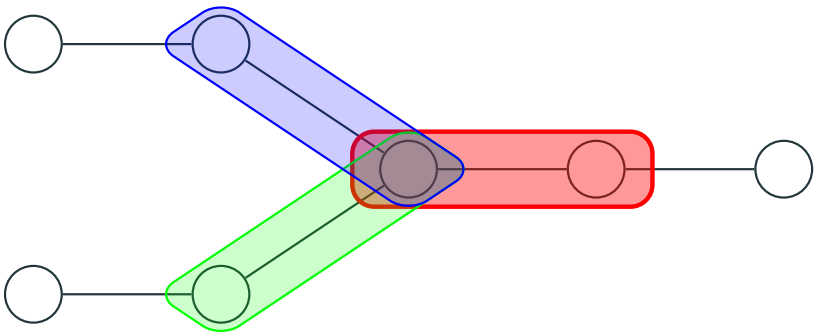
Treewidth

Treewidth by example:



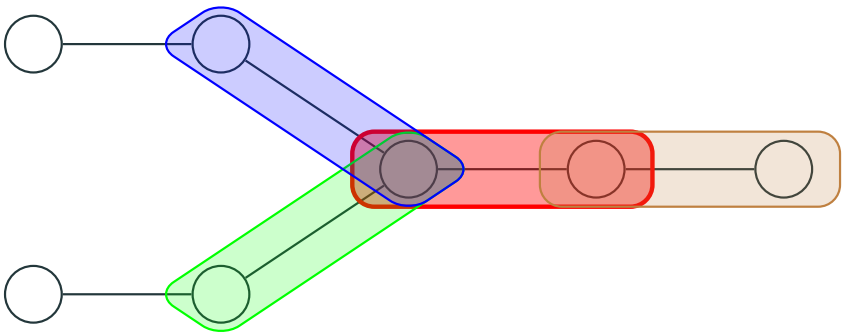
Treewidth

Treewidth by example:



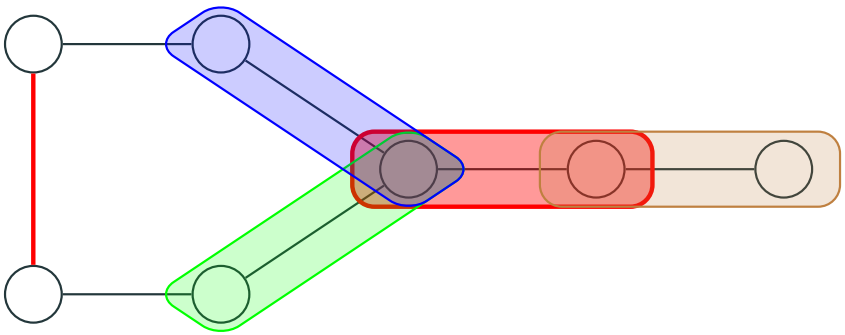
Treewidth

Treewidth by example:



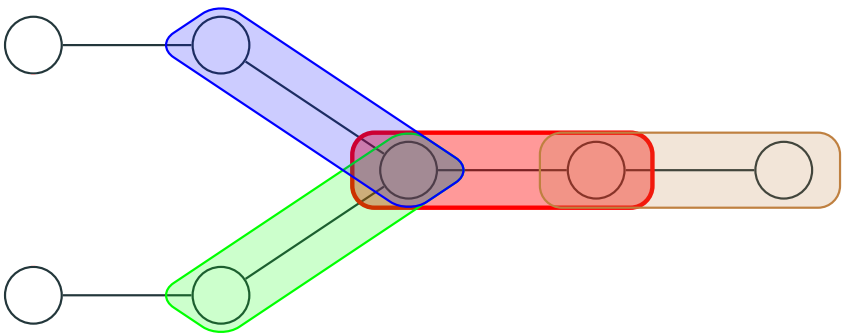
Treewidth

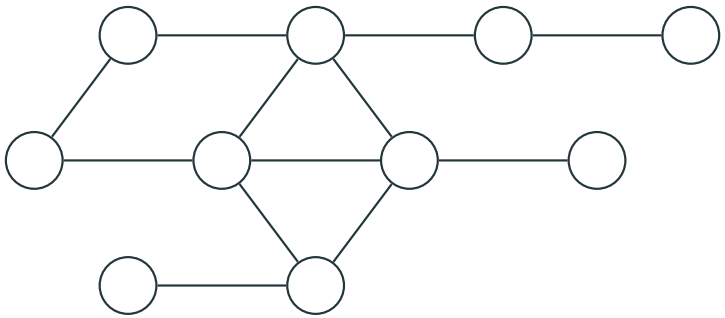
Treewidth by example:



Treewidth

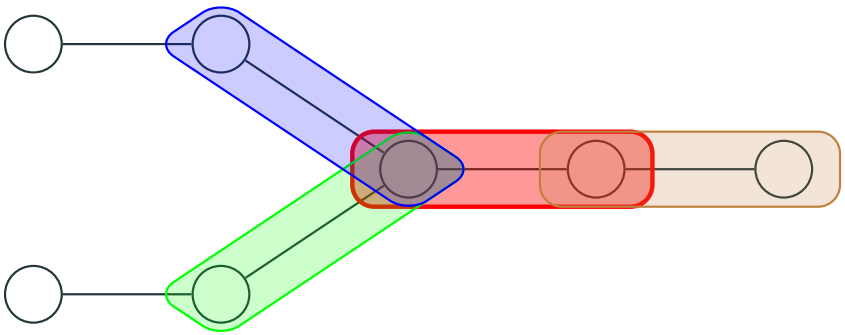
Treewidth by example:

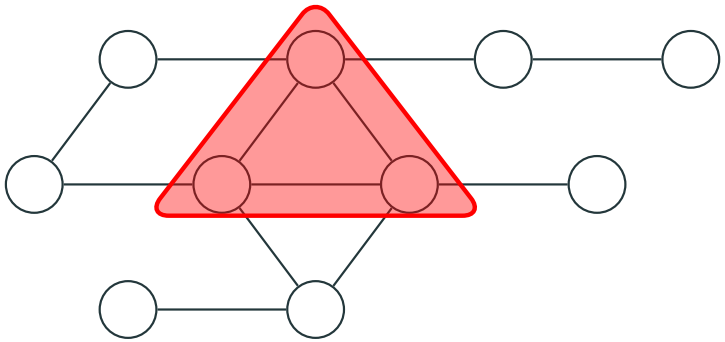




Treewidth

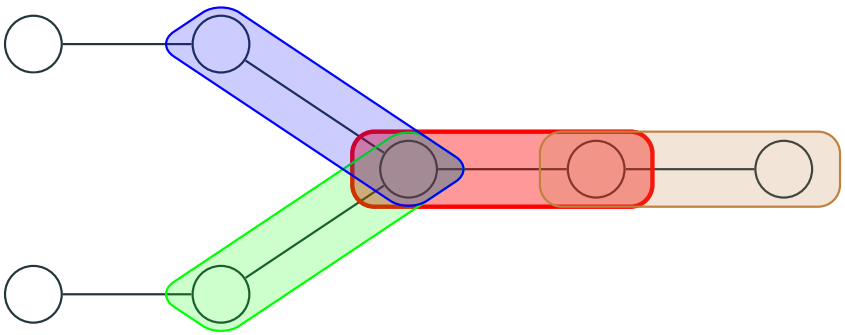
Treewidth by example:

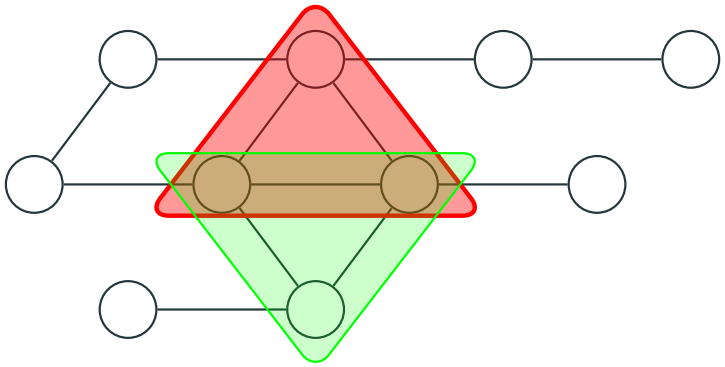




Treewidth

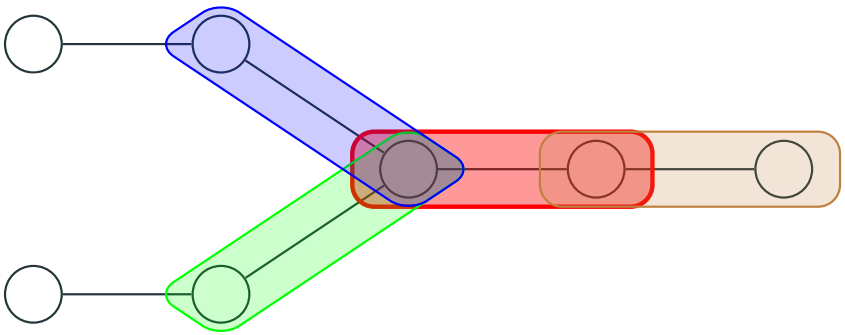
Treewidth by example:

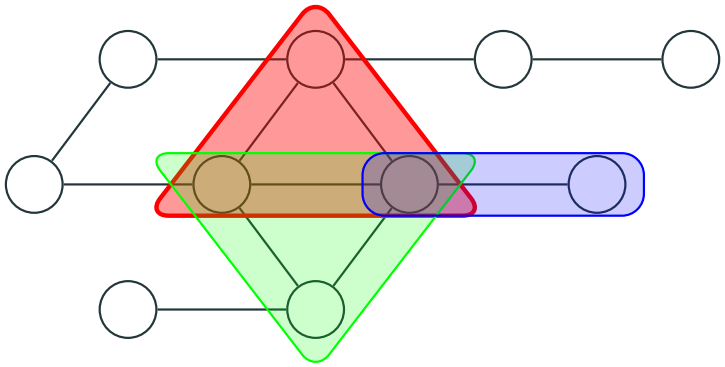




Treewidth

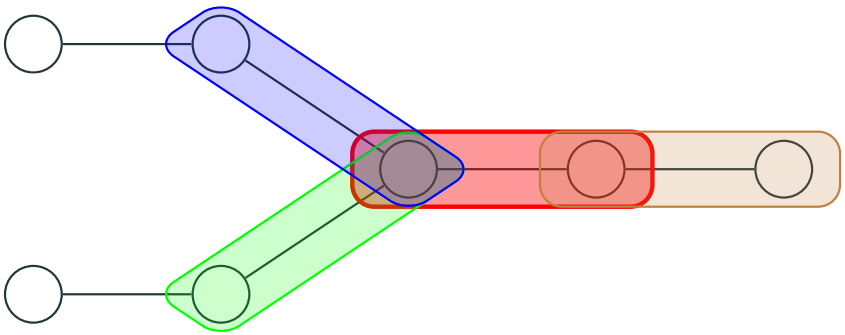
Treewidth by example:

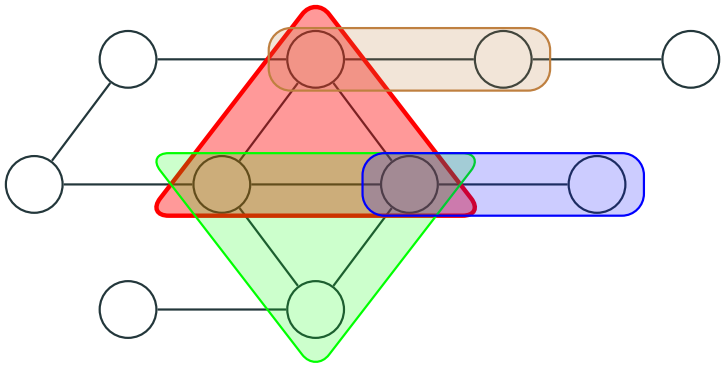




Treewidth

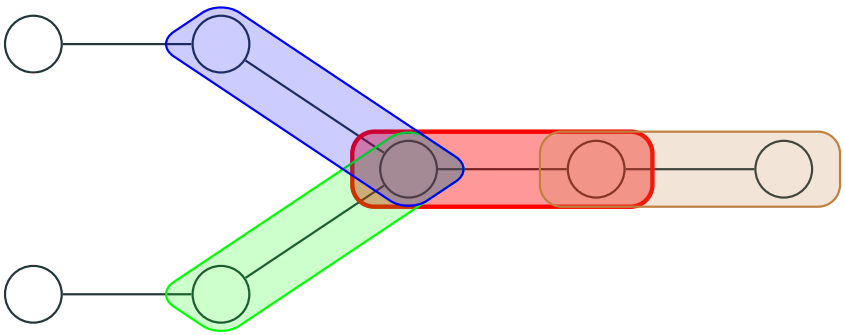
Treewidth by example:

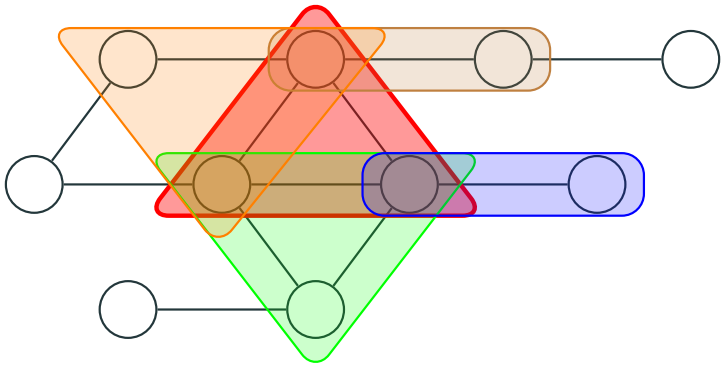




Treewidth

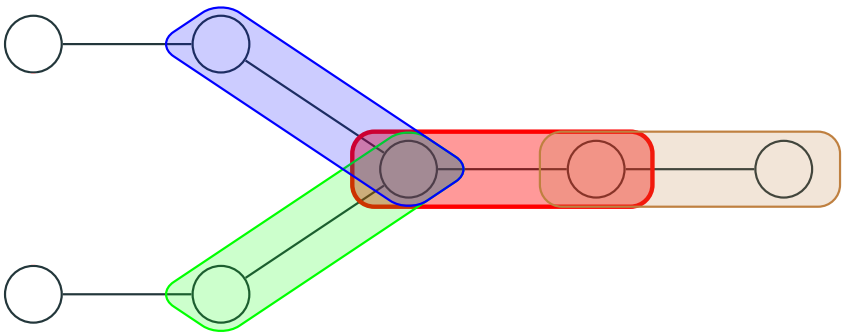
Treewidth by example:

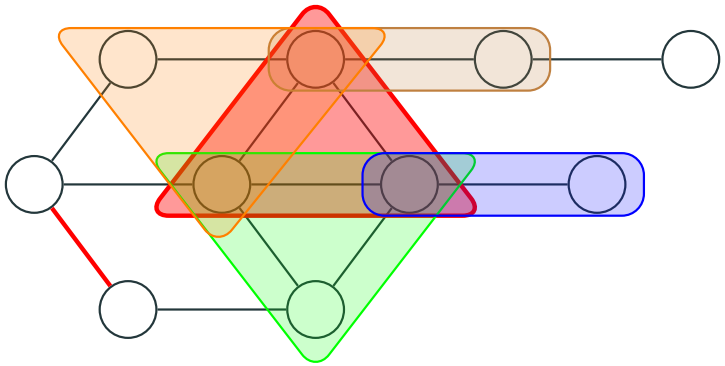




Treewidth

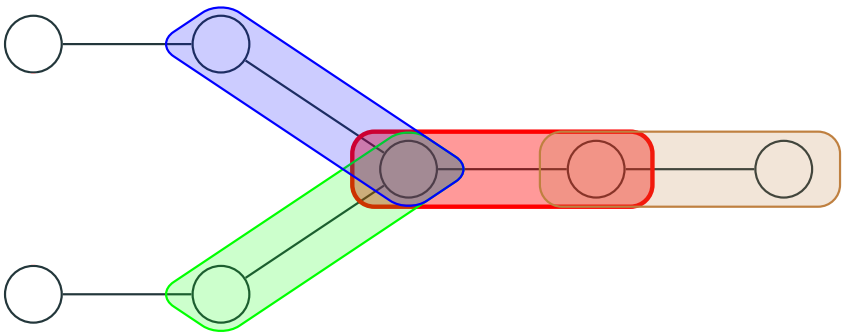
Treewidth by example:

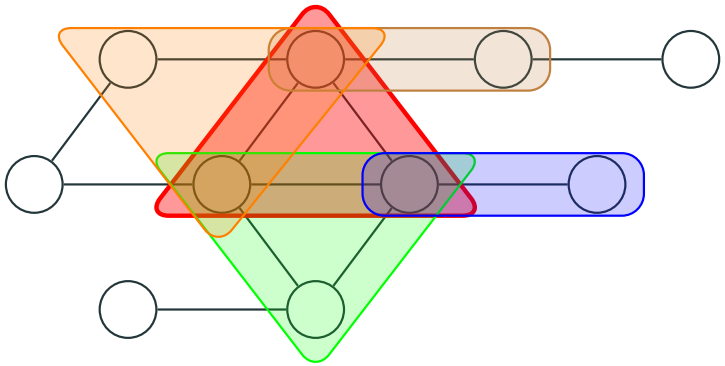




Treewidth

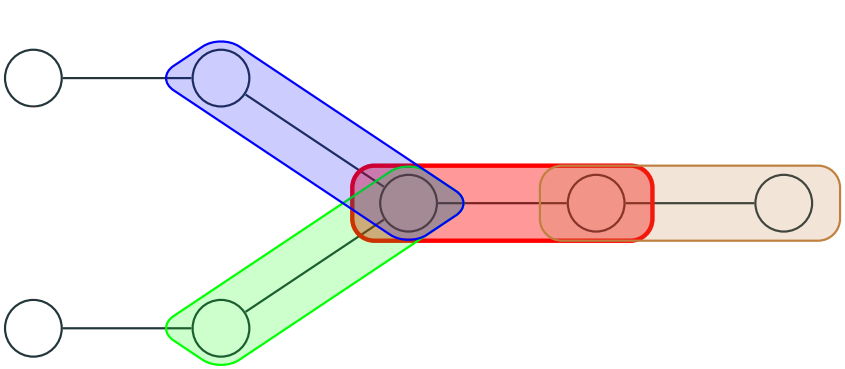
Treewidth by example:

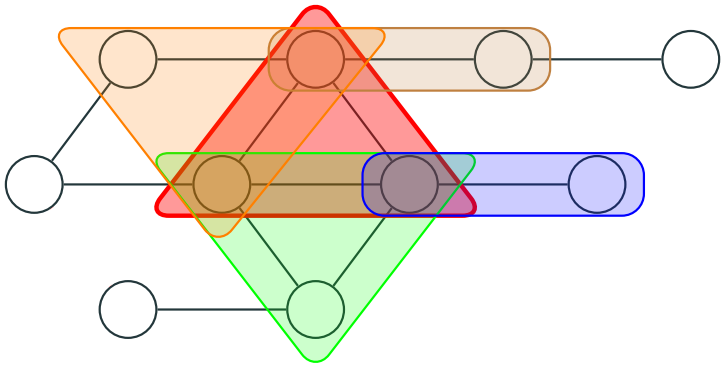




Treewidth

Treewidth by example:

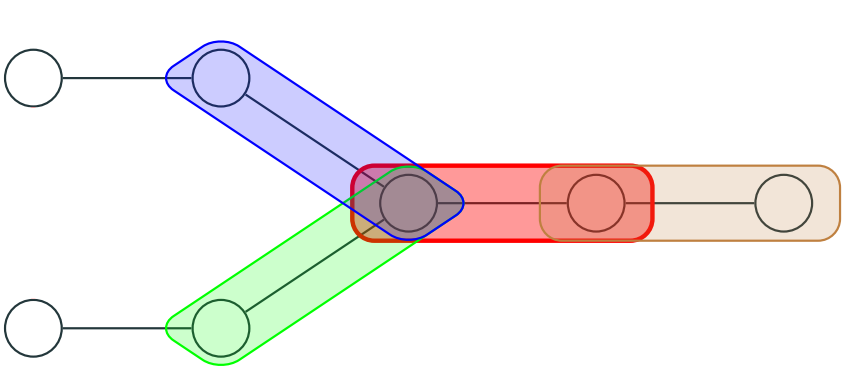


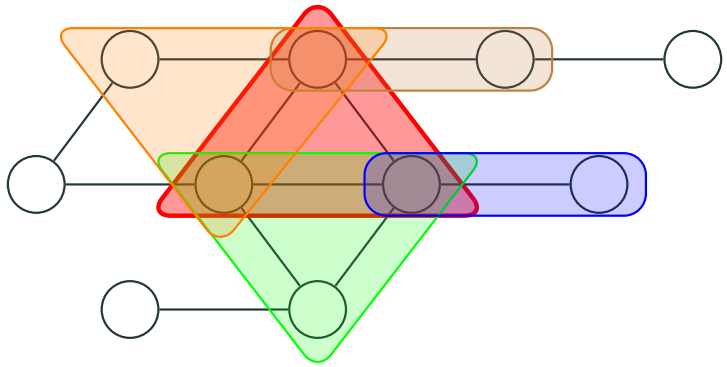


- Trees have treewidth 1
- Cycles have treewidth 2
- k-cliques and $(k-1)$-grids have treewidth $k-1$

Treewidth

Treewidth by example:



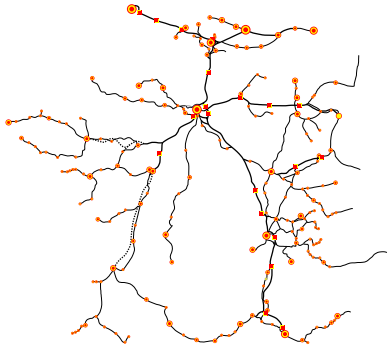


- Trees have treewidth 1
- Cycles have treewidth 2
- k-cliques and $(k-1)$-grids have treewidth $k-1$

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

Courcelle's theorem

Treelike **data**

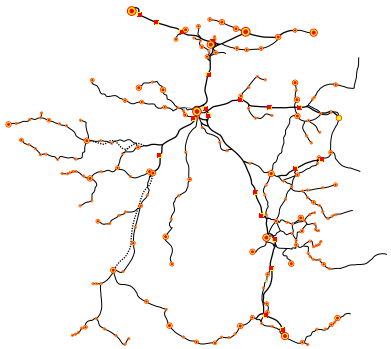


MSO query

(RER|metro)*
|(bus|tram)*

Courcelle's theorem

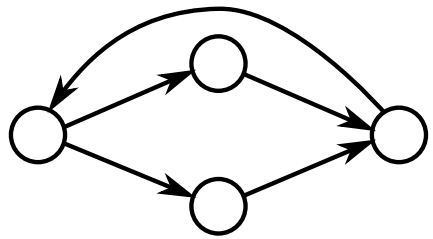
Treelike **data**



MSO query

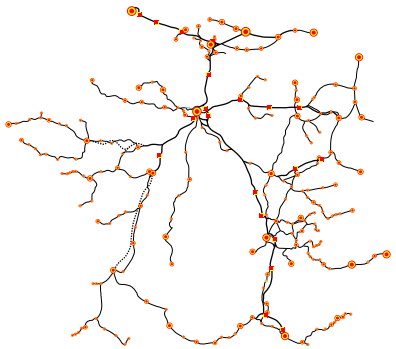
(RER|metro)*
|(bus|tram)*

Tree **automaton**



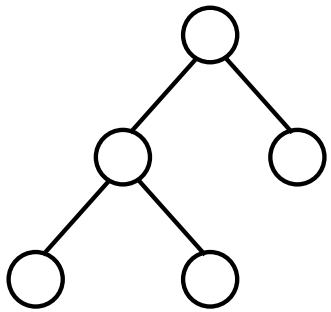
Courcelle's theorem

Treelike **data**



linear

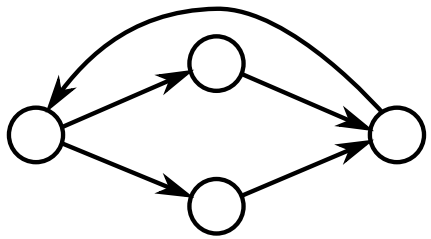
Tree **encoding**



MSO query

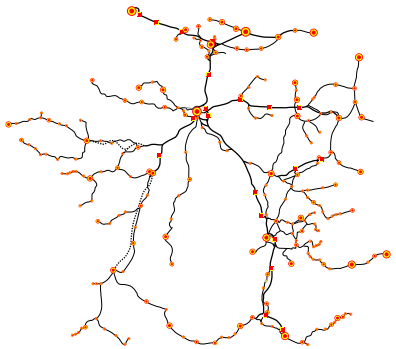
(RER|metro)*
|(bus|tram)*

Tree **automaton**



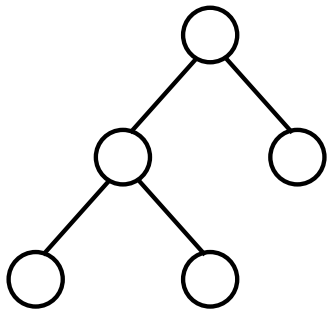
Courcelle's theorem

Treelike **data**



linear

Tree **encoding**



linear

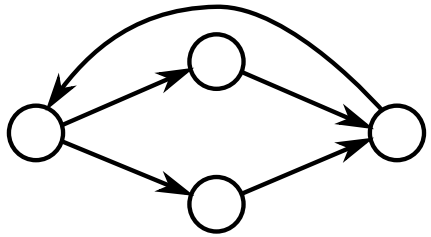
Query **answer**

TRUE

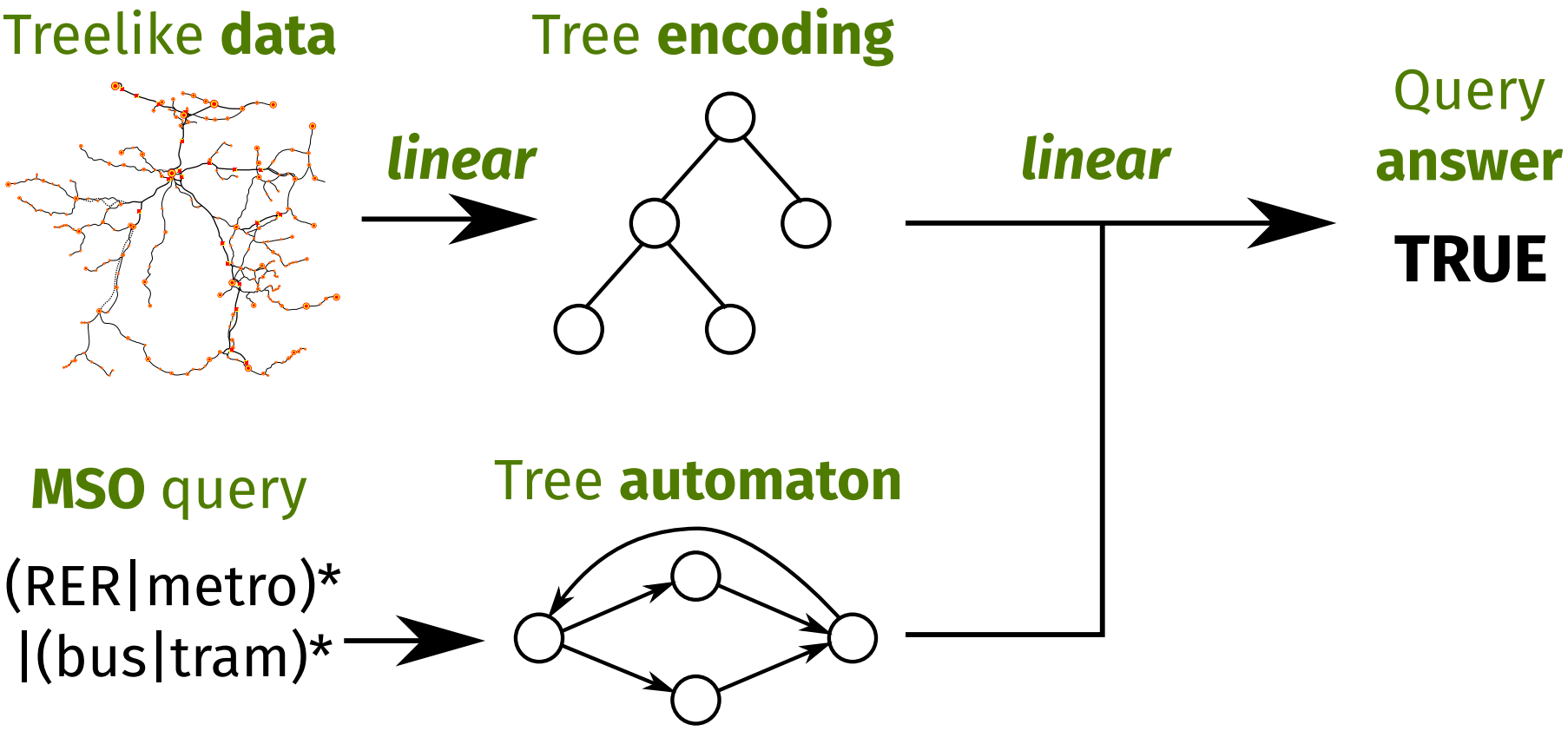
MSO query

(RER|metro)*
|(bus|tram)*

Tree **automaton**



Courcelle's theorem



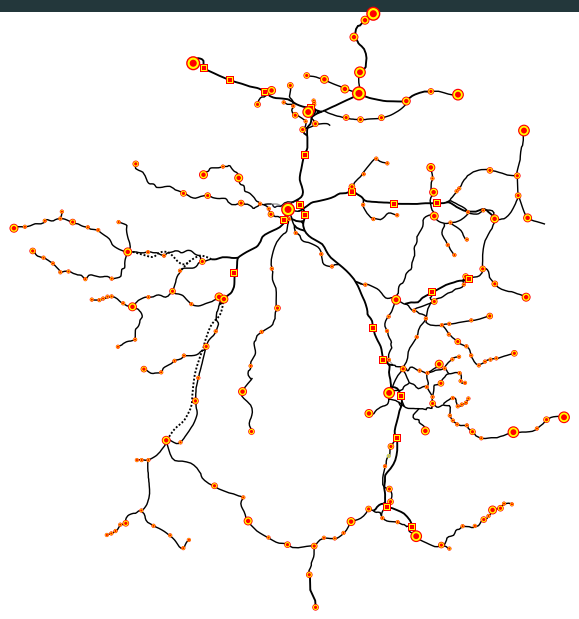
Theorem [Courcelle, 1990]

For any fixed Boolean MSO query Q and $k \in \mathbb{N}$,
given a database D of treewidth $\leq k$,
we can compute in linear time in D whether D satisfies Q

Table of contents

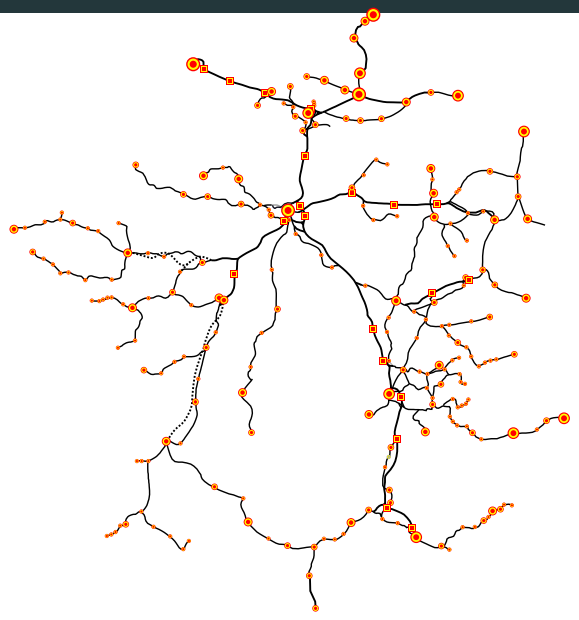
Probabilistic query evaluation on treelike data

- Database D with treewidth $\leq k$ for some constant k
- Probability of each fact of D to be actually present in the data (independently from other facts)



Probabilistic query evaluation on treelike data

- Database D with treewidth $\leq k$ for some constant k
- Probability of each fact of D to be actually present in the data (independently from other facts)

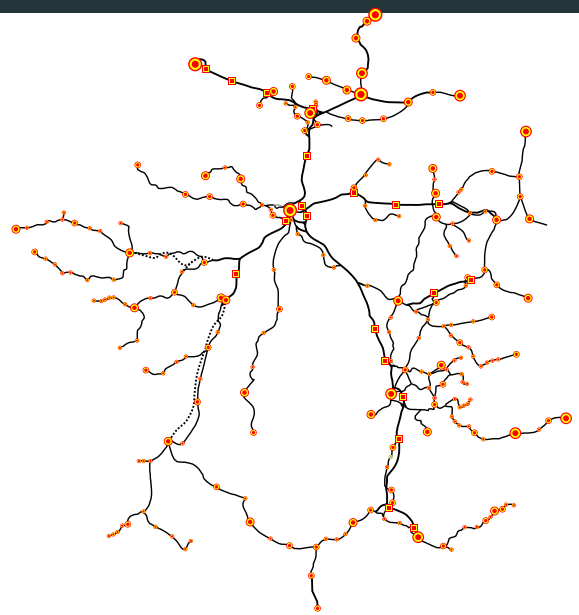


Query Q: a sentence in monadic second-order logic (MSO)

(Metro|RER)*
| (Bus|Tram)*

Probabilistic query evaluation on treelike data

- Database D with treewidth $\leq k$ for some constant k
- Probability of each fact of D to be actually present in the data (independently from other facts)



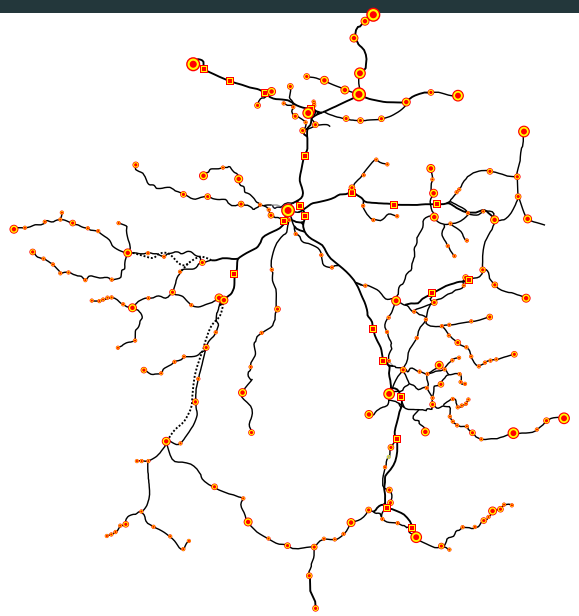
Query Q: a sentence in monadic second-order logic (MSO)

(Metro|RER)*
| (Bus|Tram)*

Result: Probability that the database D satisfies query Q

Probabilistic query evaluation on treelike data

- Database D with treewidth $\leq k$ for some constant k
- Probability of each fact of D to be actually present in the data (independently from other facts)



Query Q: a sentence in monadic second-order logic (MSO)

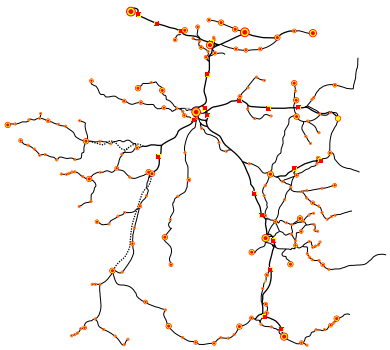
(Metro|RER)*
| (Bus|Tram)*

Result: Probability that the database D satisfies query Q

Computational complexity as a function of the database D
(the query Q is fixed)

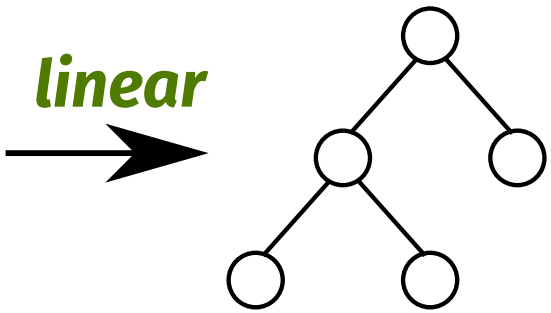
Roadmap

Treelike **data**



Tree **encoding**

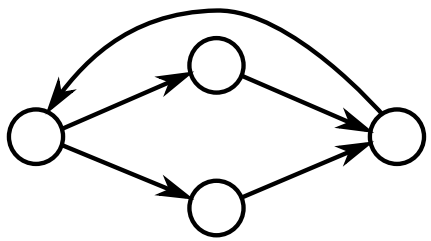
linear



MSO query

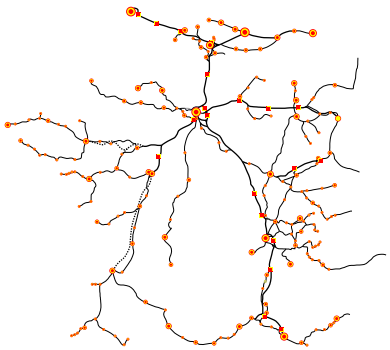
(RER|metro)*
|(bus|tram)*

Tree **automaton**



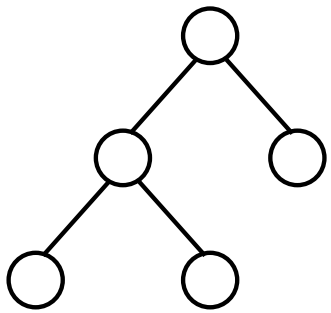
Roadmap

Treelike **data**



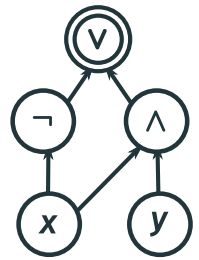
linear

Tree **encoding**



linear

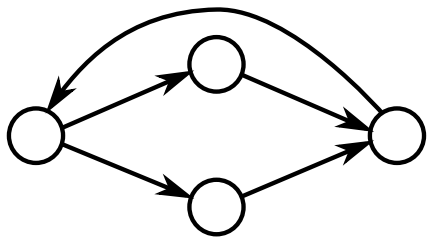
Provenance circuit



MSO query

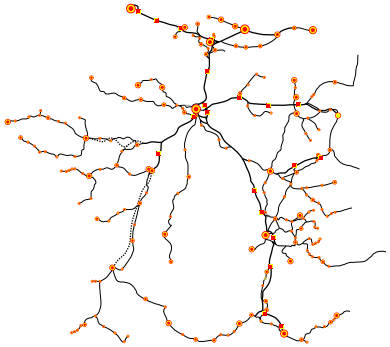
(RER|metro)*
|(bus|tram)*

Tree **automaton**



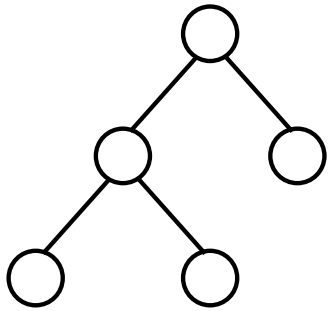
Roadmap

Probabilistic treelike **data**



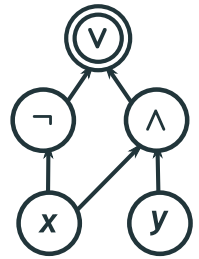
linear

Uncertain tree **encoding**



linear

Provenance circuit +probabilities

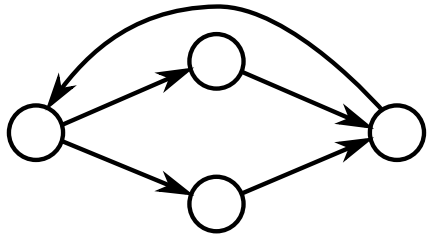


linear

MSO query

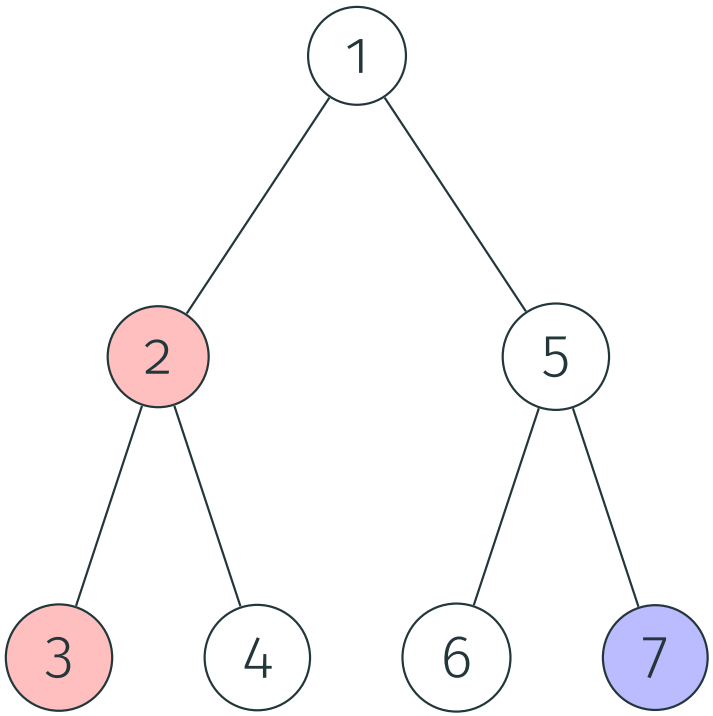
(RER|metro)*
|(bus|tram)*

Tree **automaton**

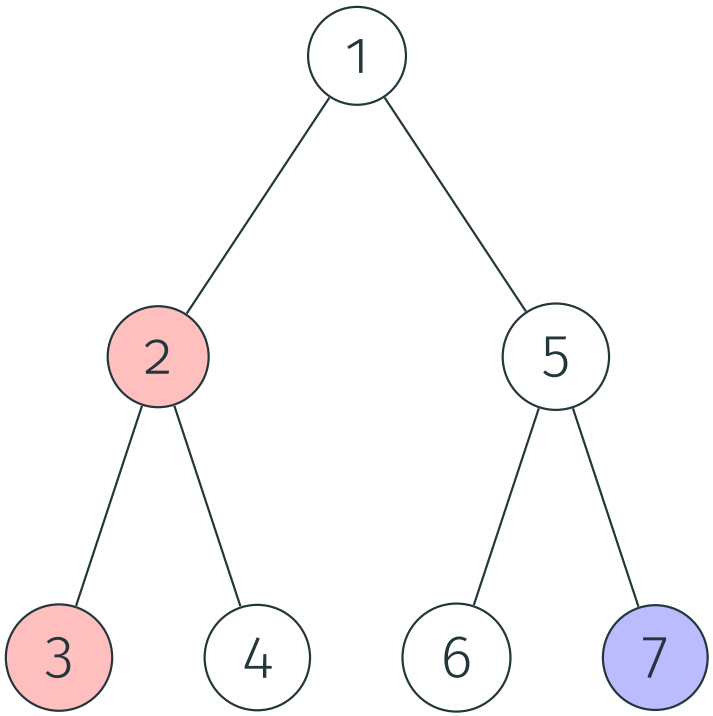


95%
Probability

Uncertain trees

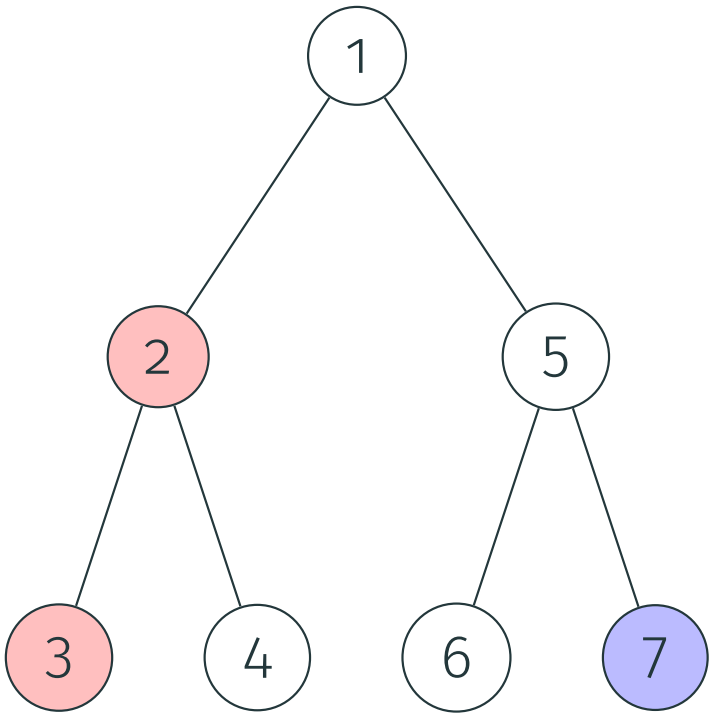


Uncertain trees



A valuation of a tree decides whether to keep (1) or discard (0) node labels

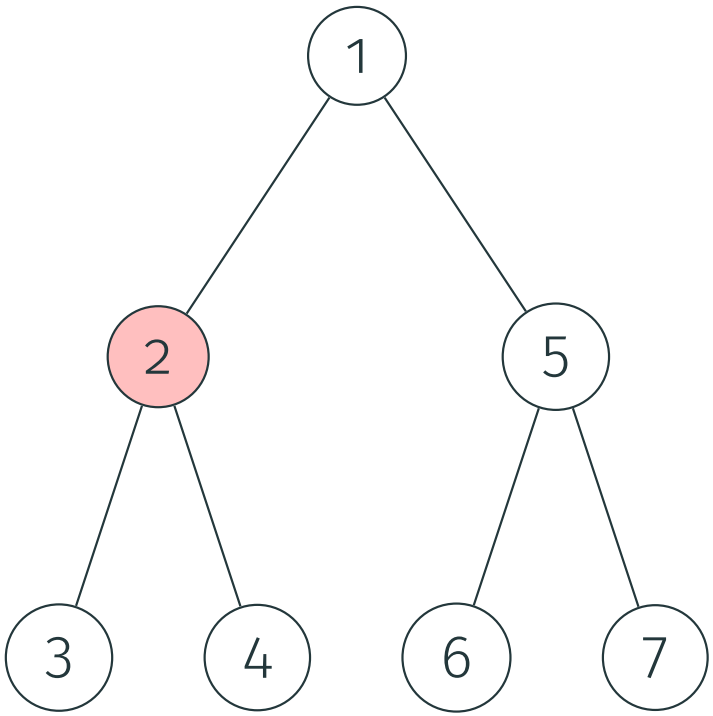
Uncertain trees



A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2, 3, 7 \mapsto 1, \quad * \mapsto 0\}$

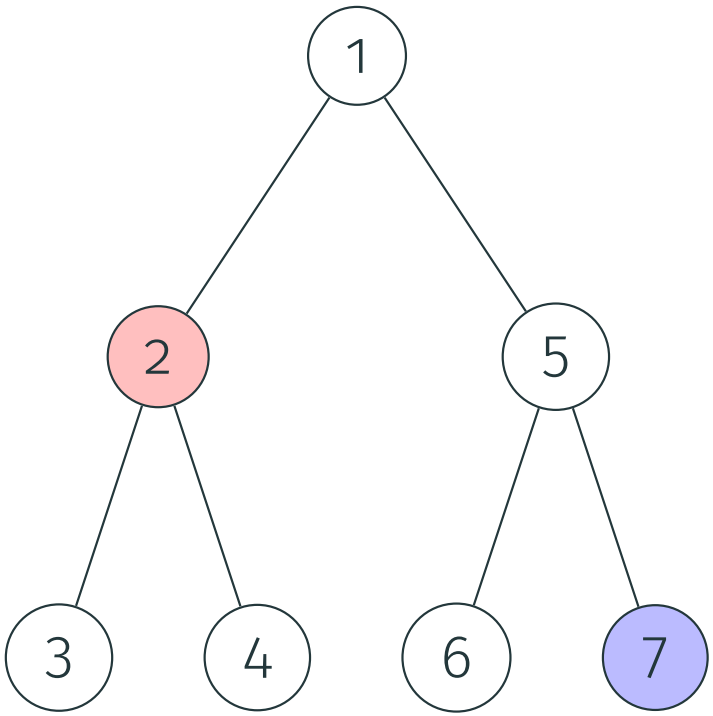
Uncertain trees



A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2 \mapsto 1, \quad * \mapsto 0\}$

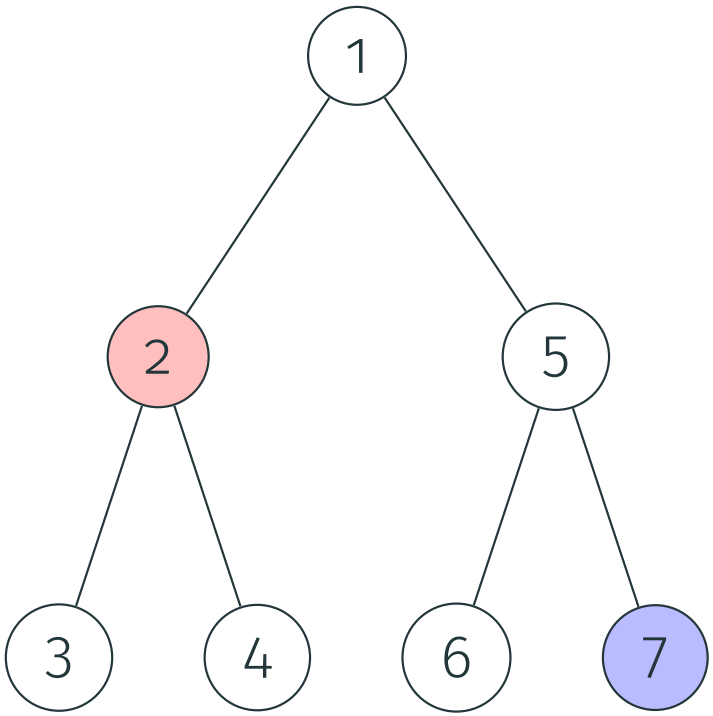
Uncertain trees



A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2, 7 \mapsto 1, \quad * \mapsto 0\}$

Uncertain trees

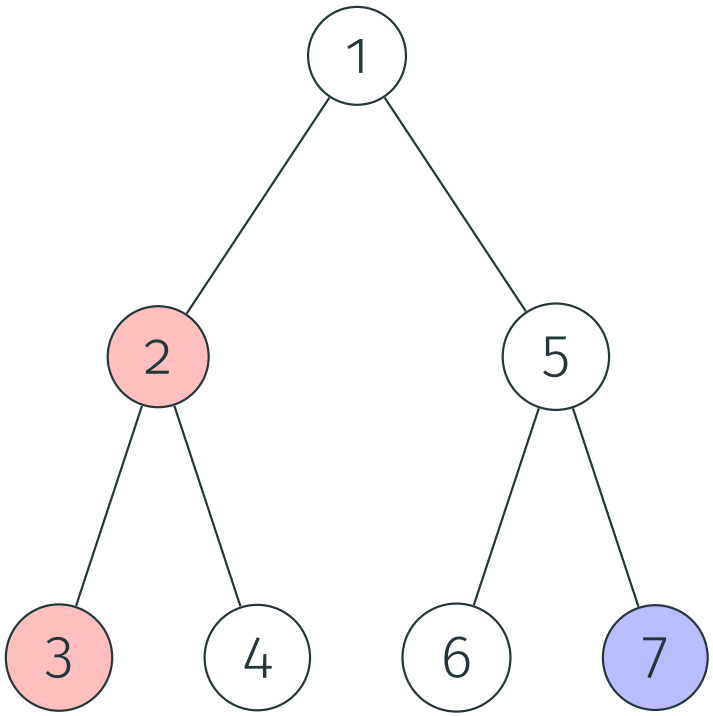


A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2, 7 \mapsto 1, \quad * \mapsto 0\}$

A: *"Is there both a pink and a blue node?"*

Uncertain trees



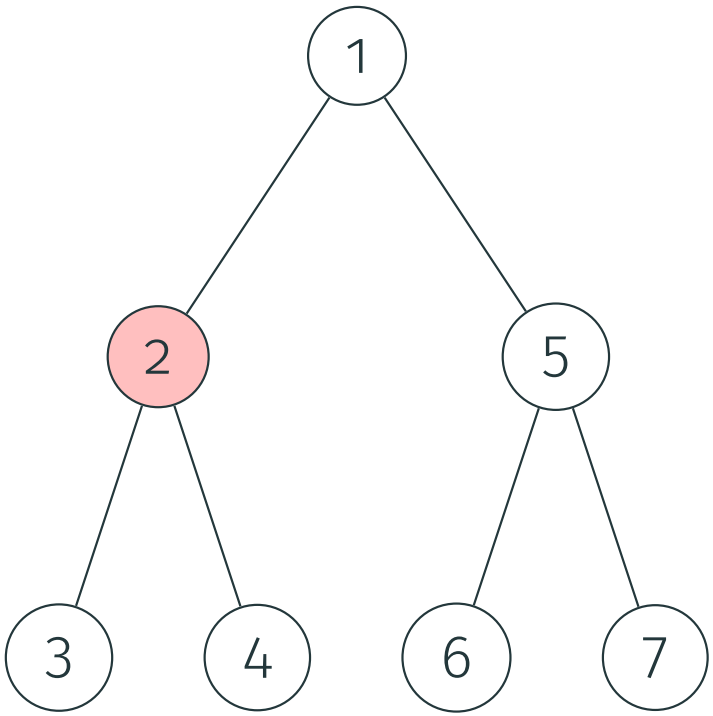
A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2, 3, 7 \mapsto 1, \quad * \mapsto 0\}$

A: *"Is there both a pink and a blue node?"*

The tree automaton A accepts

Uncertain trees



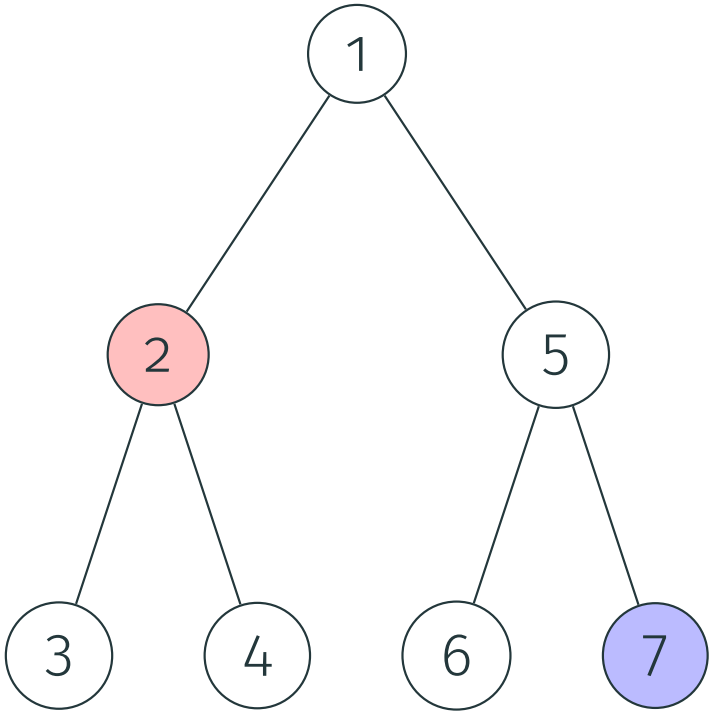
A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2 \mapsto 1, \quad * \mapsto 0\}$

A: *"Is there both a pink and a blue node?"*

The tree automaton A rejects

Uncertain trees



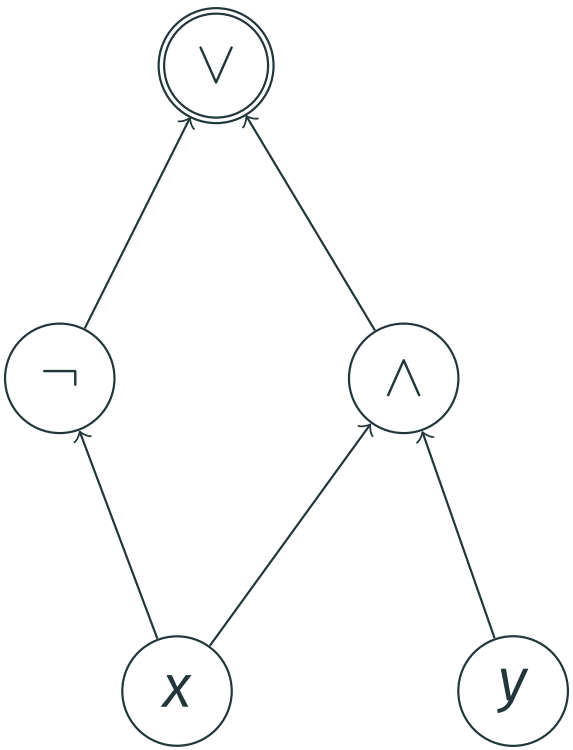
A valuation of a tree decides whether to keep (1) or discard (0) node labels

Valuation: $\{2, 7 \mapsto 1, \quad * \mapsto 0\}$

A: *"Is there both a pink and a blue node?"*

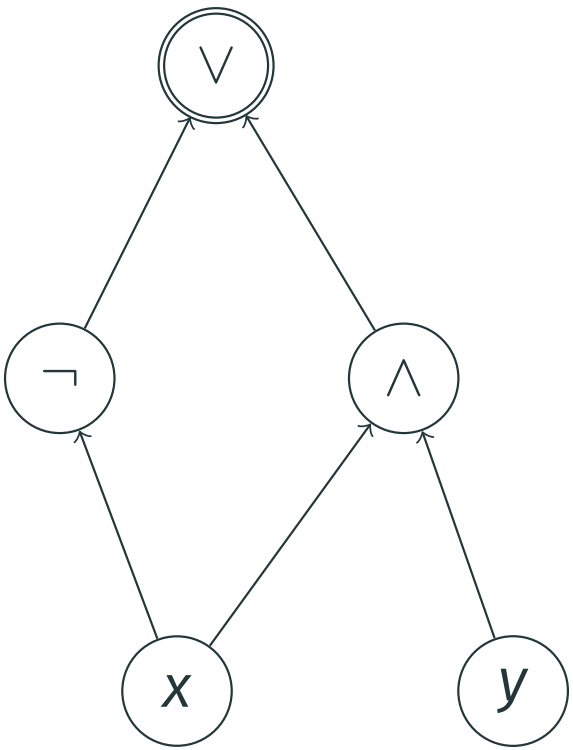
The tree automaton A accepts

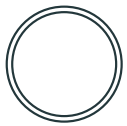
Boolean circuit



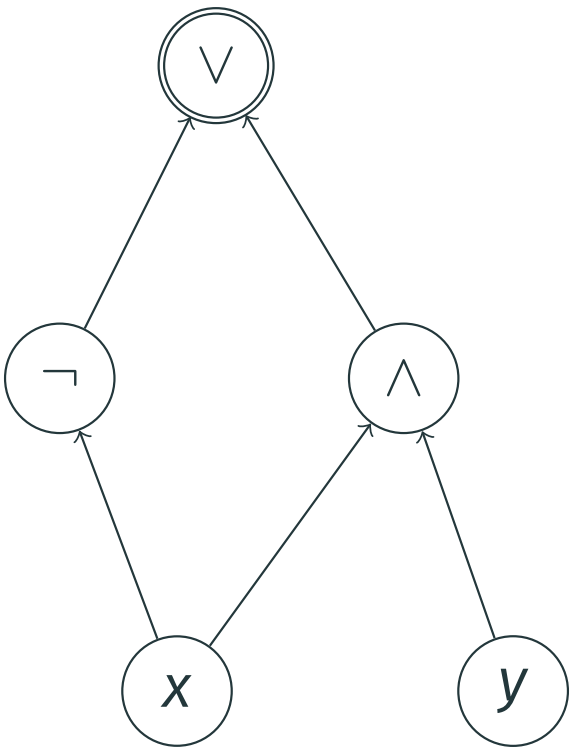
- Directed acyclic graph of gates

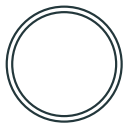
Boolean circuit



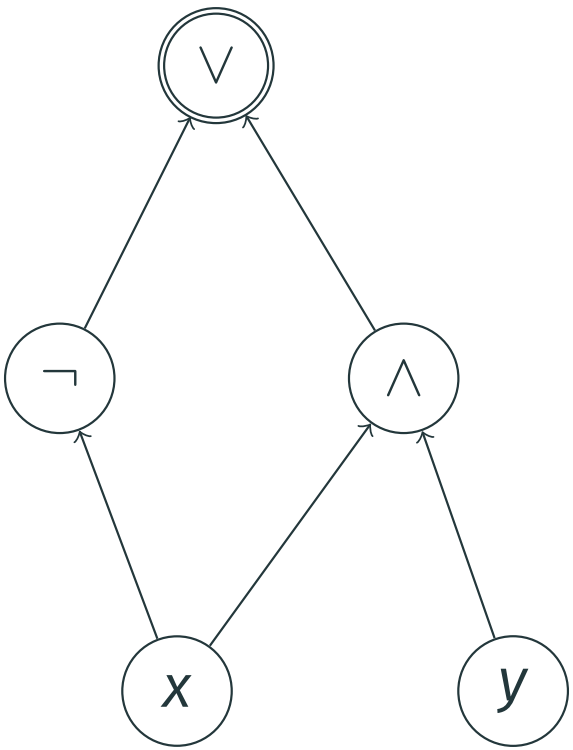
- Directed acyclic graph of gates
- Output gate: 

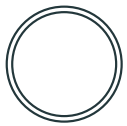
Boolean circuit



- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 

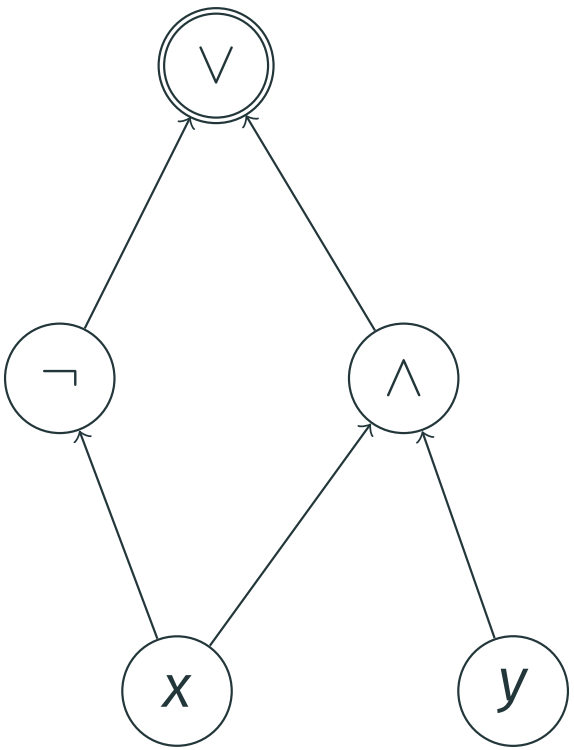
Boolean circuit

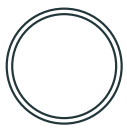


- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 

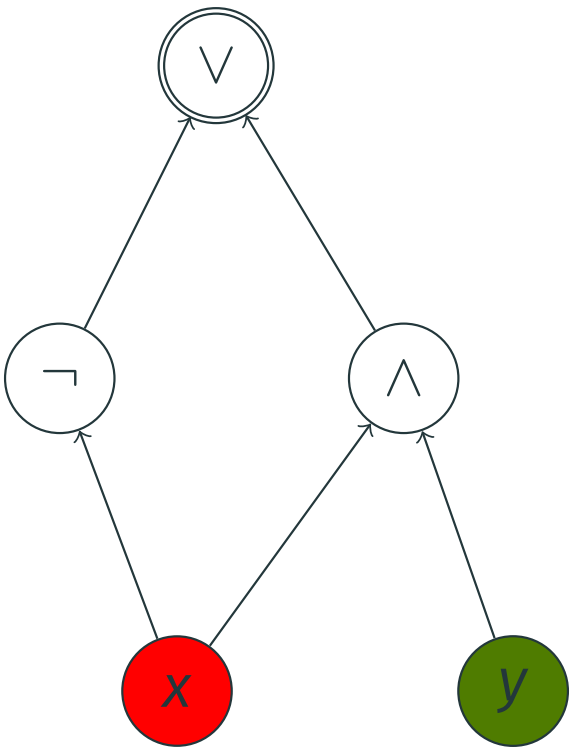
- Internal gates: 

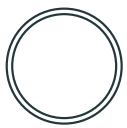
Boolean circuit



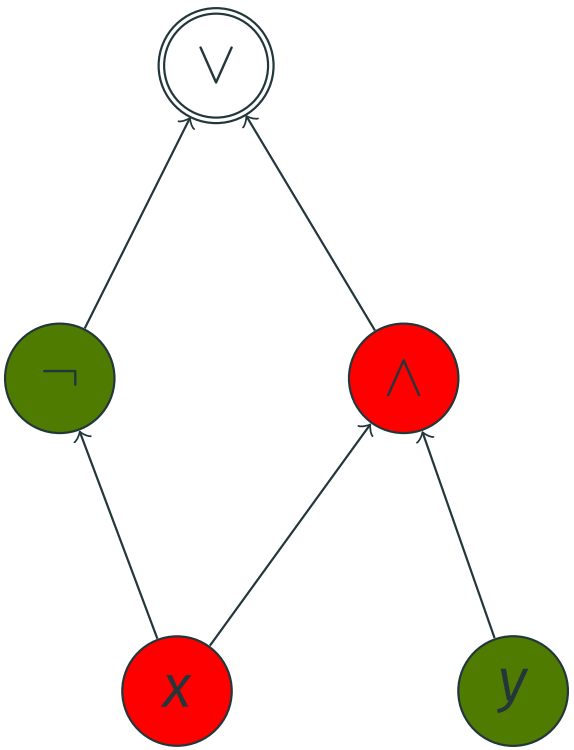
- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 
- Internal gates: 
- Valuation: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

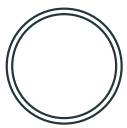
Boolean circuit



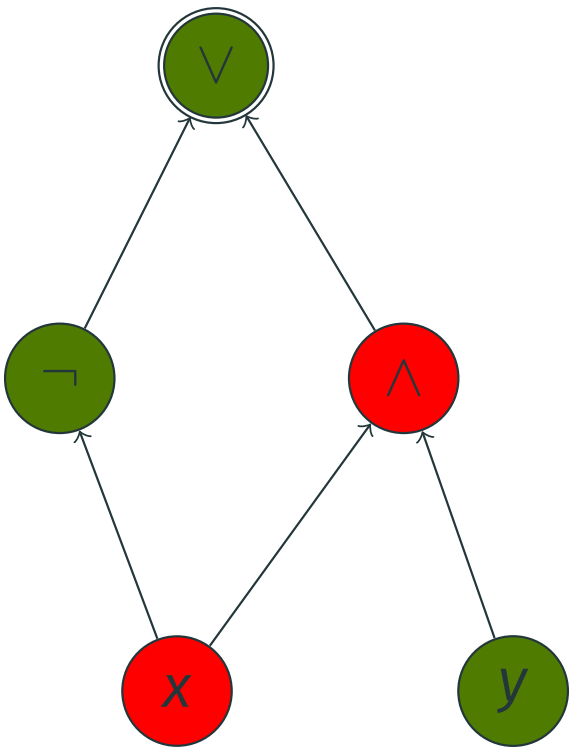
- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 
- Internal gates: 
- Valuation: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

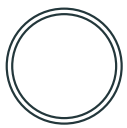
Boolean circuit



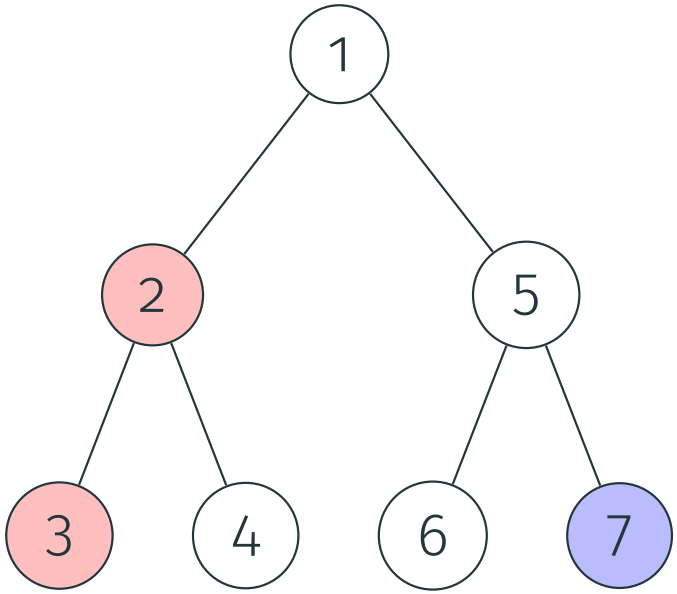
- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 
- Internal gates: 
- Valuation: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

Boolean circuit



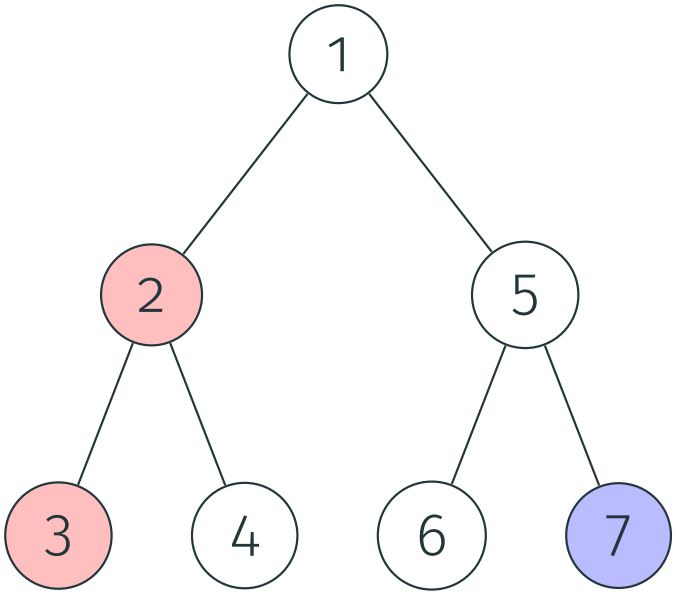
- Directed acyclic graph of gates
- Output gate: 
- Variable gates: 
- Internal gates: 
- Valuation: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$... mapped to 1

Provenance circuit



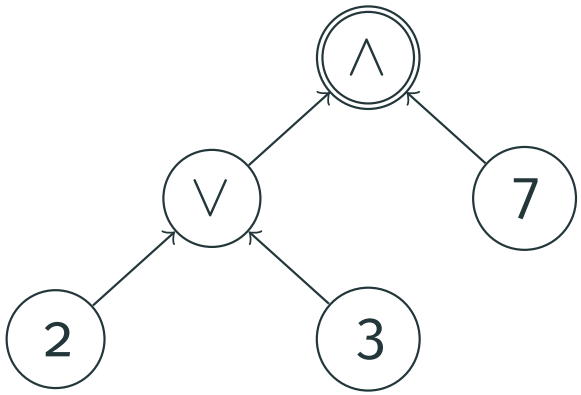
Query: *Is there both a pink and a blue node?*

Provenance circuit

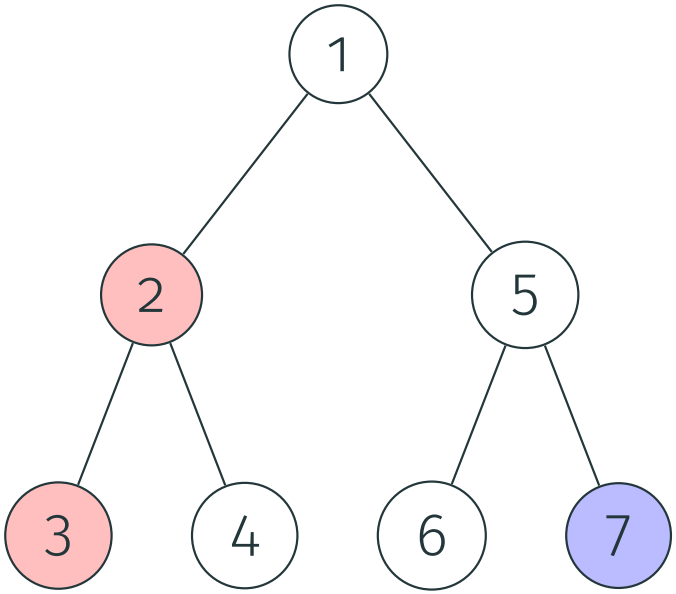


Query: *Is there both a pink and a blue node?*

Provenance circuit:

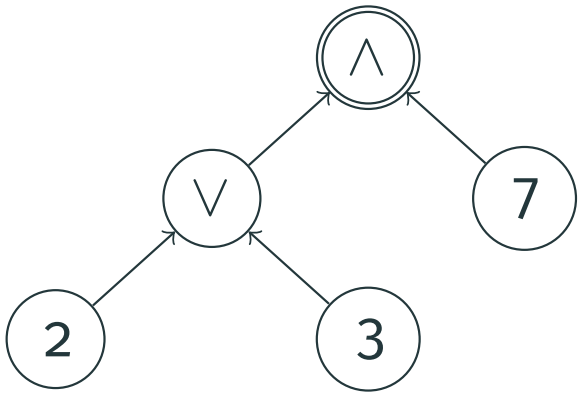


Provenance circuit



Query: *Is there both a pink and a blue node?*

Provenance circuit:



Formally:

- Tree automaton A, uncertain tree T, circuit C
- Variable gates of C: nodes of T
- Condition: Let ν be a valuation of T, then $\nu(C)$ iff A accepts $\nu(T)$

Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T,
we can build a provenance circuit of A on T in $O(|A| \times |T|)$

Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T,
we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: 
- Automaton: *"Is there both a pink and a blue node?"*

- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$

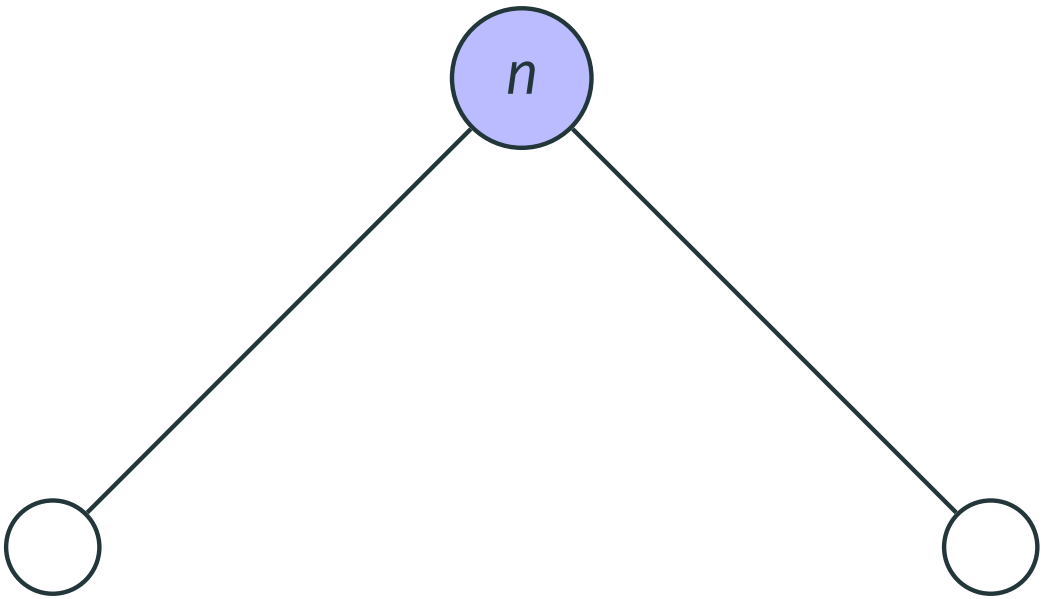
- Transitions:

Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:

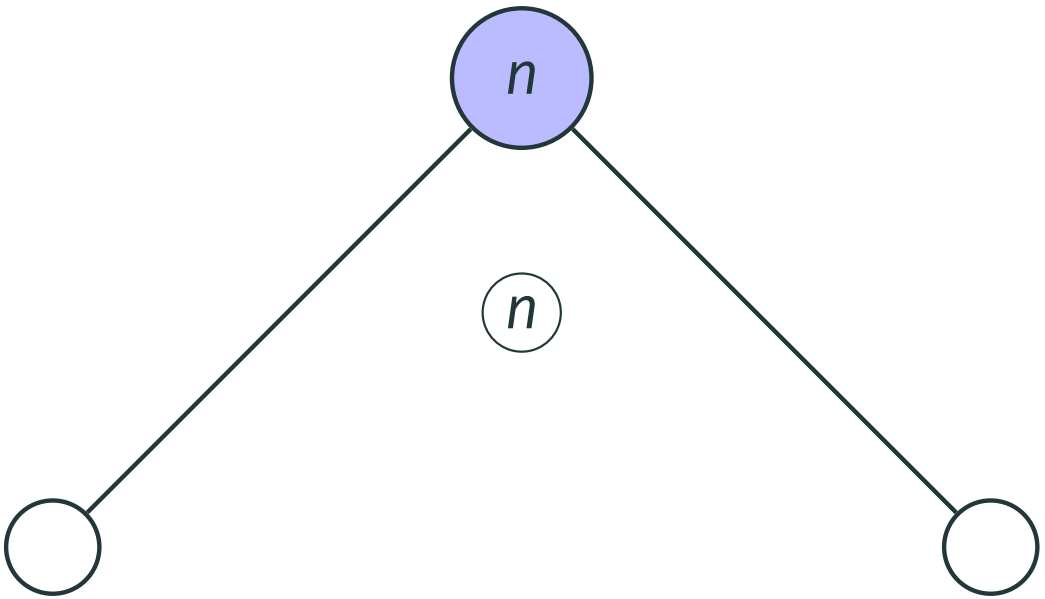


Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:

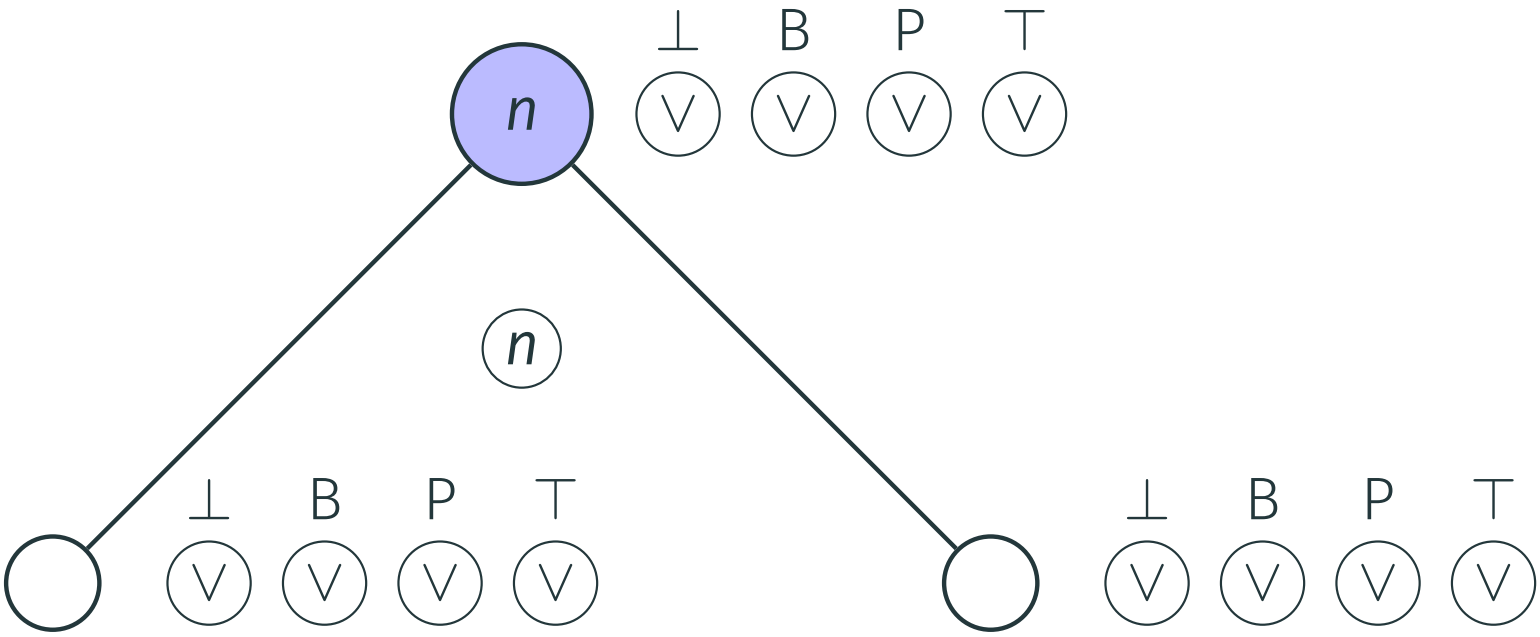


Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:

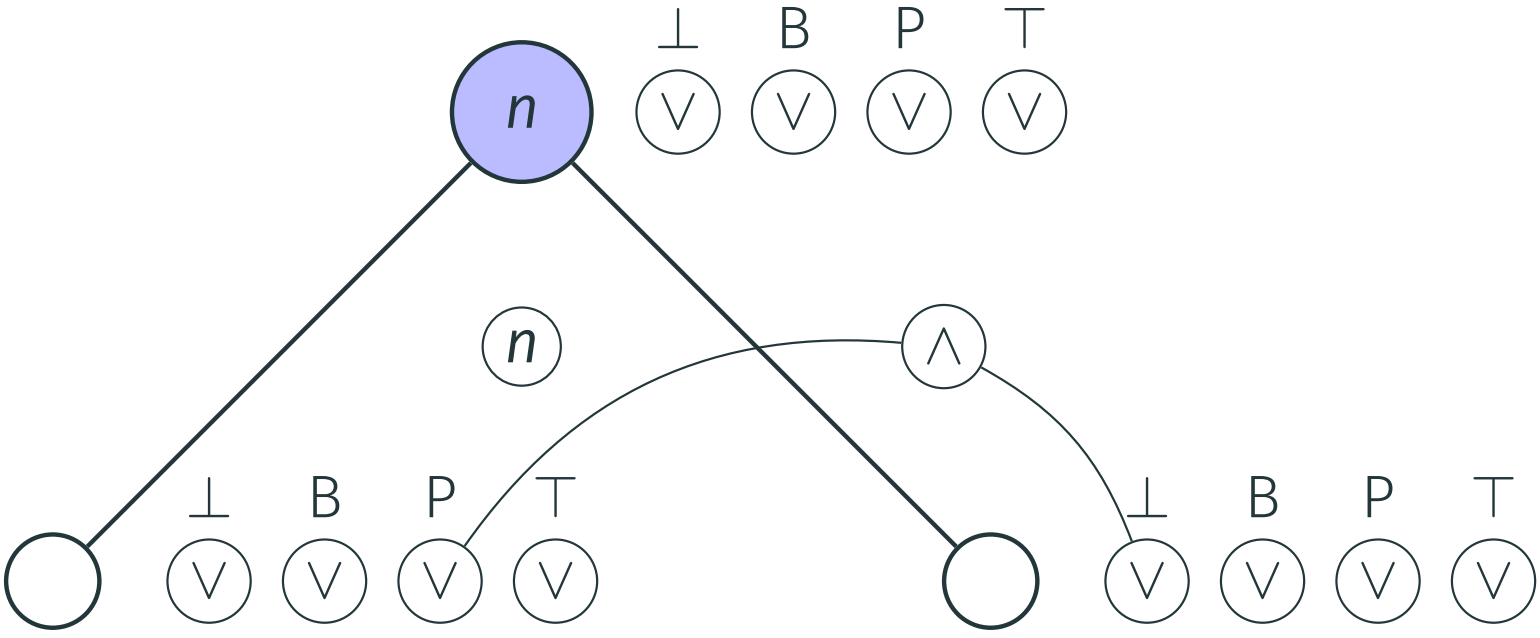


Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:

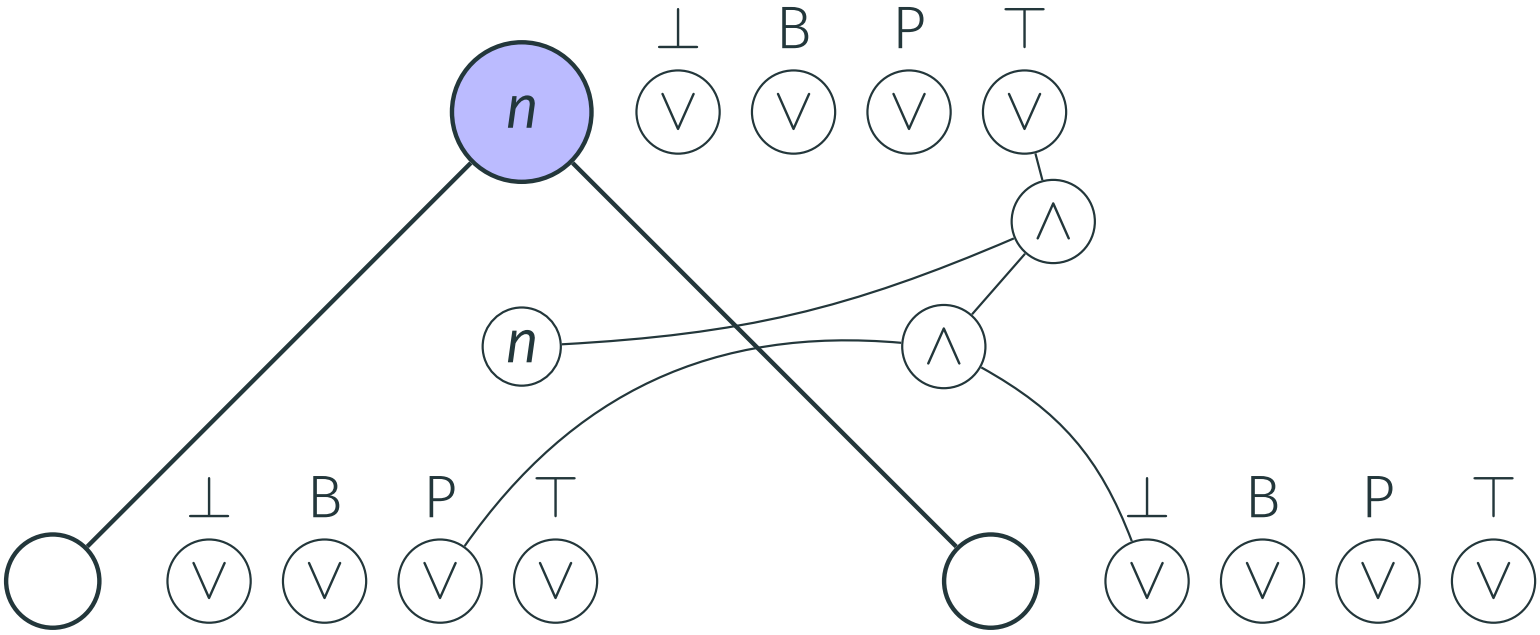


Building provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:

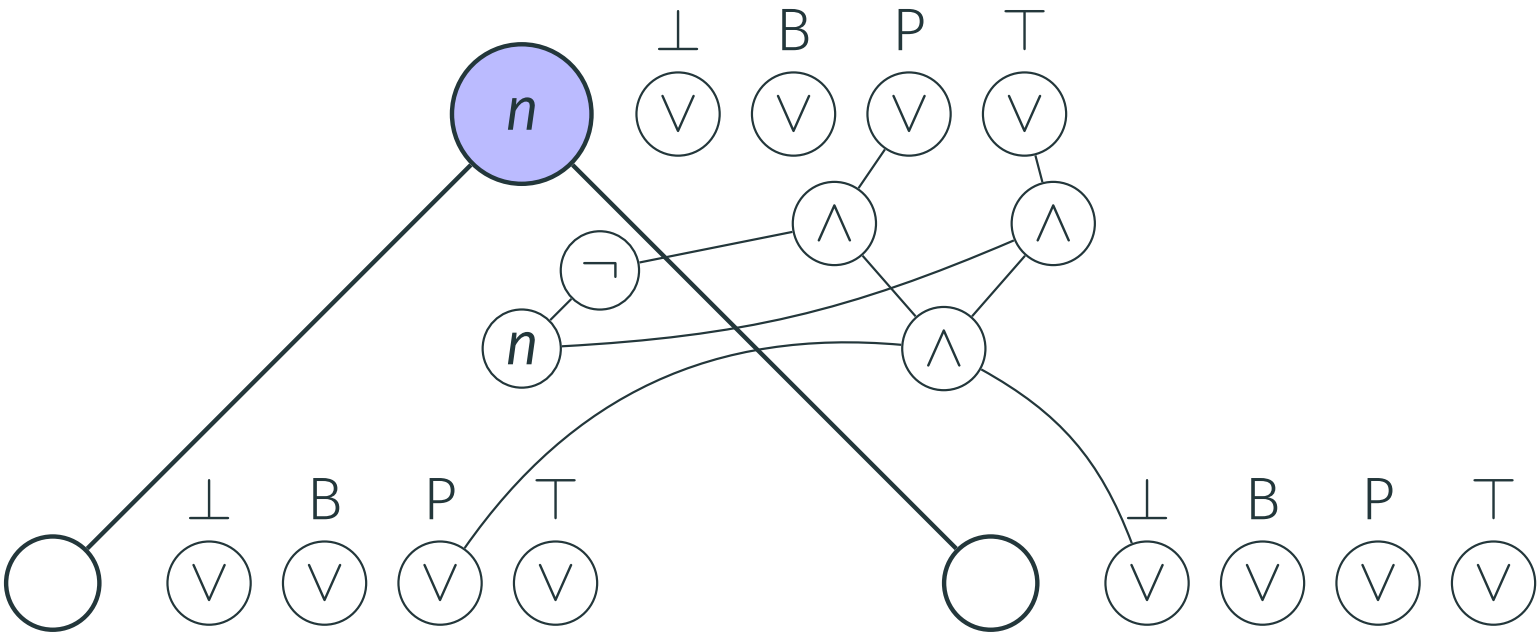


Building provenance circuits on trees

Theorem

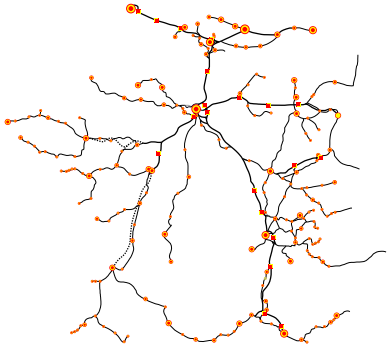
For any bottom-up tree automaton A and input tree T, we can build a provenance circuit of A on T in $O(|A| \times |T|)$

- Alphabet: ○ ● ●
- Automaton: *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final: $\{\top\}$
- Transitions:



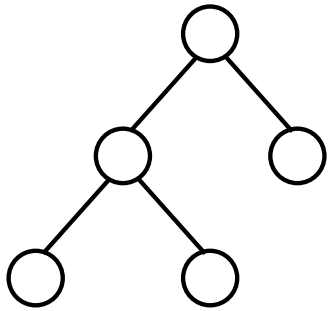
Probabilistic query evaluation

Probabilistic treelike **data**



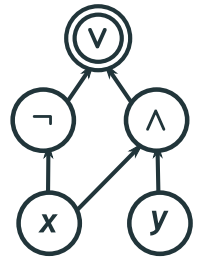
linear

Uncertain tree **encoding**



linear

Provenance circuit +probabilities



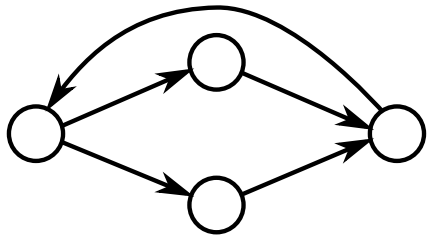
linear

95%
Probability

MSO query

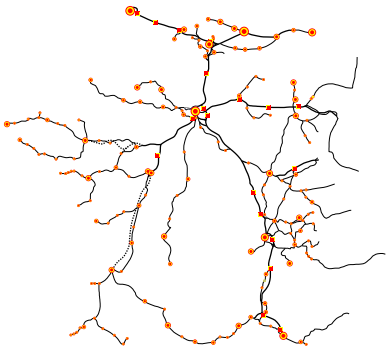
(RER|metro)*
|(bus|tram)*

Tree **automaton**



Details of the approach

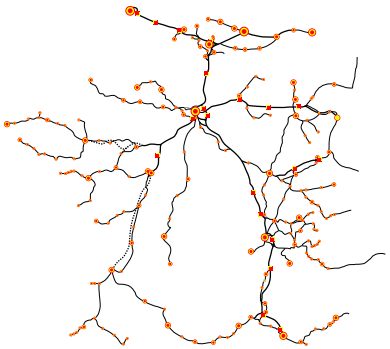
Probabilistic treelike **data**



*Each **fact** can **disappear** with some probability*

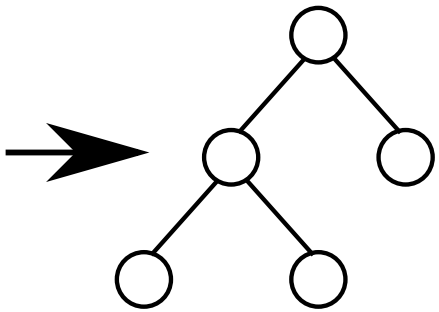
Details of the approach

Probabilistic treelike **data**



*Each **fact** can **disappear** with some probability*

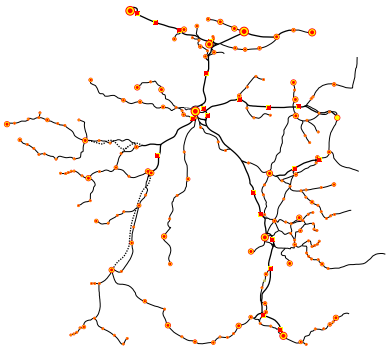
Uncertain tree **encoding**



*Each **node label** can **disappear** with the probability of the coded **fact***

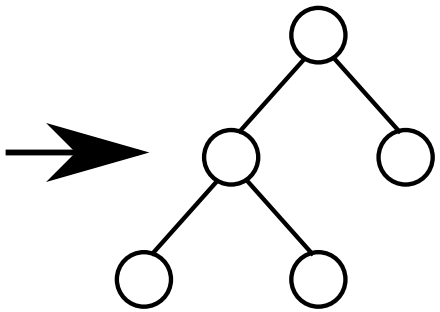
Details of the approach

Probabilistic treelike **data**



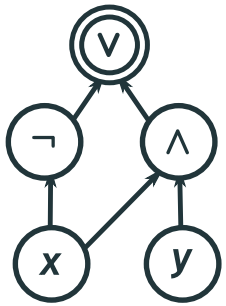
*Each **fact** can **disappear** with some probability*

Uncertain tree **encoding**



*Each **node label** can **disappear** with the probability of the coded **fact***

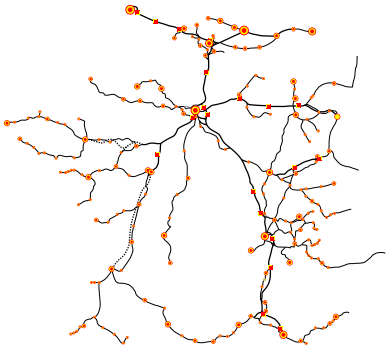
Provenance circuit +probabilities



*Each **variable** can **be true** with the probability of the coded **fact***

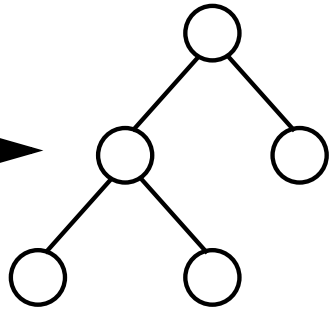
Details of the approach

Probabilistic treelike **data**



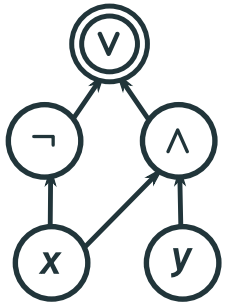
*Each **fact** can **disappear** with some probability*

Uncertain tree **encoding**



*Each **node label** can **disappear** with the probability of the coded **fact***

Provenance circuit +probabilities



*Each **variable** can **be true** with the probability of the coded **fact***

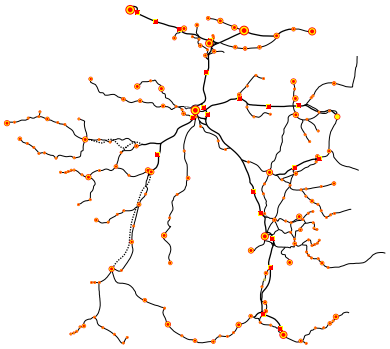
Probability

95%

*Probability that the **circuit** evaluates to **true***

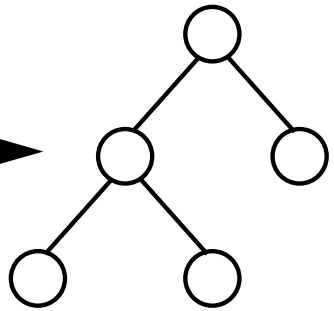
Details of the approach

Probabilistic treelike **data**



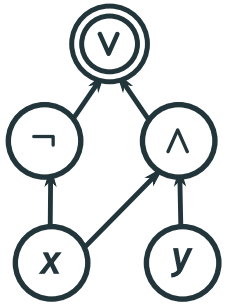
*Each **fact** can **disappear** with some probability*

Uncertain tree **encoding**



*Each **node label** can **disappear** with the probability of the coded **fact***

Provenance circuit +probabilities



*Each **variable** can **be true** with the probability of the coded **fact***

Probability

95%

*Probability that the **circuit** evaluates to **true***

→ How to compute efficiently the probability of the circuit?

Computing the circuit probability

The circuit is a d-DNNF...

Computing the circuit probability

The circuit is a d-DNNF...

-  gates only have variables as inputs

Computing the circuit probability

The circuit is a d-DNNF...

-  gates only have variables as inputs
-  gates always have mutually exclusive inputs

Computing the circuit probability

The circuit is a d-DNNF...

-  gates only have variables as inputs
-  gates always have mutually exclusive inputs
-  gates are all on independent inputs

Computing the circuit probability

The circuit is a d-DNNF...

... so probability computation is easy!

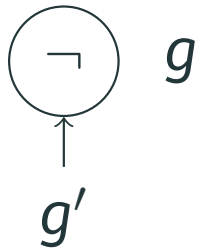
-  gates only have variables as inputs
-  gates always have mutually exclusive inputs
-  gates are all on independent inputs

Computing the circuit probability

The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

... so probability computation is easy!

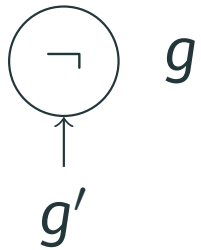


Computing the circuit probability

The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

... so probability computation is easy!



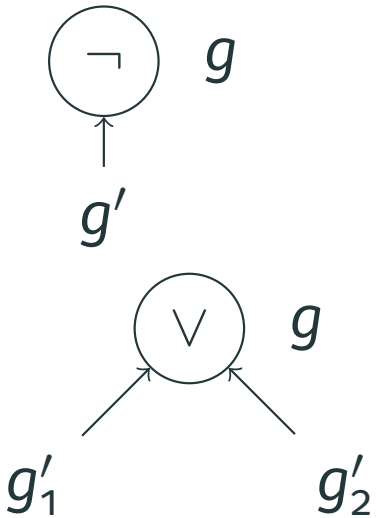
$$P(g) := 1 - P(g')$$

Computing the circuit probability

The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

... so probability computation is easy!



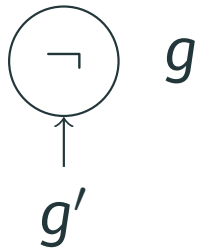
$$P(g) := 1 - P(g')$$

Computing the circuit probability

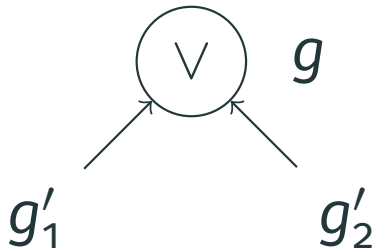
The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

... so probability computation is easy!



$$P(g) := 1 - P(g')$$



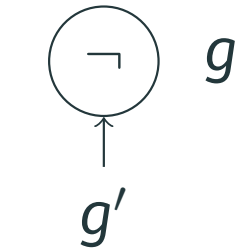
$$P(g) := P(g'_1) + P(g'_2)$$

Computing the circuit probability

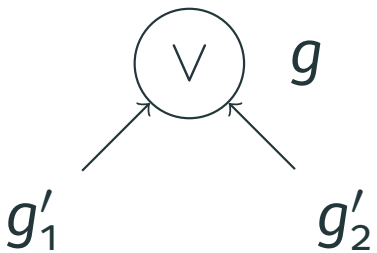
The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

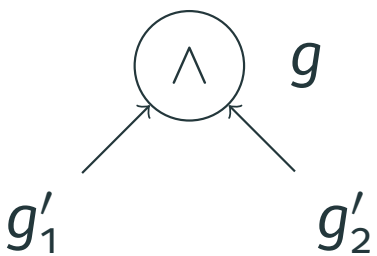
... so probability computation is easy!



$$P(g) := 1 - P(g')$$



$$P(g) := P(g'_1) + P(g'_2)$$

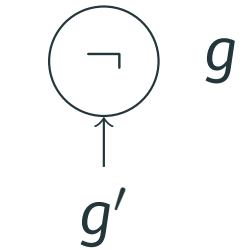


Computing the circuit probability

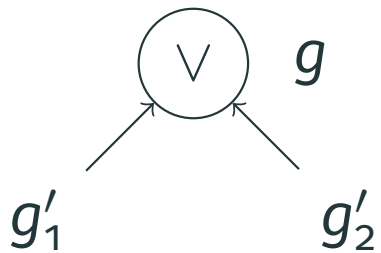
The circuit is a d-DNNF...

- $\neg$ gates only have variables as inputs
- $\vee$ gates always have mutually exclusive inputs
- $\wedge$ gates are all on independent inputs

... so probability computation is easy!



$$P(g) := 1 - P(g')$$



$$P(g) := P(g'_1) + P(g'_2)$$

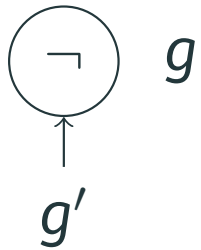
$$P(g) := P(g'_1) \times P(g'_2)$$

Computing the circuit probability

The circuit is a d-DNNF...

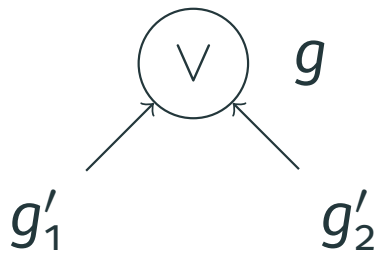
... so probability computation is easy!

- $\neg$ gates only have variables as inputs



$$P(g) := 1 - P(g')$$

- $\vee$ gates always have mutually exclusive inputs



$$P(g) := P(g'_1) + P(g'_2)$$

- $\wedge$ gates are all on independent inputs

$$P(g) := P(g'_1) \times P(g'_2)$$

Theorem [Amarilli et al., 2015]

For any fixed Boolean MSO query Q and $k \in \mathbb{N}$, given a database D of treewidth $\leq k$ with independent probabilities, we can compute in linear time the probability that D satisfies Q

Table of contents

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

$\rightarrow$ Add special facts to materialize all possible assignments (linear)

- e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

$\rightarrow$ Add special facts to materialize all possible assignments (linear)

- e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

$\rightarrow$ Rewrite Q to a Boolean query that reads the special facts

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

$\rightarrow$ Add special facts to materialize all possible assignments (linear)
  - e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

$\rightarrow$ Rewrite Q to a Boolean query that reads the special facts

$\rightarrow$ Provenance circuit: factorized representation of the query result

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

$\rightarrow$ Add special facts to materialize all possible assignments (linear)
  - e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

$\rightarrow$ Rewrite Q to a Boolean query that reads the special facts

$\rightarrow$ Provenance circuit: factorized representation of the query result

X	Y
a	b
a	b'

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

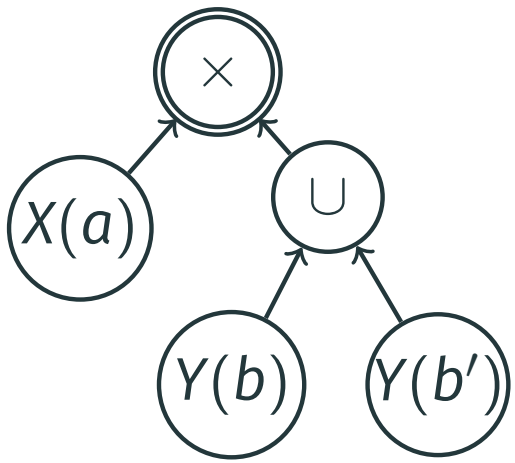
→ Add special facts to materialize all possible assignments (linear)
  - e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

→ Rewrite Q to a Boolean query that reads the special facts

→ Provenance circuit: factorized representation of the query result

X	Y
a	b
a	b'

represented by



Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

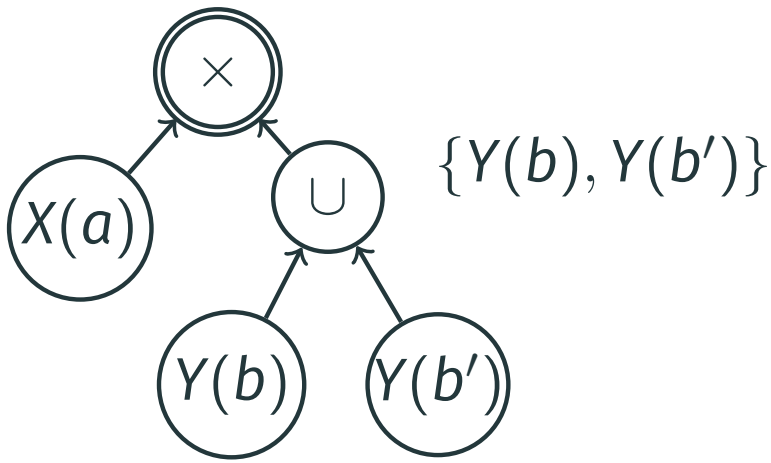
$\rightarrow$ Add special facts to materialize all possible assignments (linear)
  - e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

$\rightarrow$ Rewrite Q to a Boolean query that reads the special facts

$\rightarrow$ Provenance circuit: factorized representation of the query result

X	Y
a	b
a	b'

represented by



$\{Y(b), Y(b')\}$

Circuits as factorized representations of query results

- Query $Q(\vec{X})$ with free variables
- Query result: all tuples $\vec{a}$ such that D satisfies $Q(\vec{a})$

This task can also be solved efficiently! Make the query Boolean again

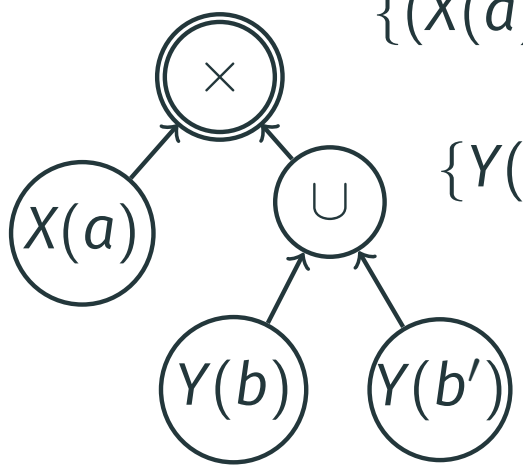
→ Add special facts to materialize all possible assignments (linear)
  - e.g., $X_i(a_j)$: means element a_i is mapped to variable X_j

→ Rewrite Q to a Boolean query that reads the special facts

→ Provenance circuit: factorized representation of the query result

X	Y
a	b
a	b'

represented by



$\{(X(a), Y(b)), (X(a), Y(b'))\}$

$\{Y(b), Y(b')\}$

Application of factorized representations

Factorized representation computable in linear time in the data

Application of factorized representations

Factorized representation computable in linear time in the data

- Application: Counting query results [Arnborg et al., 1991]

Application of factorized representations

Factorized representation computable in linear time in the data

- Application: Counting query results [Arnborg et al., 1991]
  - Exclusive $\vee$ means $+$, independent $\wedge$ means $\times$
  - Reproves existing result: [Arnborg et al., 1991]

Application of factorized representations

Factorized representation computable in linear time in the data

- Application: Counting query results [Arnborg et al., 1991]
  - Exclusive $\vee$ means $+$, independent $\wedge$ means $\times$
  - Reproves existing result: [Arnborg et al., 1991]
- Application: Constant-delay enumeration of query results

Application of factorized representations

Factorized representation computable in linear time in the data

- Application: Counting query results [Arnborg et al., 1991]
  - Exclusive $\vee$ means $+$, independent $\wedge$ means $\times$
  - Reproves existing result: [Arnborg et al., 1991]
- Application: Constant-delay enumeration of query results
  - Requires some linear-time preprocessing of the input circuit
  - Exclusive $\vee$ means disjoint $\cup$, independent $\wedge$ means relational $\times$
  - New modular proof of existing enumeration result
    [Bagan, 2006, Kazana and Segoufin, 2013, Amarilli et al., 2017a]

Application of factorized representations

Factorized representation computable in linear time in the data

- Application: Counting query results [Arnborg et al., 1991]
  - Exclusive $\vee$ means $+$, independent $\wedge$ means $\times$
  - Reproves existing result: [Arnborg et al., 1991]
- Application: Constant-delay enumeration of query results
  - Requires some linear-time preprocessing of the input circuit
  - Exclusive $\vee$ means disjoint $\cup$, independent $\wedge$ means relational $\times$
  - New modular proof of existing enumeration result [Bagan, 2006, Kazana and Segoufin, 2013, Amarilli et al., 2017a]
- Application: Semiring provenance [Green et al., 2007]

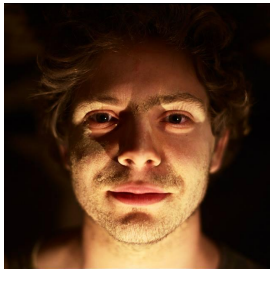
Conclusion and perspectives

- Other results:
  - Lower bounds: probabilistic query evaluation is hard unless treewidth is bounded (modulo assumptions) [Amarilli et al., 2016]
  - Complexity in the query: generally nonelementary but can be improved [Amarilli et al., 2017b, Amarilli et al., 2017c]

Conclusion and perspectives

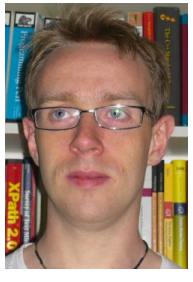
- Other results:
  - Lower bounds: probabilistic query evaluation is hard unless treewidth is bounded (modulo assumptions) [Amarilli et al., 2016]
  - Complexity in the query: generally nonelementary but can be improved [Amarilli et al., 2017b, Amarilli et al., 2017c]
- Ongoing work (with my wonderful co-authors):
  - Efficient enumeration algorithms under updates
  - More lower bounds results, connections to knowledge compilation
  - More expressive provenance: cycluits (circuits with cycles)
  - Combined tractability for probabilistic query evaluation
  - Connections to message passing probabilistic algorithms


Pierre


Louis


Stefan

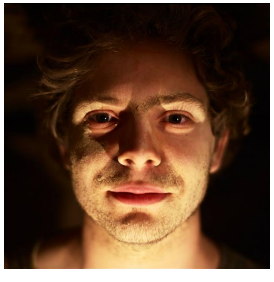

Mikaël


Pierre

Conclusion and perspectives

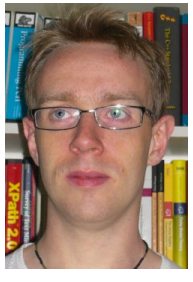
- Other results:
  - Lower bounds: probabilistic query evaluation is hard unless treewidth is bounded (modulo assumptions) [Amarilli et al., 2016]
  - Complexity in the query: generally nonelementary but can be improved [Amarilli et al., 2017b, Amarilli et al., 2017c]
- Ongoing work (with my wonderful co-authors):
  - Efficient enumeration algorithms under updates
  - More lower bounds results, connections to knowledge compilation
  - More expressive provenance: cycluits (circuits with cycles)
  - Combined tractability for probabilistic query evaluation
  - Connections to message passing probabilistic algorithms


Pierre


Louis


Stefan


Mikaël


Pierre

Thanks for your attention!

References i

Amarilli, A., Bourhis, P., Jachiet, L., and Mengel, S. (2017a).
A Circuit-Based Approach to Efficient Enumeration.
In *ICALP*.

Amarilli, A., Bourhis, P., Monet, M., and Senellart, P. (2017b).
Combined Tractability of Query Evaluation via Tree Automata and Cycluits.
In *ICDT*.

Amarilli, A., Bourhis, P., and Senellart, P. (2015).
Provenance Circuits for Trees and Treelike Instances.
In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2016).
Tractable Lineages on Treelike Instances: Limits and Extensions.
In *PODS*.

References ii


Amarilli, A., Monet, M., and Senellart, P. (2017c).
Conjunctive Queries on Probabilistic Graphs: Combined Complexity.
In *PODS*.


Arnborg, S., Lagergren, J., and Seese, D. (1991).
Easy problems for tree-decomposable graphs.
J. Algorithms, 12(2):308–340.


Bagan, G. (2006).
MSO queries on tree decomposable structures are computable with linear delay.
In *CSL*.

References iii


Courcelle, B. (1990).
The monadic second-order logic of graphs. I. Recognizable sets of finite graphs.
Inf. Comput., 85(1).


Green, T. J., Karvounarakis, G., and Tannen, V. (2007).
Provenance semirings.
In *PODS*.


Kazana, W. and Segoufin, L. (2013).
Enumeration of monadic second-order queries on trees.
TOCL, 14(4).

References iv

Thatcher, J. W. and Wright, J. B. (1968).
Generalized finite automata theory with an application to a decision problem of second-order logic.
Mathematical systems theory, 2(1):57–81.

Image credits

- Slides 2 and 5–6:
  - Subway map: `https://commons.wikimedia.org/wiki/File:Paris_Metro_map.svg` (edited), by user Umx on Wikimedia Commons, public domain
  - Ticket t+: `http://www.parisvoyage.com/images/cartoon18.jpg`, ParisVoyage, fair use
  - Terms and conditions: `http://www.vianavigo.com/fileadmin/galerie/pdf/CGU_t_.pdf` (cropped), RATP, fair use
- Slides 3–4: screenshots from `http://lab.vianavigo.com`, Stif, fair use
- Slide 4: newpaper articles (fair use) :
  - `http://www.leparisien.fr/transports/circulation-alternee-a-paris-et-en-banlieue-une-panne-de-rer-et-des-bouchons-06-12-2016-6419610.php`
  - `http://www.rtl.fr/actu/societe-faits-divers/paris-le-trafic-totalement-interrompu-gare-du-nord-7786171150`
  - `https://www.rerb-leblog.fr/incident-rer-b-sest-passe-matin/`
  - `http://www.huffingtonpost.fr/2016/12/06/le-rer-b-en-panne-les-voyageurs-nont-pas-eu-dautres-choix-que/`
  - `http://www.lexpress.fr/actualite/societe/trafic/rer-b-en-panne-retards-du-d-circulation-alternee-deuxieme-journee-de-galere_1857905.html`
  - `http://www.lemonde.fr/entreprises/article/2016/12/07/ile-de-france-le-trafic-toujours-interrompu-sur-le-rer-b-en-direction-de-roissy_5044717_1656994.html`
- Slides 6, 13, 16, 21–22: Train map `https://commons.wikimedia.org/wiki/File:Carte_TGV.svg?uselang=fr` (edited), by users Jack ma, Muselaar, Benjism89, Pic-Sou, Uwe Dedering, Madcap on Wikimedia Commons, license CC-BY-SA 3.0
- Slide 27: Photos `http://www.lifl.fr/~bourhis/pb.png`, `http://tyrex.inria.fr/people/img/jachiet.png`, `http://www.cril.univ-artois.fr/~mengel/snap.jpeg`, `http://mikael-monet.net/images/moi.jpg`, `http://pierre.senellart.com/bubu.jpg`, fair use