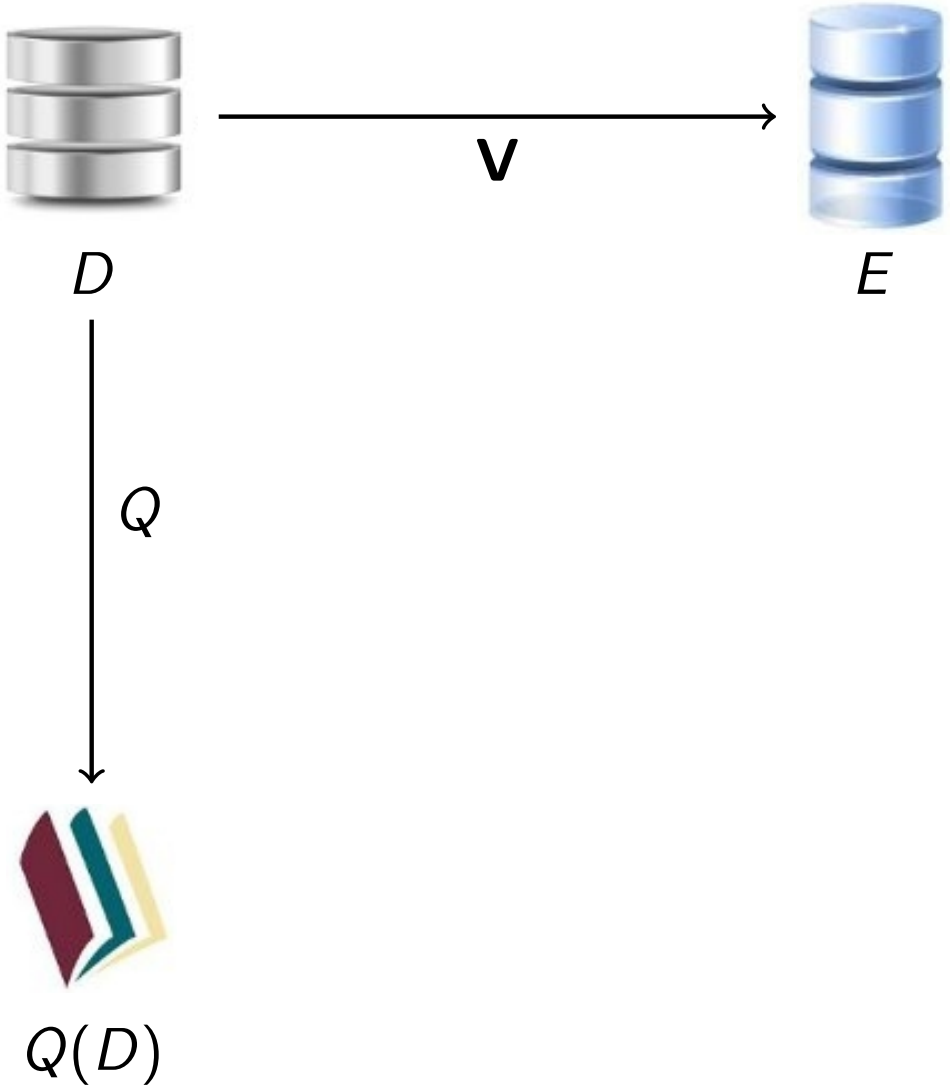
Datalog Rewritings of Regular Path Queries using Views

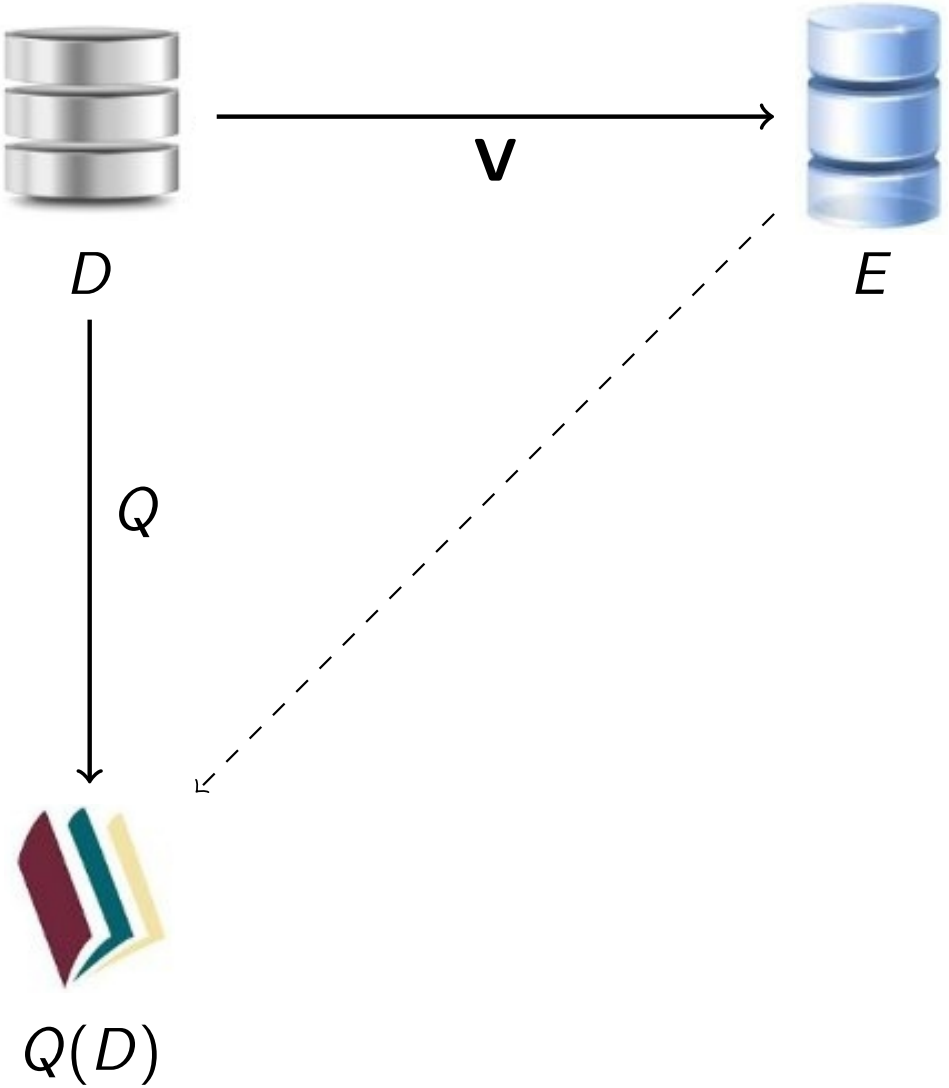
Nadime Francis
with Luc Segoufin and Cristina Sirangelo

ENS-Cachan, Inria

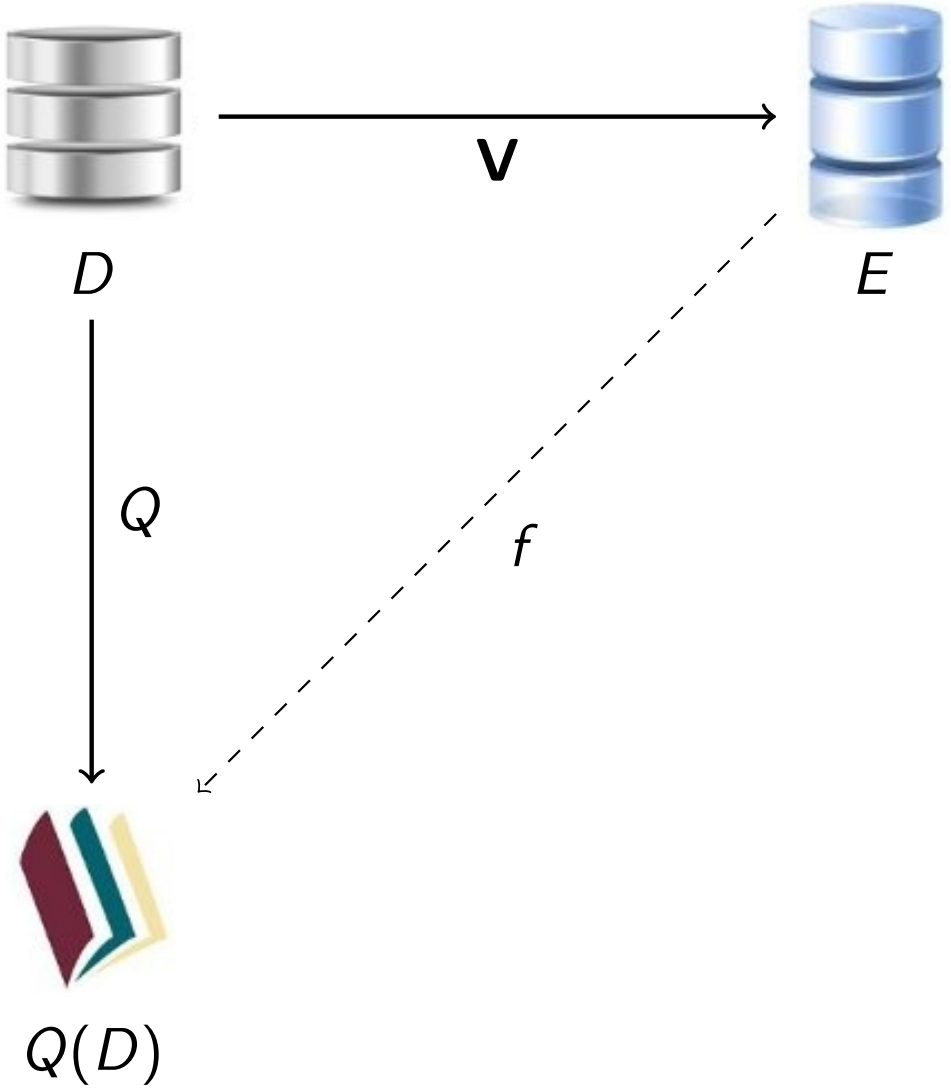
Tuesday, March, 25th
ICDT 2014 - Athens

Determinacy and Query Rewriting

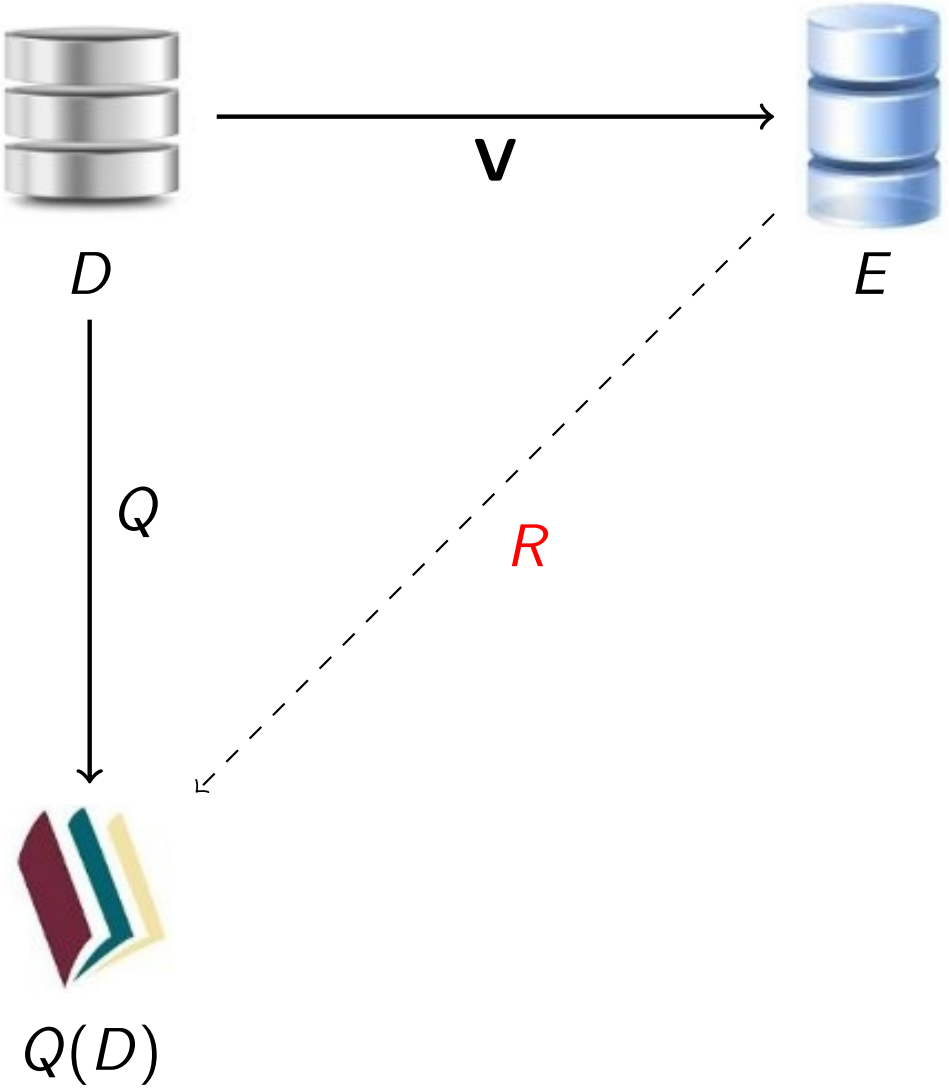

V
D
E
Q
Q(D)



1. Can I get Q(D) from E ?



1. Can I get Q(D) from E ?
2. Does it work for all D ?



1. Can I get Q(D) from E ?
2. Does it work for all D ?
3. If so, can I compute R ? (in what language ?)

Setting

- Graph databases : D
- Regular Path Queries : Q
- Regular Path Views : $\mathbf{V} = \{V_1, \ldots, V_n\}$
- View extensions : E

Determinacy

- $\forall D, D' \qquad \mathbf{V}(D) = \mathbf{V}(D') \Rightarrow Q(D) = Q(D')$
- $Q, \mathbf{V} \rightsquigarrow f$: function induced by Q using $\mathbf{V}$

Rewriting

- Query R such that
  $\forall D \qquad R(\mathbf{V}(D)) = f(\mathbf{V}(D))$

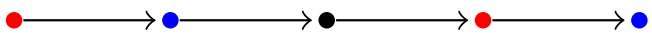
Example 1

$$V_3 = a^3 \qquad Q = a^4$$

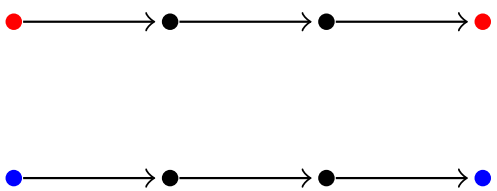
Example 1

$$V_3 = a^3 \qquad Q = a^4$$

The view does not determine the query.



D



D'

Example 2 (Afrati)

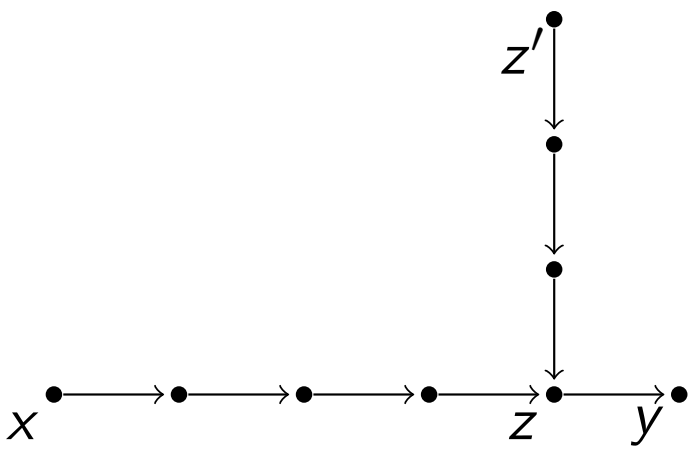
$$V_3 = a^3 \qquad V_4 = a^4 \qquad Q = a^5$$

Example 2 (Afrati)

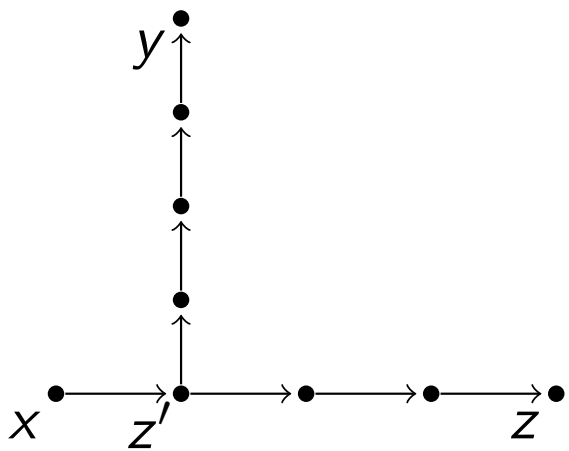
$$V_3 = a^3 \quad V_4 = a^4 \quad Q = a^5$$

$$\mathrm{R}(x, y) = \exists z,\, V_4(x, z) \wedge \forall z'\, V_3(z', z) \Rightarrow V_4(z', y)$$

$\mathrm{Q} \Rightarrow \mathrm{R}$



$\mathrm{R} \Rightarrow \mathrm{Q}$



Example 2 (Afrati)

$$V_3 = a^3 \quad V_4 = a^4 \quad Q = a^5$$

$$\boxed{R(x,y) = \exists z, V_4(x,z) \wedge \forall z' V_3(z',z) \Rightarrow V_4(z',y)}$$

$$\boxed{R'(x,y) = \exists z, t, V_4(x,z) \wedge V_3(x,t) \wedge \forall z' V_3(z',z) \Rightarrow V_4(z',y)}$$

R' is also a rewriting :

$$\forall D, R(\mathbf{V}(D)) = R'(\mathbf{V}(D))$$

R and R' behave differently outside of view images :

$$\exists E, R(E) \neq R'(E)$$

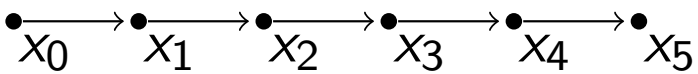
Example 2 (Afrati)

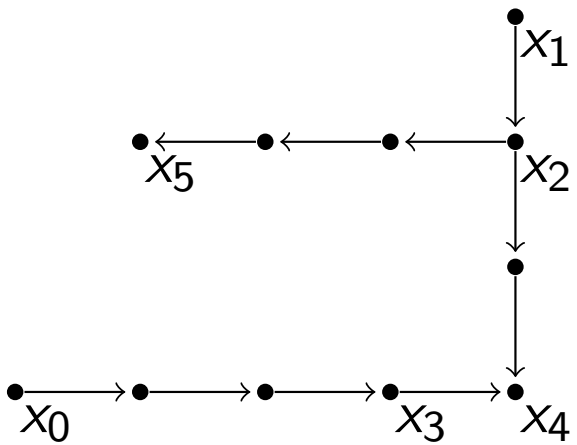
$$V_3 = a^3 \quad V_4 = a^4 \quad Q = a^5$$

$\mathbf{V} \twoheadrightarrow Q$ but f is not monotone.

$Q(D) = \{(x_0, x_5)\}$

$\mathbf{V}(D') \subseteq \mathbf{V}(D)$ but $Q(D) = \emptyset$





Monotone determinacy

- $\mathbf{V} \twoheadrightarrow Q$ and f is monotone.
- Equivalently :
  $\forall D, D' \qquad \mathbf{V}(D) \subseteq \mathbf{V}(D') \Rightarrow Q(D) \subseteq Q(D')$

Motivations

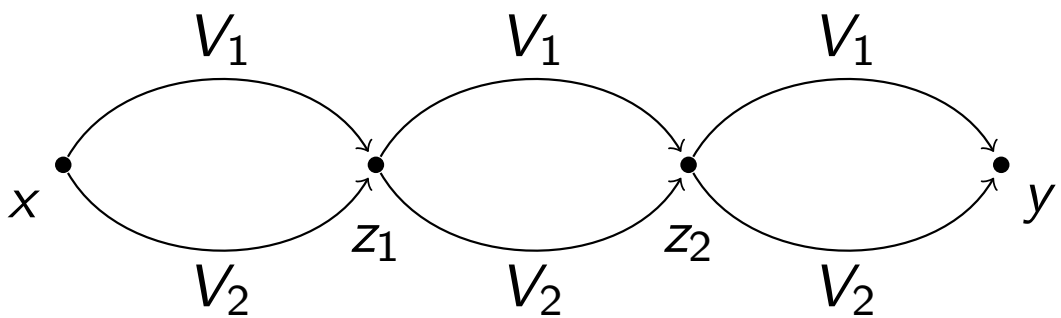
- Many interesting languages are monotone : conjunctive queries, regular path queries, Datalog, etc.
- [Nash et al.] : for CQ Views and Queries, monotone determinacy corresponds to the existence of a CQ rewriting.
- [Calvanese et al.] : decidability for RPQ Views and Queries.

Example 3

$V_1 = a \mid aa \quad V_2 = aa \mid aaa \quad Q = a(a^6)^* \mid aa(a^6)^*$

$$\boxed{R(x,y) = \exists z \; V_1(x,z) \wedge T^*(z,y)}$$

where $T(x,y)$ is defined as :



Our goal is to prove the following theorem :

Theorem

If **V** and Q are RPQs and **V** determines Q in a monotone way, then there exists a Datalog rewriting of Q using **V**.

Corollary

Given an RPQ query Q and RPQ views **V**, it is deciable whether there exists a Datalog rewriting of Q using **V**.

Certain Answers and CSP

Certain answers (Abiteboul, Duschka)

$$cert_{Q,\mathbf{V}}(E) = \bigcap_{D \mid E \subseteq \mathbf{V}(D)} Q(D)$$

Remarks

- $cert_{Q,\mathbf{V}}$ is a rewriting.
- BUT $cert_{Q,\mathbf{V}}$ is coNP-hard to evaluate.
  (even if $\mathbf{V} \twoheadrightarrow Q$ in a monotone way)
- This does not prevent the existence of PTime rewritings.

Example 4

$$V_1 = \{rg, gr, rb, br, gb, bg\}$$
$$\mathsf{Q} = \{\Sigma^* \cdot \alpha_1\beta_1 \cdot \alpha_2\beta_2 \cdot \Sigma^* \mid \beta_1 \neq \alpha_2\}$$

Let D be a database such that, for all $(x, y) \in \mathsf{D}$, $(x, y) \in V_1(\mathsf{D}) \Leftrightarrow (y, x) \in V_1(\mathsf{D})$.

Then $(x, y) \in \mathit{cert}_{\mathsf{Q},\mathbf{V}}(\mathbf{V}(\mathsf{D}))$ iff D is not 3-colorable.

Example 4

$$V_1 = \{rg, gr, rb, br, gb, bg\} \qquad V_2 = \mathsf{Q}$$
$$\mathsf{Q} = \{\Sigma^* \cdot \alpha_1\beta_1 \cdot \alpha_2\beta_2 \cdot \Sigma^* \mid \beta_1 \neq \alpha_2\}$$

Let D be a database such that, for all $(x, y) \in \mathsf{D}$, $(x, y) \in V_1(\mathsf{D}) \Leftrightarrow (y, x) \in V_1(\mathsf{D})$.

Then $(x, y) \in cert_{\mathsf{Q},\mathbf{V}}(\mathbf{V}(\mathsf{D}))$ iff D is not 3-colorable.

A trivial rewriting is $\mathsf{R} = V_2$.
But $cert_{\mathsf{Q},\mathbf{V}}$ is still as hard to evaluate.

Constraint Satisfaction Problem

- Fix a template structure T.
- $\mathrm{E} \in \mathrm{CSP}(T)$ iff $\mathrm{E} \xrightarrow{h} T$.

From CSP to queries

- Extend T with source and target nodes
- $(\mathrm{E}, u, v) \in \mathrm{CSP}(T)$ iff there exists h such that
  - $\mathrm{E} \xrightarrow{h} T$
  - $h(u)$ is a source node
  - $h(v)$ is a target node
- $\mathrm{CSP}(T)$ selects all u, v such that $(\mathrm{E}, u, v) \in \mathrm{CSP}(T)$

Theorem (Calvanese et al.)

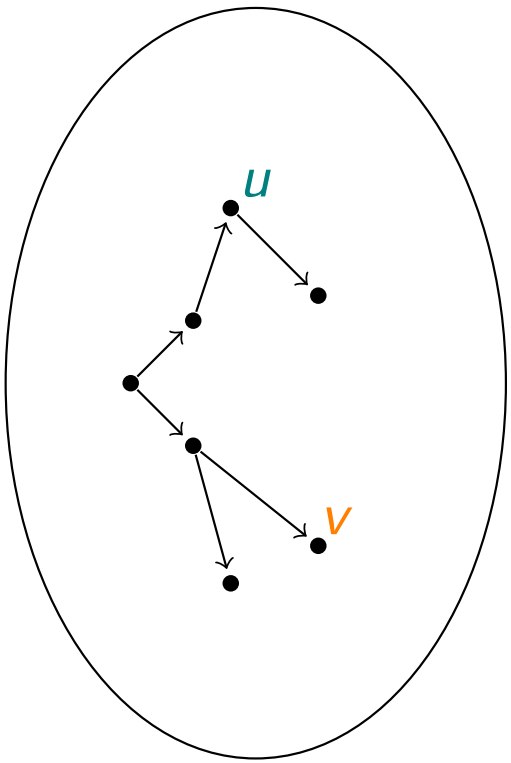
There exists a structure $T_{Q,\mathbf{V}}$ such that, for all D,

$$(u, v) \in cert_{Q,\mathbf{V}}(\mathbf{V}(D)) \Leftrightarrow (\mathbf{V}(D), u, v) \notin \mathrm{CSP}(T_{Q,\mathbf{V}})$$

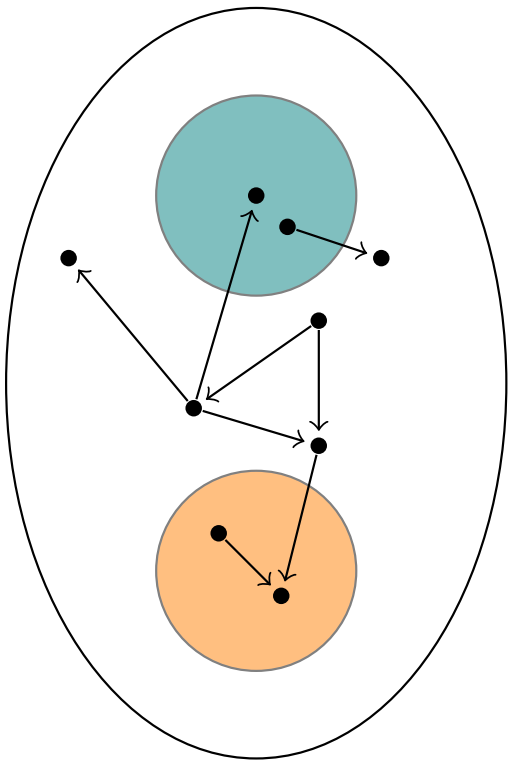
From CSP to Datalog

(l, k)-two-player-game

Arena : E, u, v and T with source and target nodes.



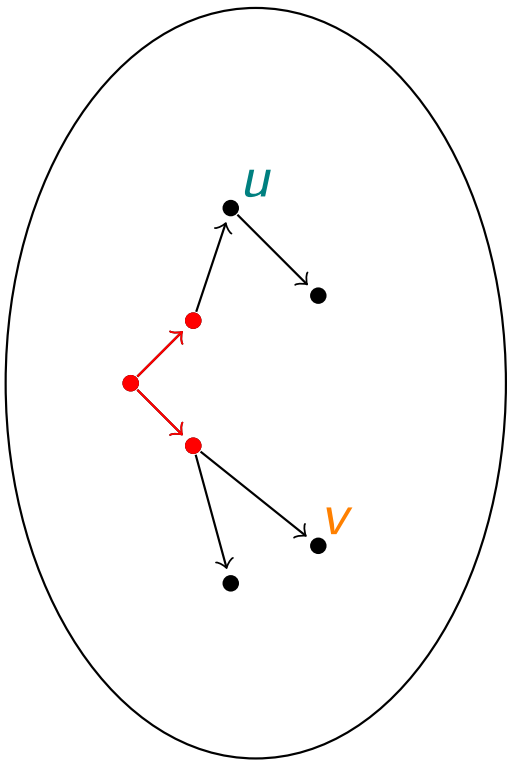
E



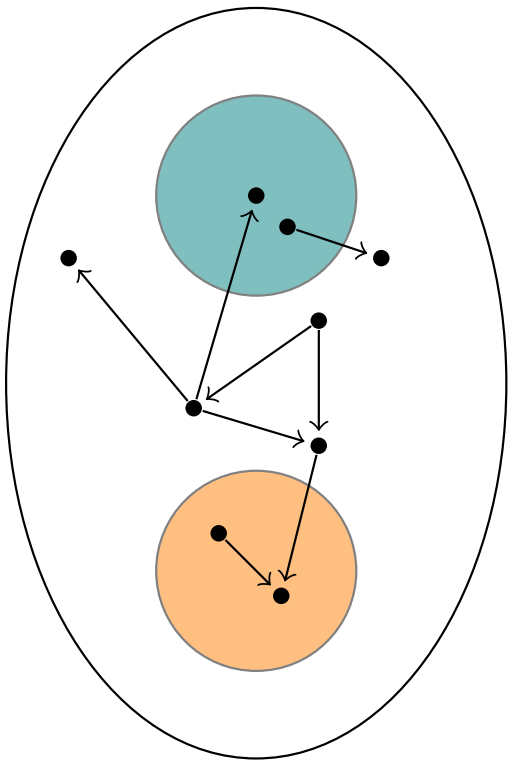
T

(l, k)-two-player-game

Player 1 starts by selecting k nodes on E.



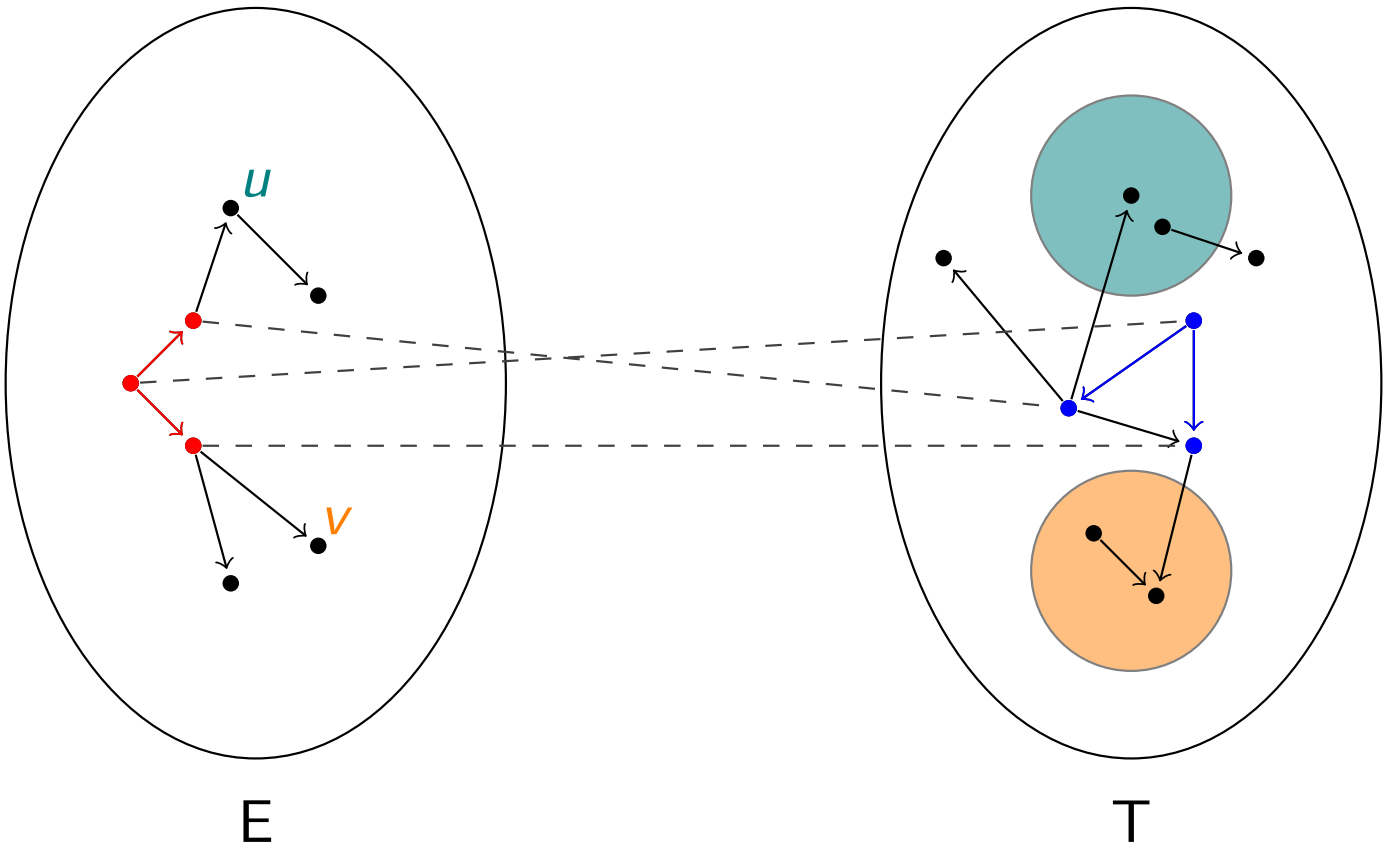
E



T

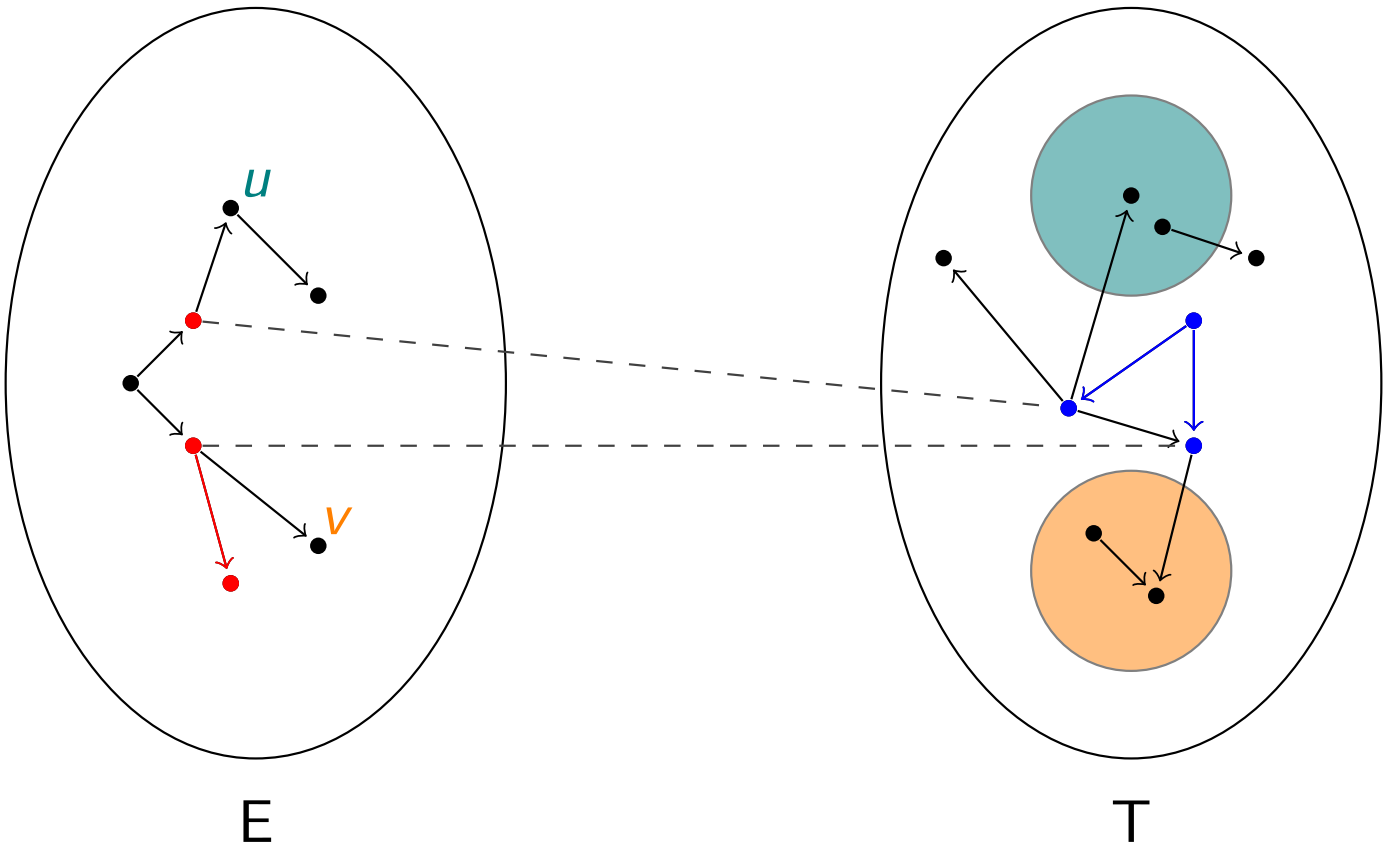
(l, k)-two-player-game

Player 2 provides a homomorphism from those nodes to T.



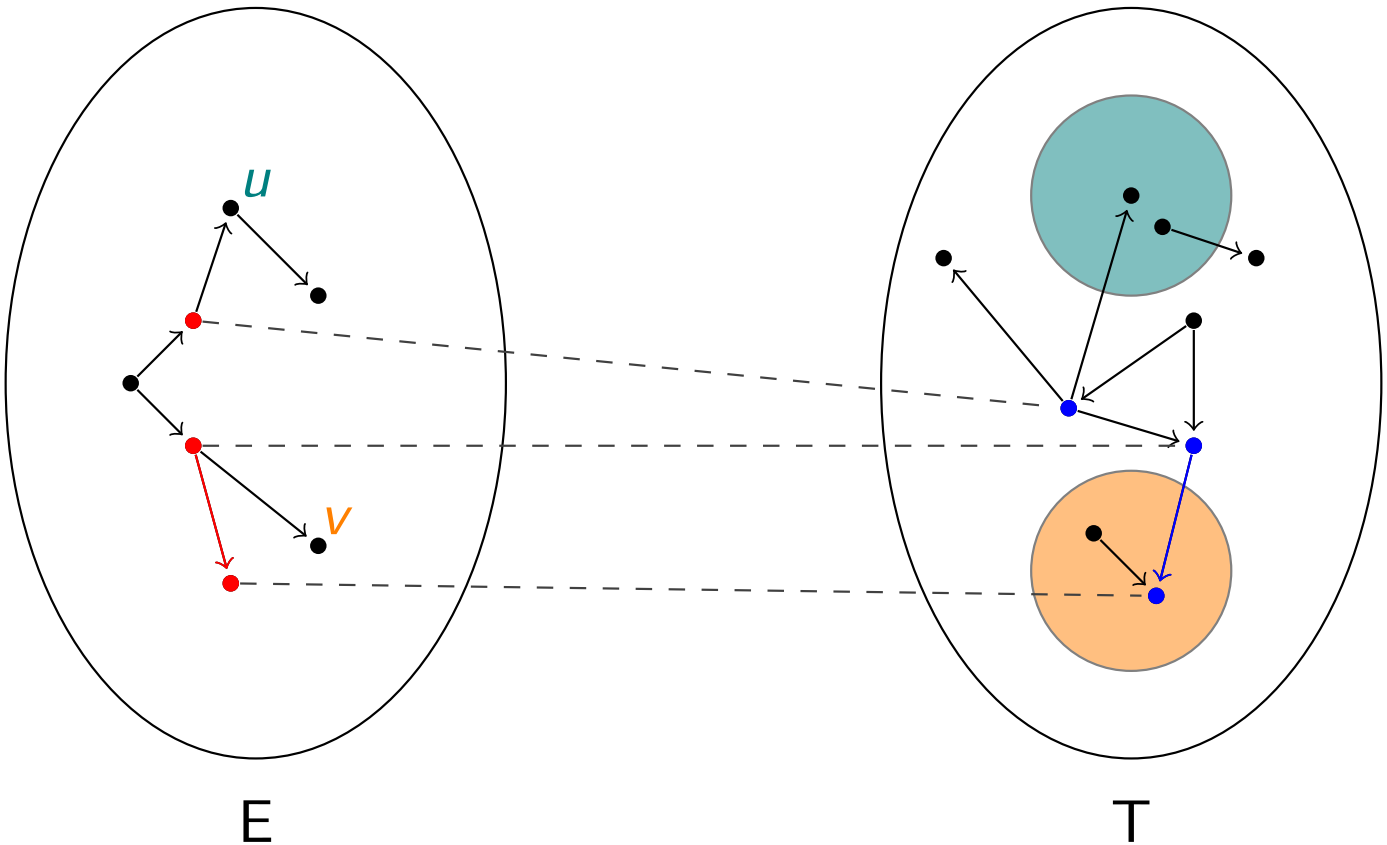
(l, k)-two-player-game

Player 1 selects k nodes, keeping at most l previous nodes.



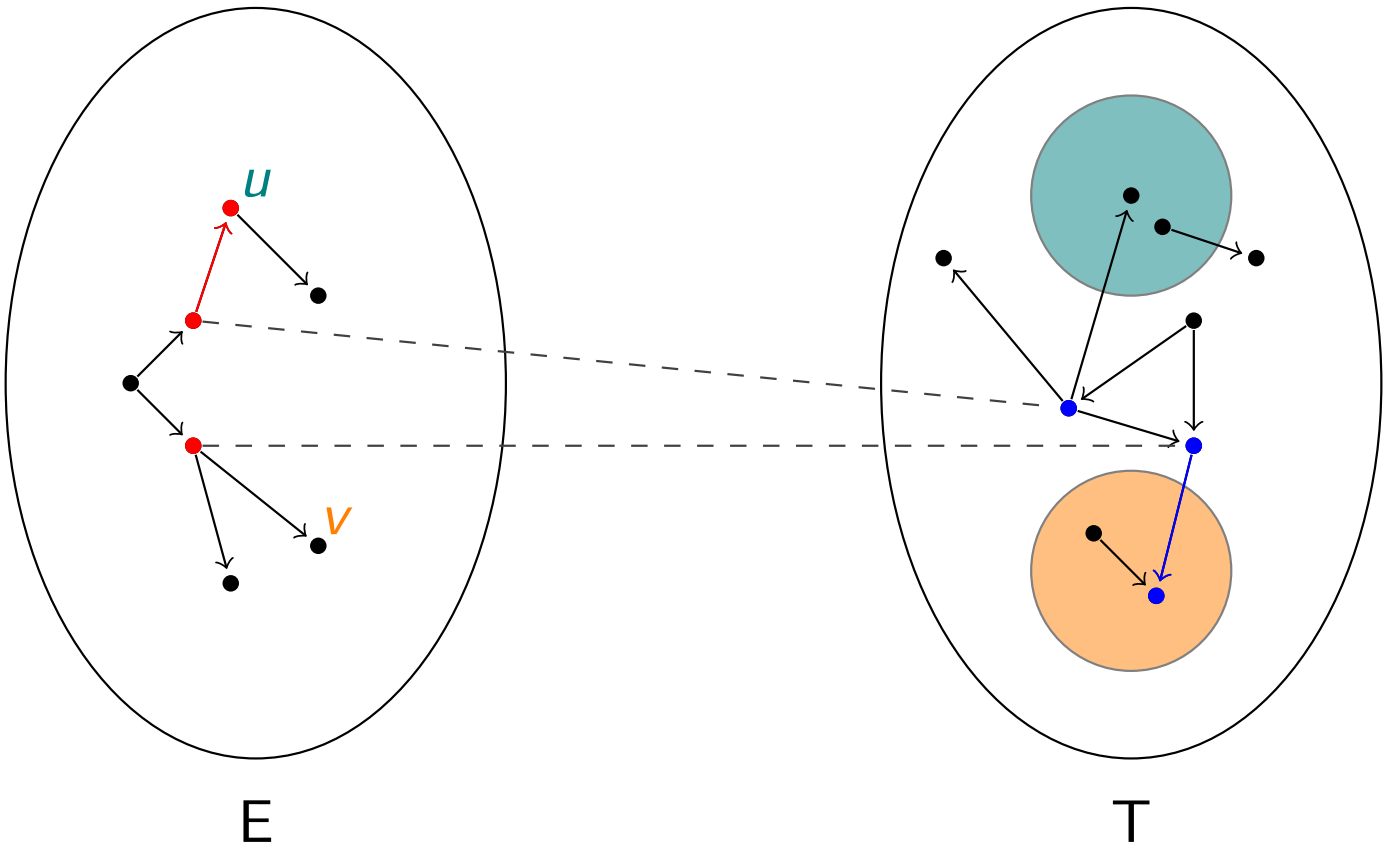
(l, k)-two-player-game

Player 2 extends previous homomorphism to new nodes.



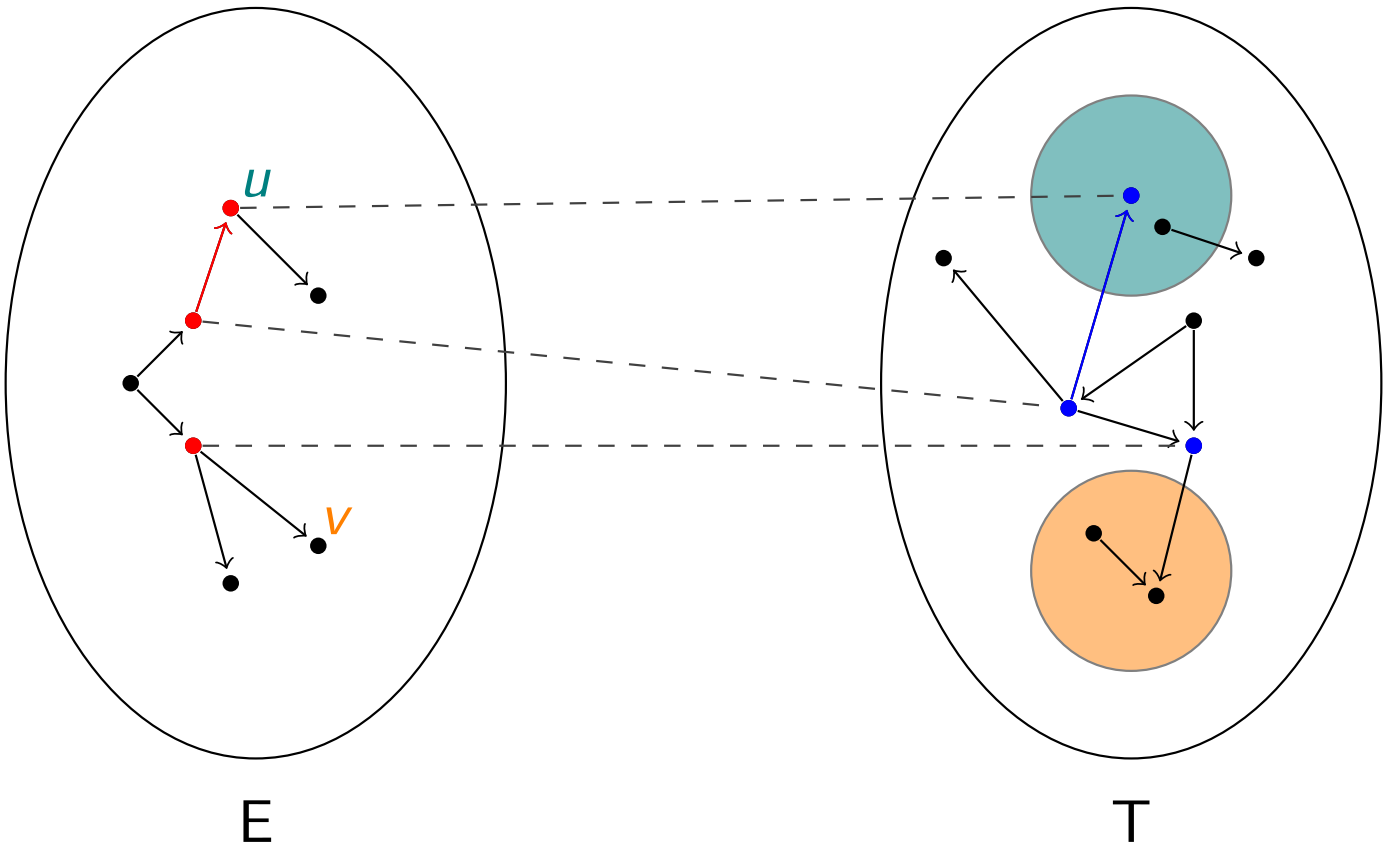
(l, k)-two-player-game

If Player 1 selects u...



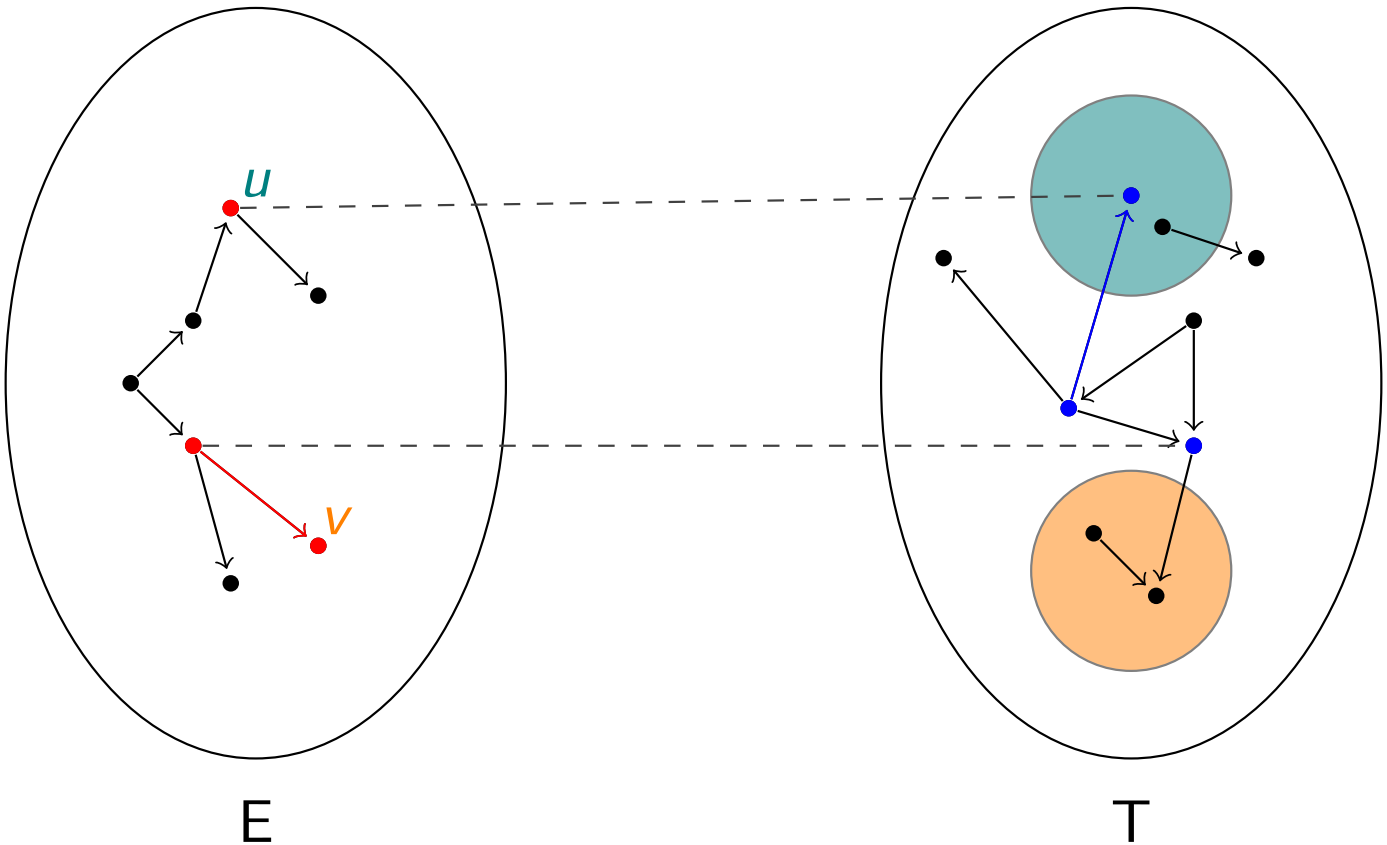
(l, k)-two-player-game

Then Player 2 must send it to a source node.



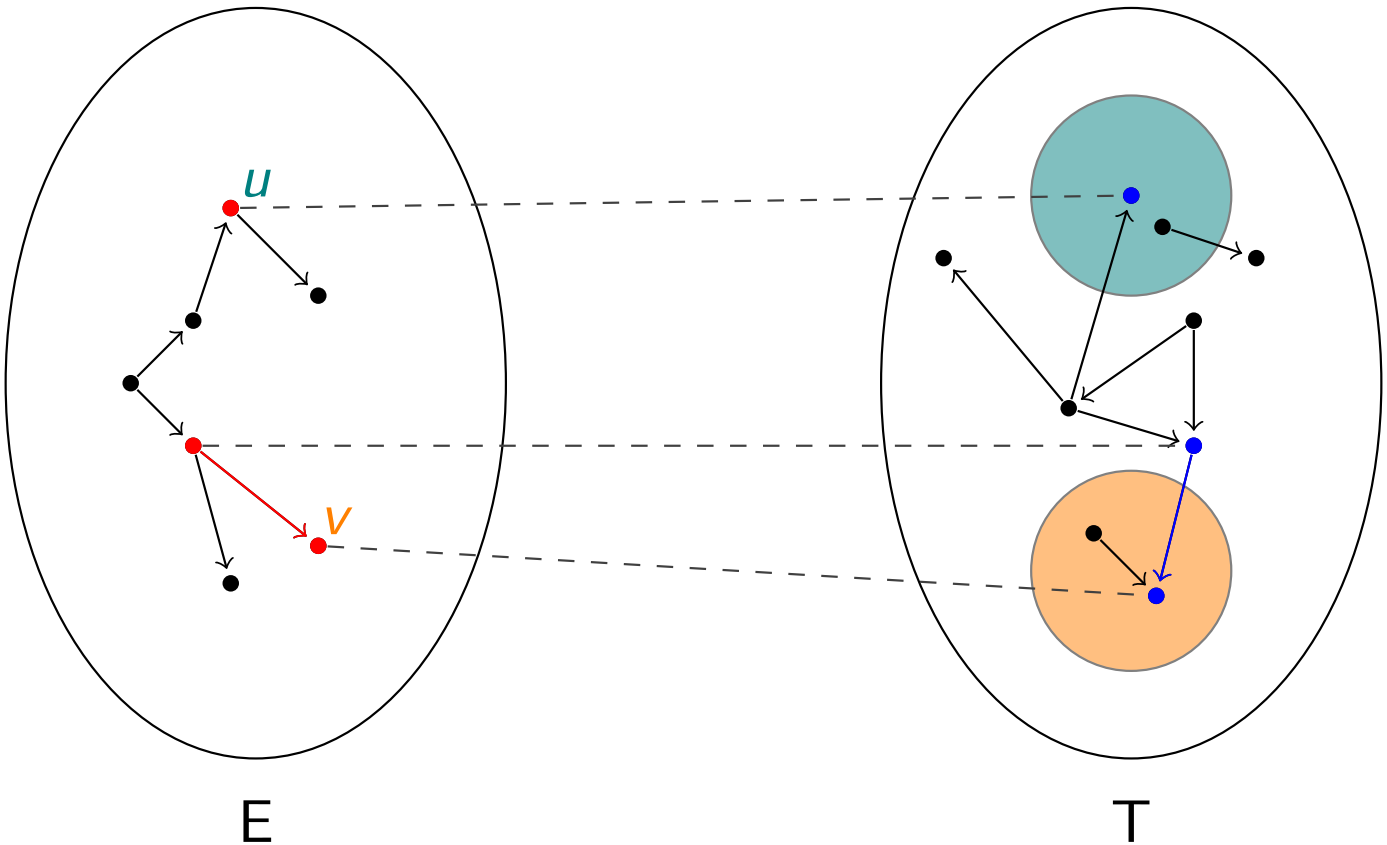
(l, k)-two-player-game

Similarly, if Player 1 selects v...



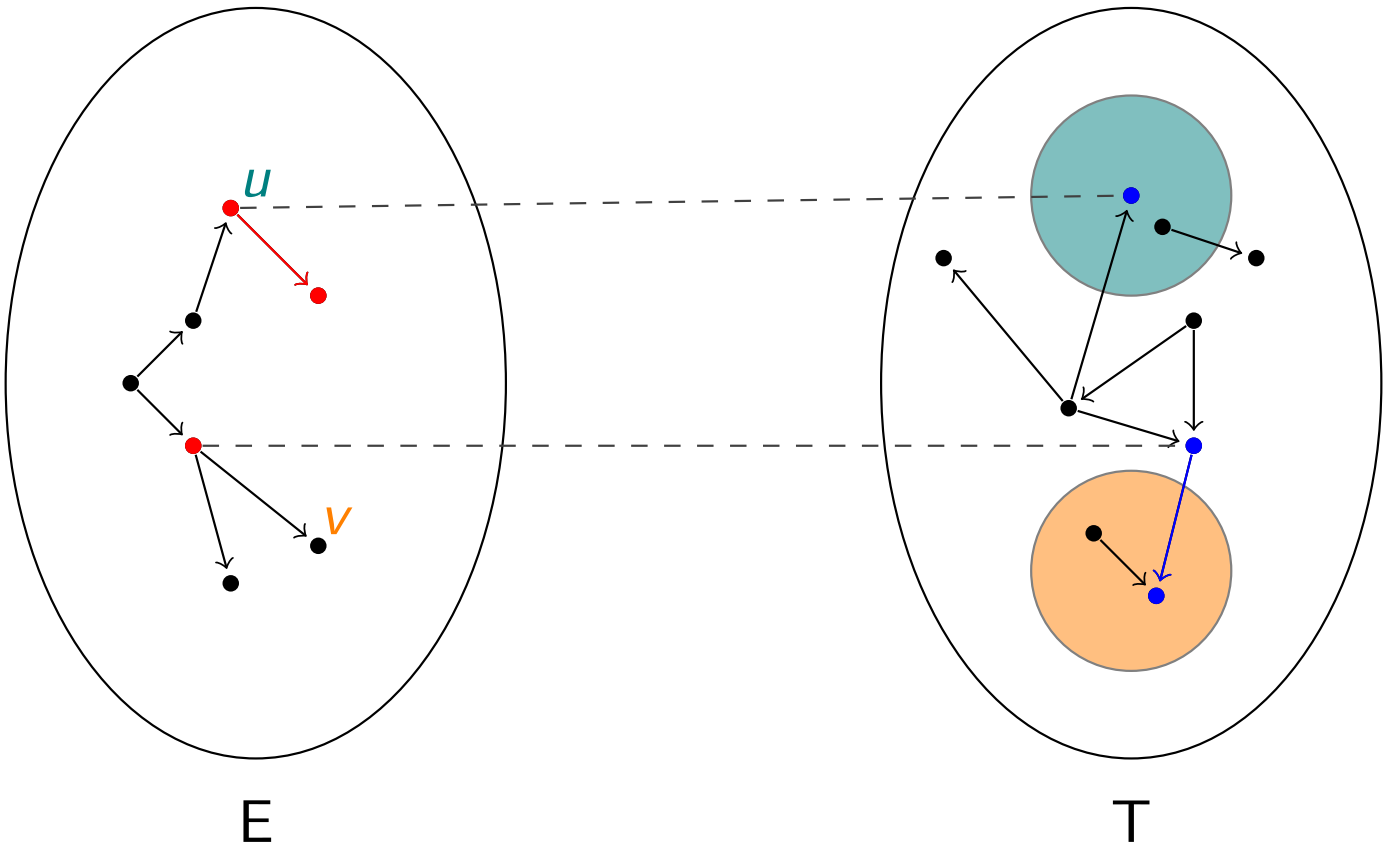
(l, k)-two-player-game

Then Player 2 must send it to a target node.



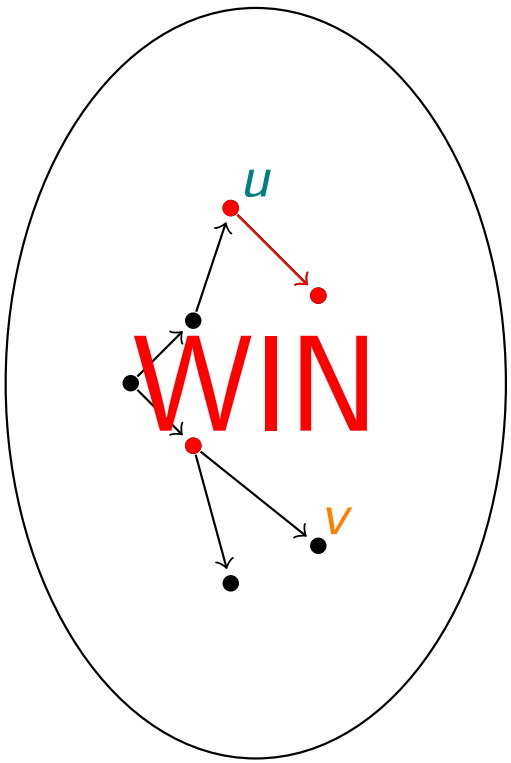
(l, k)-two-player-game

If Player 2 has no legal move...

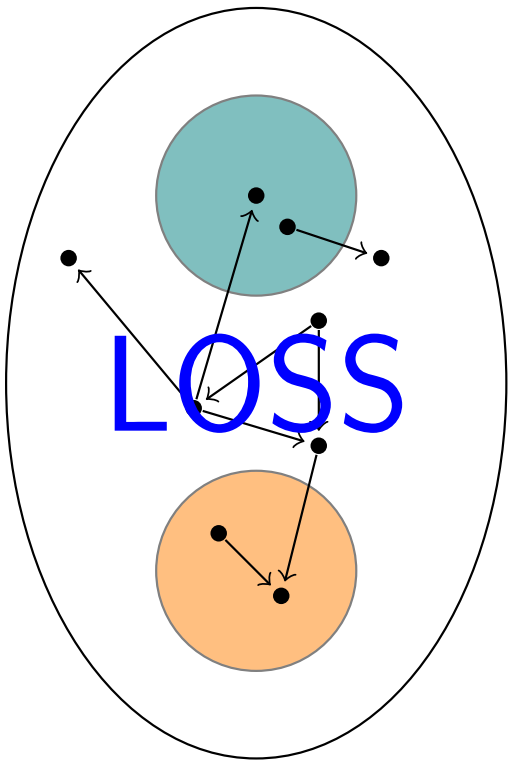


(l, k)-two-player-game

Then Player 1 wins.



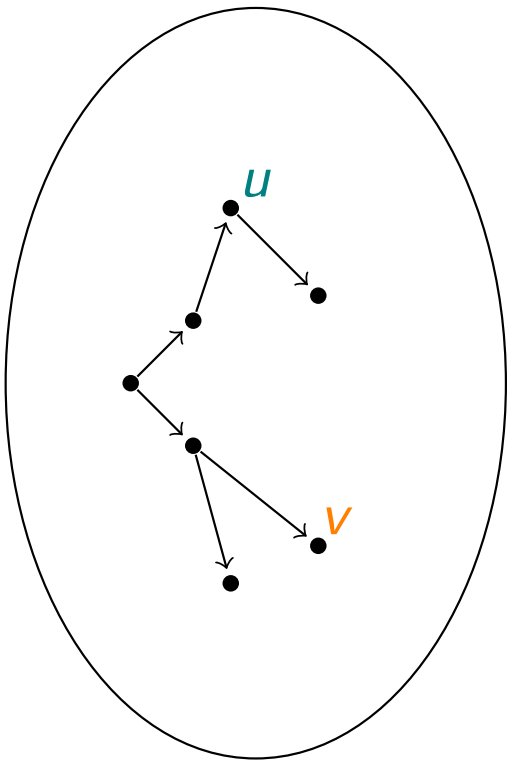
E



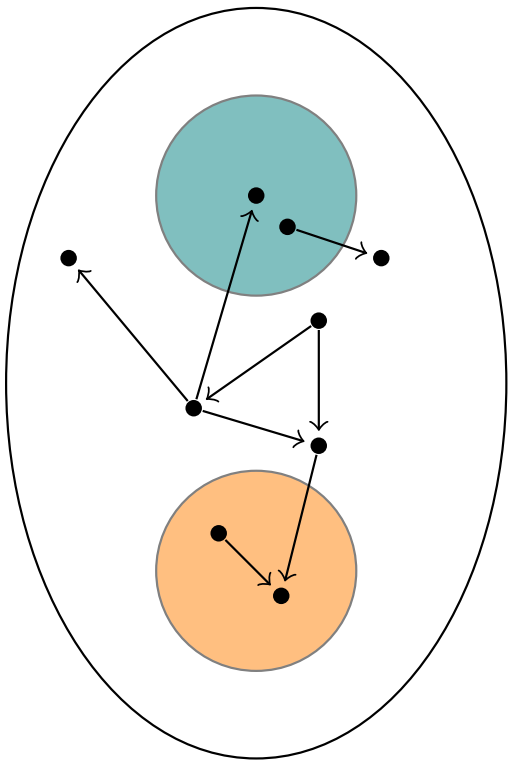
T

(l, k)-two-player-game

Player 2 wins if he can play forever.



E



T

Remark

- The (l, k)-two-player game on $T_{\mathrm{Q},\mathbf{V}}$ is an over-approximation of $\mathrm{CSP}(T_{\mathrm{Q},\mathbf{V}})$.
- If $(E, u, v) \in \mathrm{CSP}(T_{\mathrm{Q},\mathbf{V}})$, then Player 2 has a winning strategy.
- It is an under-approximation of $cert_{\mathrm{Q},\mathbf{V}}$.
- If Player 1 has a winning strategy, then $(u, v) \in cert_{\mathrm{Q},\mathbf{V}}(E)$.

Theorem (Feder, Vardi)

There exists a $\mathrm{Datalog}_{l,k}$ program $Q_{l,k}$ such that $(u, v) \in Q_{l,k}(\mathrm{E})$ if and only if Player 1 has a winning strategy for the game played on (E, u, v) and $T_{\mathrm{Q},\mathbf{V}}$.

The case of simple paths

- $\exists l, k$ such that $Q_{l,k}$ is exact on views of simple paths.
- $(u, v) \in Q_{l,k}(\mathbf{V}(\pi))$ if and only if $(u, v) \in cert_{Q,\mathbf{V}}(\mathbf{V}(\pi))$.
- This is the technical part of the proof !

Lifting paths to arbitrary databases

Assume that some candidate rewriting R is :

- Exact on views of simple paths.
- Sound on views of arbitrary databases.
- Closed under homomorphism.

Then it is exact on views of arbitrary databases.

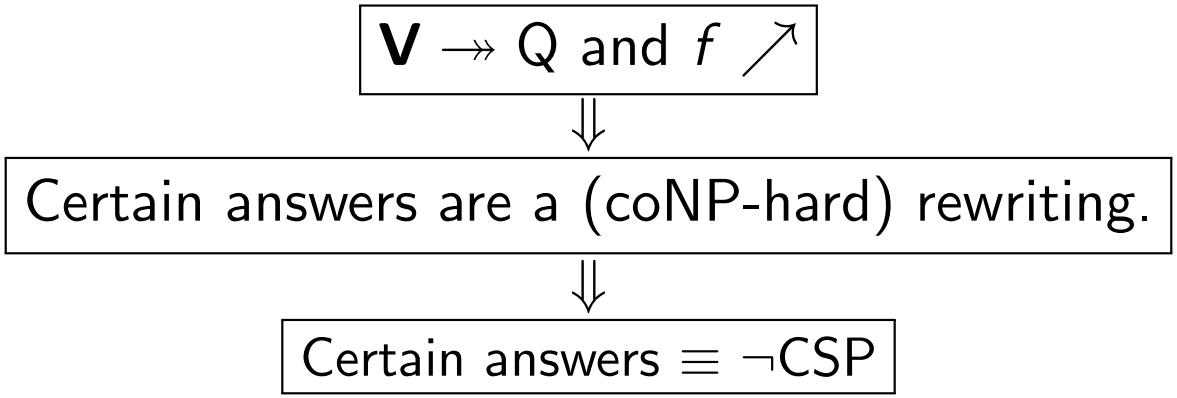
Conclusion and Perspectives

$\mathbf{V} \twoheadrightarrow \mathrm{Q}$ and $f \nearrow$

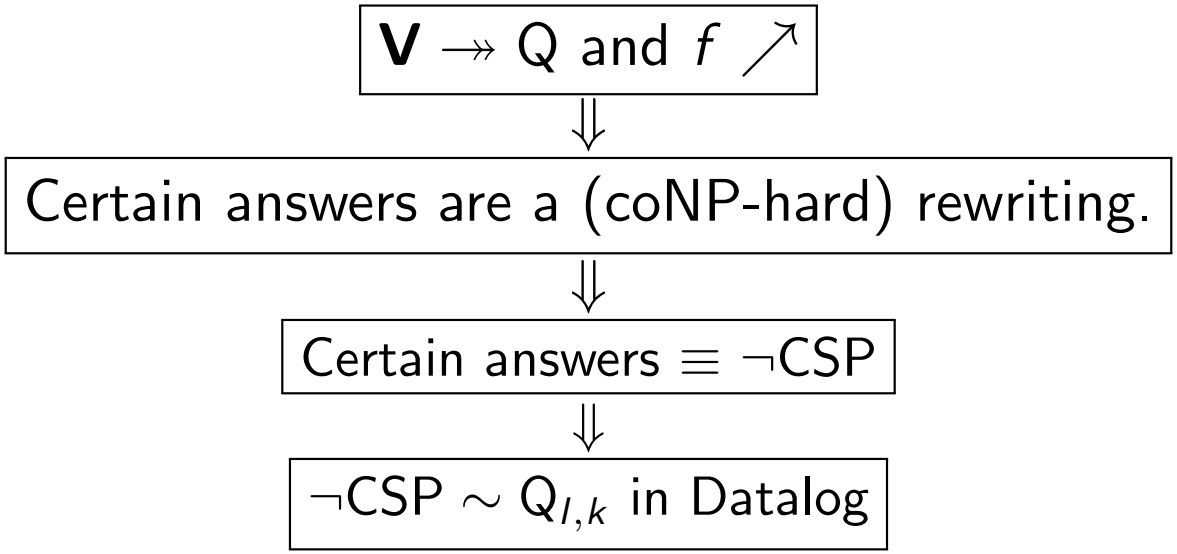
$\mathbf{V} \twoheadrightarrow \mathrm{Q}$ and $f \nearrow$

$\Downarrow$

Certain answers are a (coNP-hard) rewriting.

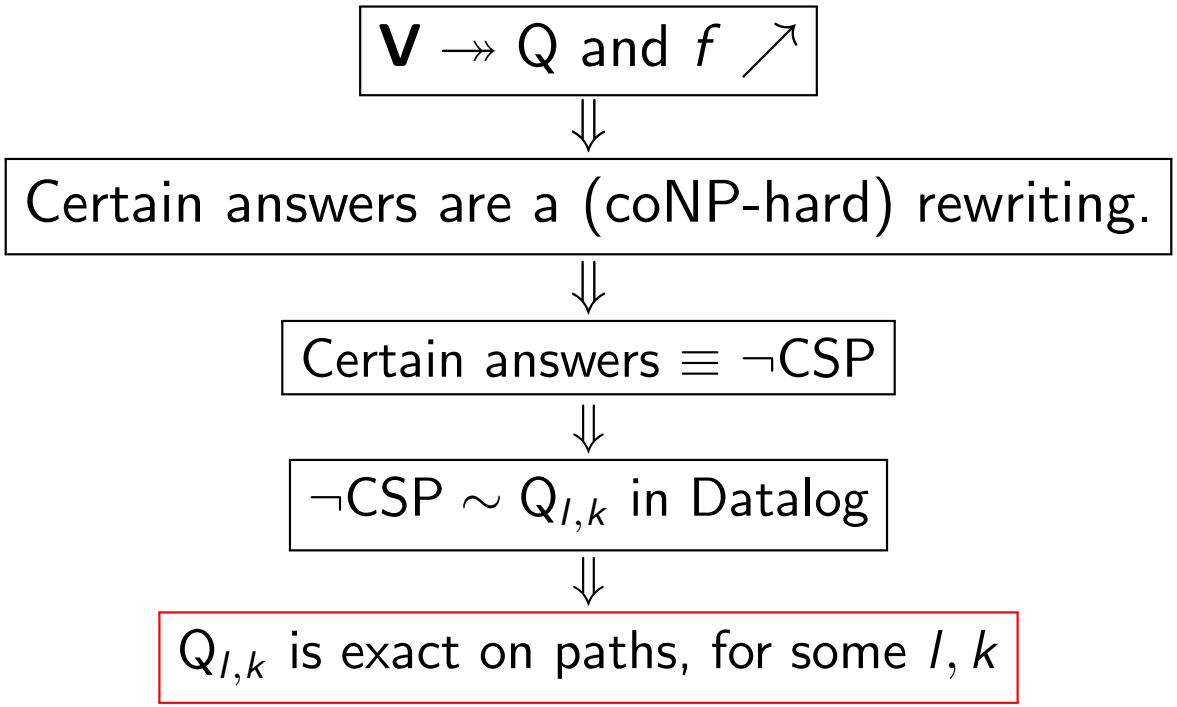


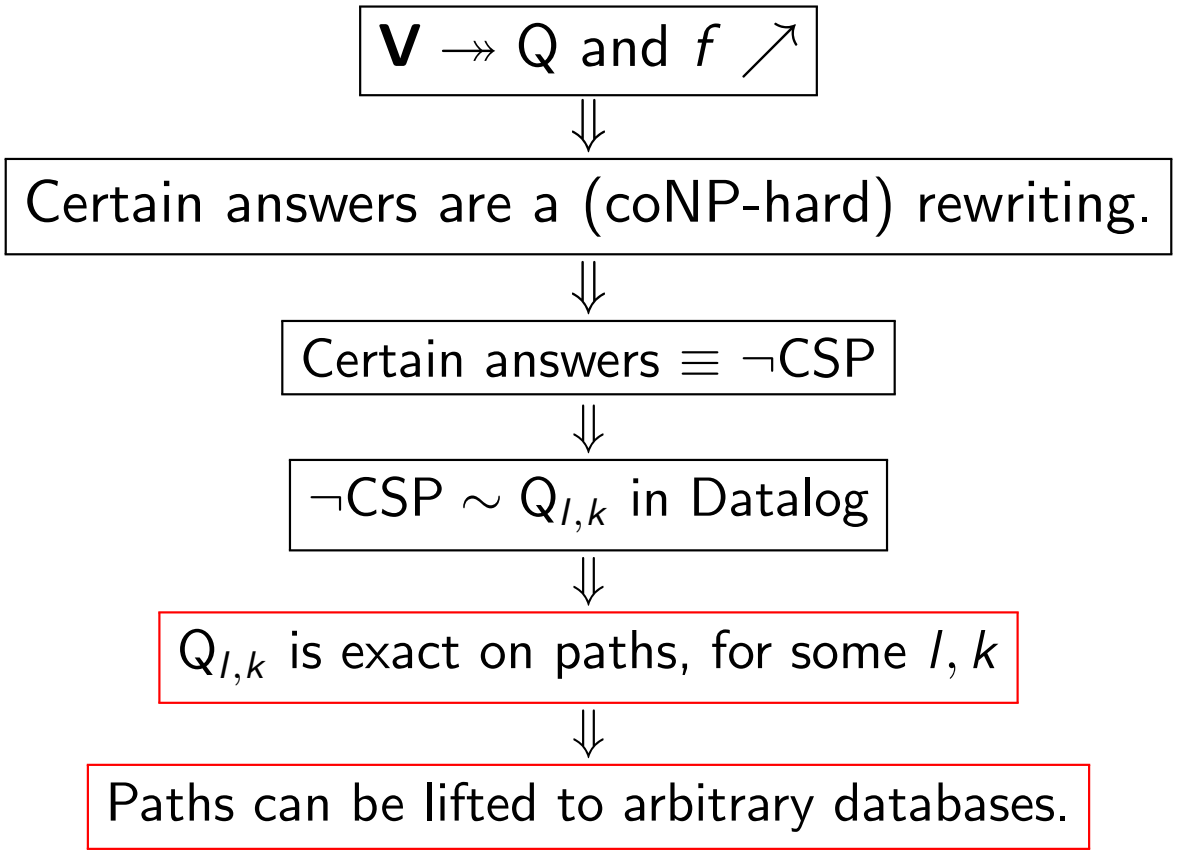
[Calvanese et al.]

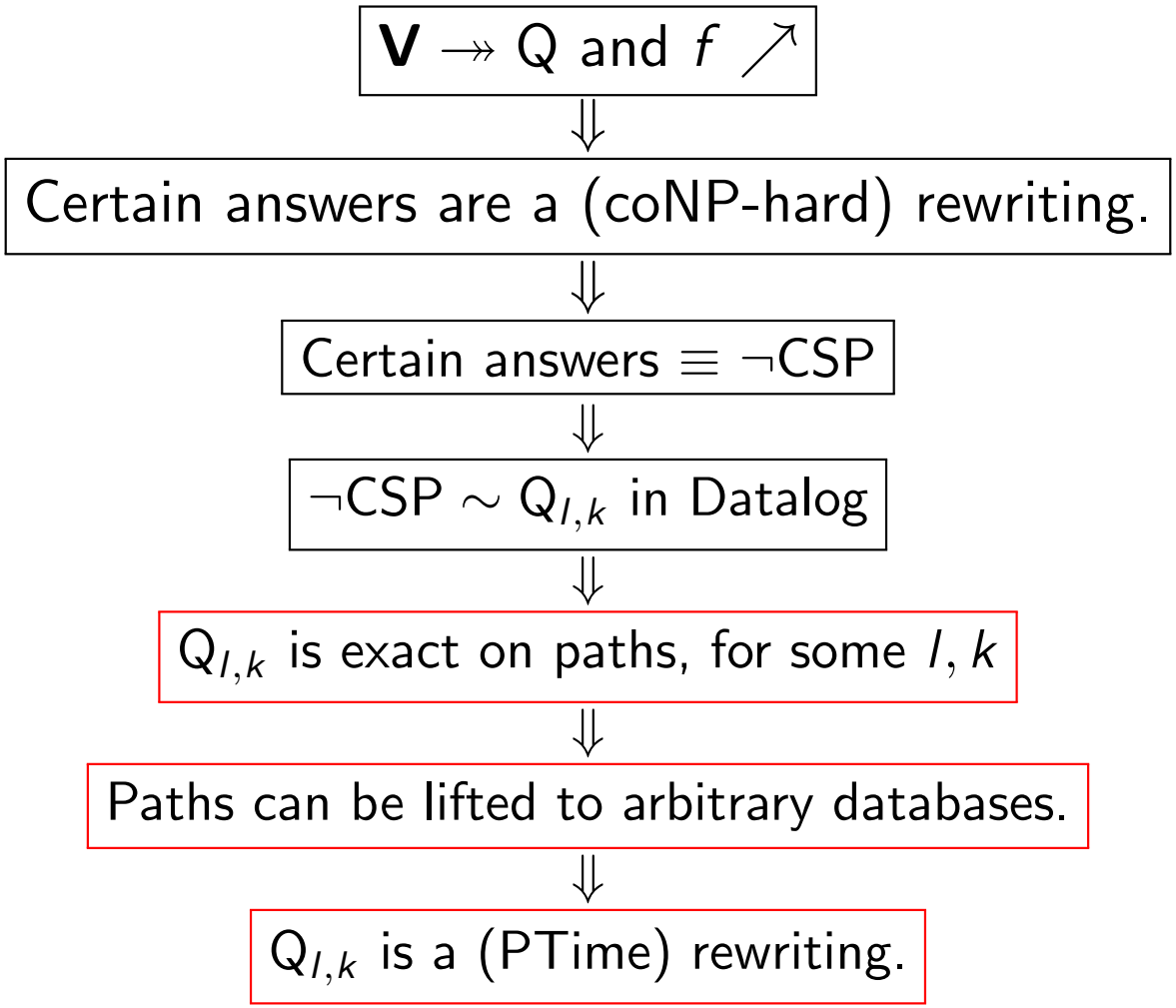


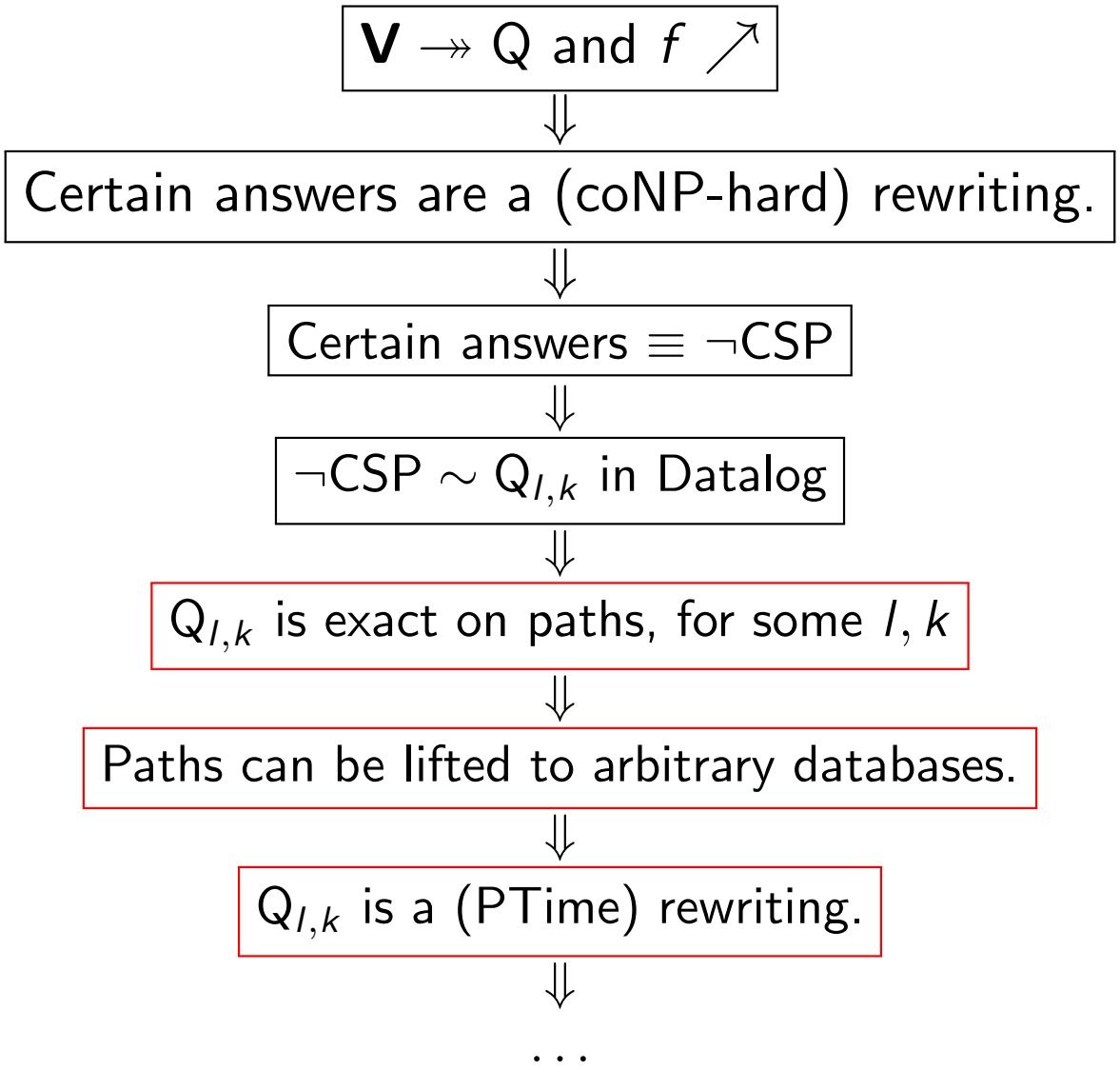
[Calvanese et al.]

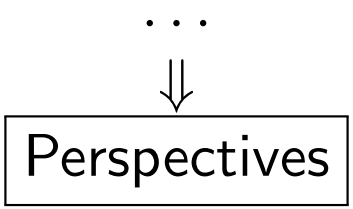
[Feder, Vardi]

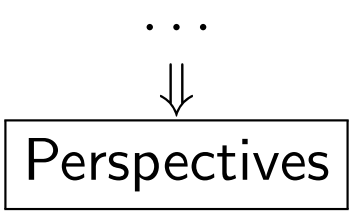

$\mathbf{V} \twoheadrightarrow \mathsf{Q}$ and $f \nearrow$
Certain answers are a (coNP-hard) rewriting.
Certain answers $\equiv \neg$CSP
[Calvanese et al.]
$\neg$CSP $\sim \mathsf{Q}_{l,k}$ in Datalog
[Feder, Vardi]
$\mathsf{Q}_{l,k}$ is exact on paths, for some l, k


$\mathbf{V} \twoheadrightarrow \mathsf{Q}$ and $f \nearrow$
Certain answers are a (coNP-hard) rewriting.
Certain answers $\equiv \neg$CSP
[Calvanese et al.]
$\neg$CSP $\sim \mathsf{Q}_{l,k}$ in Datalog
[Feder, Vardi]
$\mathsf{Q}_{l,k}$ is exact on paths, for some l, k
Paths can be lifted to arbitrary databases.


$\mathbf{V} \twoheadrightarrow \mathsf{Q}$ and $f \nearrow$
Certain answers are a (coNP-hard) rewriting.
Certain answers $\equiv \neg$CSP
[Calvanese et al.]
$\neg$CSP $\sim \mathsf{Q}_{l,k}$ in Datalog
[Feder, Vardi]
$\mathsf{Q}_{l,k}$ is exact on paths, for some l, k
Paths can be lifted to arbitrary databases.
$\mathsf{Q}_{l,k}$ is a (PTime) rewriting.


$\mathbf{V} \twoheadrightarrow \mathsf{Q}$ and $f \nearrow$
Certain answers are a (coNP-hard) rewriting.
Certain answers $\equiv \neg$CSP
[Calvanese et al.]
$\neg$CSP $\sim \mathsf{Q}_{l,k}$ in Datalog
[Feder, Vardi]
$\mathsf{Q}_{l,k}$ is exact on paths, for some l, k
Paths can be lifted to arbitrary databases.
$\mathsf{Q}_{l,k}$ is a (PTime) rewriting.
...


. . .
Perspectives



Rewriting language

- $\text{Datalog}_{l,k}$ is not user-friendly.
- All examples in Linear Datalog.

View and query languages

- More expressive languages.
- Closed language ?

Non monotone determinacy

- [Afrati] : first-order rewriting for path views and queries.
- Work in progress : towards RPQs $\rightarrow$ add disjunction.