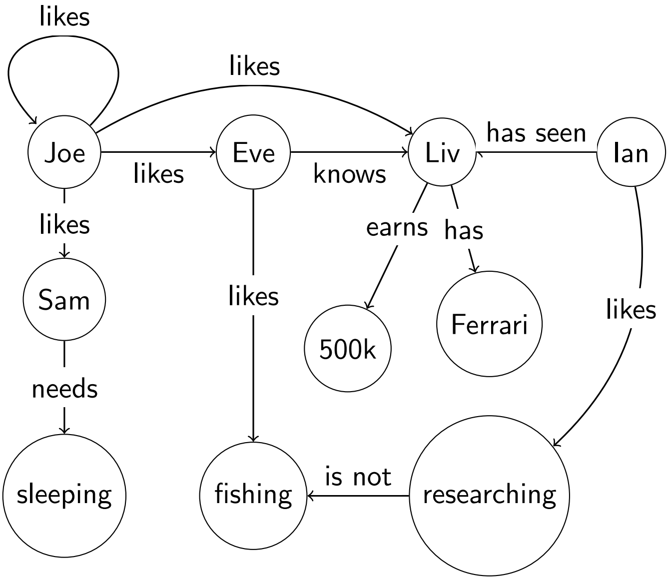
Querying Best Paths in Graph Databases

Jakub Michaliszyn, Jan Otop, Piotr Wieczorek

University of Wrocław, Poland

December 14, 2017

Graph Databases

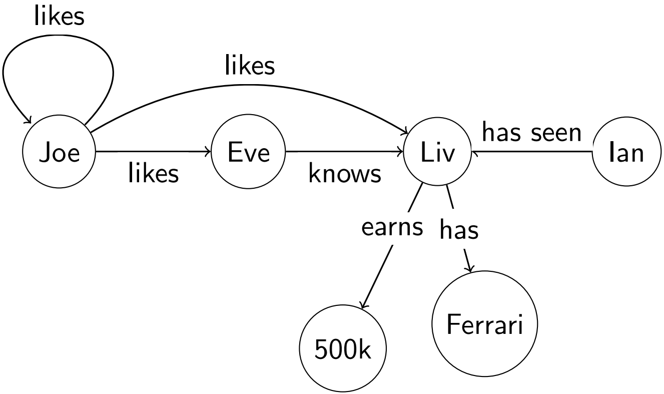


Regular Path Queries

$$x \to^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^* \cdot \mathrm{has})(\pi)$$

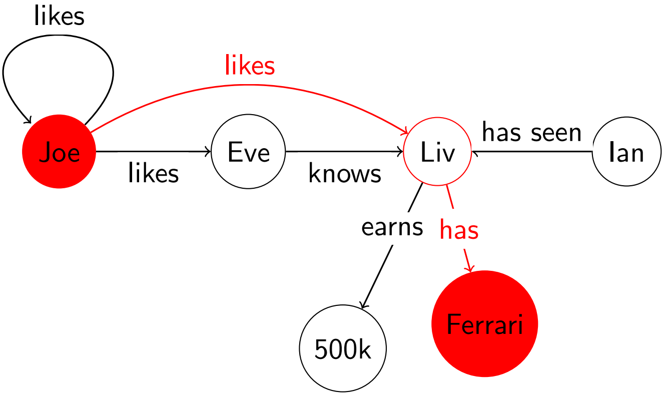
Regular Path Queries

$x \to^{\pi} y \wedge ((\text{likes} + \text{knows})^{*} \cdot \text{has})(\pi)$



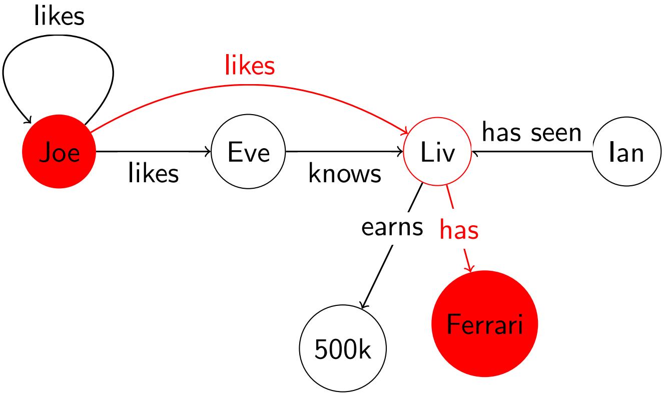
Regular Path Queries

$x \rightarrow^{\pi} y \wedge ((\text{likes} + \text{knows})^{*} \cdot \text{has})(\pi)$



Regular Path Queries

$x \rightarrow^{\pi} y \wedge ((\text{likes} + \text{knows})^{*} \cdot \text{has})(\pi)$



Evaluation

- PSPACE-complete (combined complexity)
- NL-complete (data complexity)

Extensions of RPQs

- conjunctions of RPQs, inverses,

Extensions of RPQs

- conjunctions of RPQs, inverses,
- comparing paths: ECRPQs (regular relations), $+$linear constraints (Barcelo, Libkin, Lin, Wood),

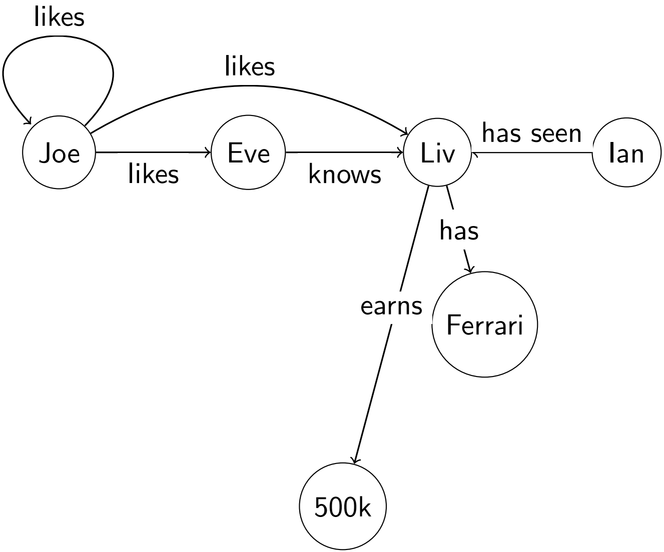
Extensions of RPQs

- conjunctions of RPQs, inverses,
- comparing paths: ECRPQs (regular relations), +linear constraints (Barcelo, Libkin, Lin, Wood),
- data values (properties)?
  - register automata and regular queries with memory (Vrgoč and Libkin)
  - LARE (arithmetical regular expressions) (Graboń, Michaliszyn, Otop, Wieczorek)

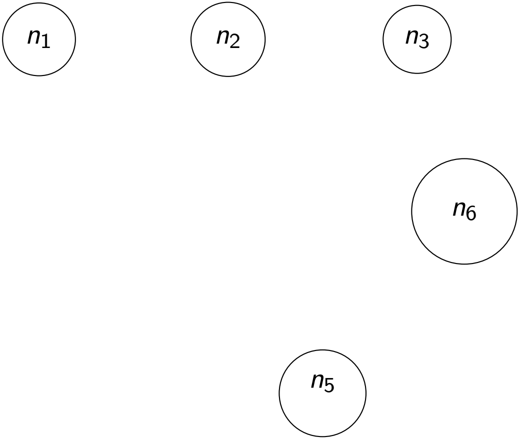
Our approach: (O)PRA

- the comparison of paths and the use of data values,
- aggregation of data values along paths,
- computation of extremal values among aggregated data.
- modular structure (views/ontologies),
- good expressive power: subsumes earlier approaches,
- good complexity: PSPACE-complete (combined) and NL-complete (data),

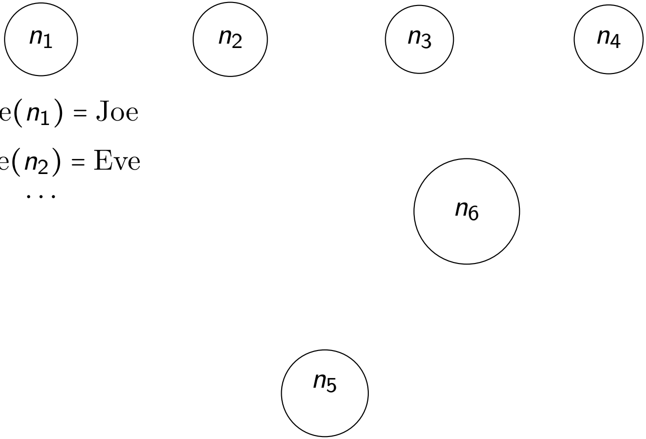
Graph Databases: labellings



Graph Databases: labellings



Graph Databases: labellings



$\text{name}(n_1) = \text{Joe}$

$\text{name}(n_2) = \text{Eve}$

$\dots$

Graph Databases: labellings

$$\text{name}(n_1) = \text{Joe}$$

$$\text{name}(n_2) = \text{Eve}$$

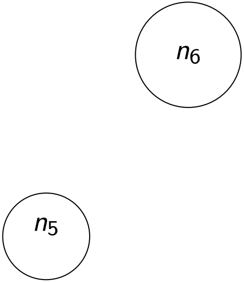
$$\dots$$

$$\text{edge}(n_1, n_2) = \mathit{true}$$

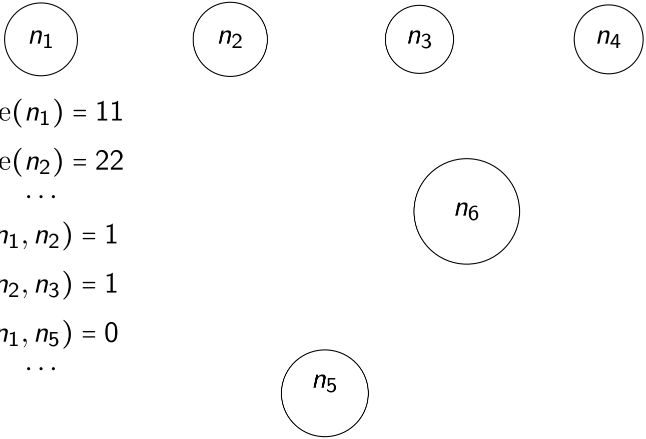
$$\text{edge}(n_2, n_3) = \mathit{true}$$

$$\text{edge}(n_1, n_5) = \mathit{false}$$

$$\dots$$

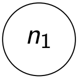


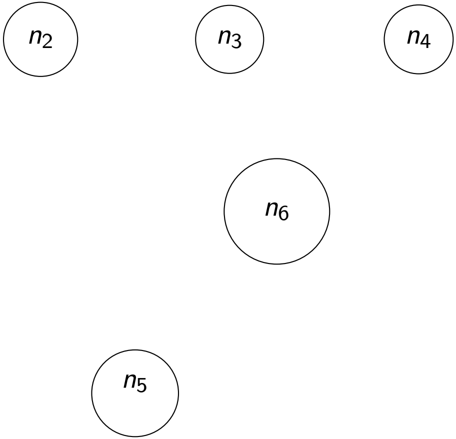
Graph Databases: labellings



$$\mathrm{name}(n_1) = 11$$
$$\mathrm{name}(n_2) = 22$$
$$\ldots$$
$$\mathrm{edge}(n_1, n_2) = 1$$
$$\mathrm{edge}(n_2, n_3) = 1$$
$$\mathrm{edge}(n_1, n_5) = 0$$
$$\ldots$$

Graph Databases: labellings





$$\mathrm{name}(n_1) = 11$$
$$\mathrm{name}(n_2) = 22$$
$$\dots$$
$$\mathrm{edge}(n_1, n_2) = 1$$
$$\mathrm{edge}(n_2, n_3) = 1$$
$$\mathrm{edge}(n_1, n_5) = 0$$
$$\dots$$
$$\mathrm{attr}(n_3) = -108$$

Building blocks: Node constraints

- $x \to^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^* \cdot \mathrm{has})(\pi)$
  Regular Path Queries - regular expressions over labels of edges

Building blocks: Node constraints

- $x \to^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^* \cdot \mathrm{has})(\pi)$
  Regular Path Queries - regular expressions over labels of edges
- More general alphabet: testing data values in nodes?, edges?, labellings values for tuples of nodes?

Building blocks: Node constraints

- $x \rightarrow^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^{*} \cdot \mathrm{has})(\pi)$
  Regular Path Queries - regular expressions over labels of edges
- More general alphabet: testing data values in nodes?, edges?, labellings values for tuples of nodes?
- We can access special vars:
  $\mathbf{C}_1$ is the current node,
  $\mathbf{C}'_1$ is the next node (a kind of look-ahead).

Building blocks: Node constraints

- $x \to^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^{*} \cdot \mathrm{has})(\pi)$
  Regular Path Queries - regular expressions over labels of edges
- More general alphabet: testing data values in nodes?, edges?, labellings values for tuples of nodes?
- We can access special vars:
  $\mathbf{C}_1$ is the current node,
  $\mathbf{C}_1'$ is the next node (a kind of look-ahead).
- Node constraints (new alphabet):
  $\langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_1') = 1 \rangle$,
  $\langle \mathrm{type}(\mathbf{C}_1) = 7 \rangle$

Building blocks: Node constraints

- $x \to^{\pi} y \wedge ((\mathrm{likes} + \mathrm{knows})^* \cdot \mathrm{has})(\pi)$
  Regular Path Queries - regular expressions over labels of edges
- More general alphabet: testing data values in nodes?, edges?, labellings values for tuples of nodes?
- We can access special vars:
  $\mathbf{C}_1$ is the current node,
  $\mathbf{C}'_1$ is the next node (a kind of look-ahead).
- Node constraints (new alphabet):
  $\langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}'_1) = 1 \rangle$,
  $\langle \mathrm{type}(\mathbf{C}_1) = 7 \rangle$
- In general: $\langle X \leq X' \rangle, \langle X < X' \rangle, \langle X = X' \rangle$
  where X, X' are integer constants or labellings applied to vars (including $\mathbf{C}_i$).

Building blocks: Regular constraints

Regular expressions over a path:

- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C_1'}) \rangle^* \langle \top \rangle (\pi)$

Building blocks: Regular constraints

Regular expressions over a path:

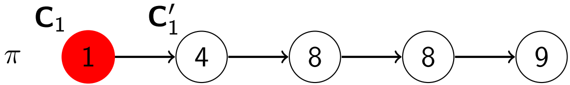
- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C'_1}) \rangle^* \langle \top \rangle (\pi)$

$$\pi \qquad 1 \longrightarrow 4 \longrightarrow 8 \longrightarrow 8 \longrightarrow 9$$

Building blocks: Regular constraints

Regular expressions over a path:

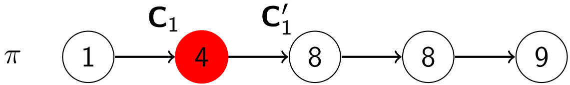
- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C_1'}) \rangle^* \langle \top \rangle (\pi)$



Building blocks: Regular constraints

Regular expressions over a path:

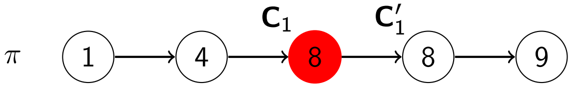
- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C'_1}) \rangle^* \langle \top \rangle (\pi)$



Building blocks: Regular constraints

Regular expressions over a path:

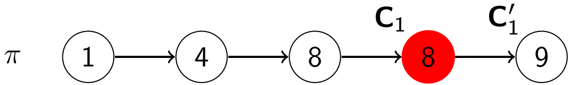
- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C_1'}) \rangle^* \langle \top \rangle (\pi)$



Building blocks: Regular constraints

Regular expressions over a path:

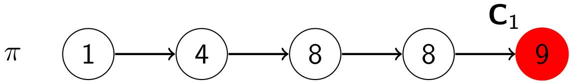
- $\langle \mathrm{attr}(\mathbf{C}_1) \leq \mathrm{attr}(\mathbf{C}'_1) \rangle^* \langle \top \rangle (\pi)$



Building blocks: Regular constraints

Regular expressions over a path:

- $\langle \mathrm{attr}(\mathbf{C_1}) \leq \mathrm{attr}(\mathbf{C_1'}) \rangle^* \langle \top \rangle (\pi)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.

Building blocks: Regular constraints

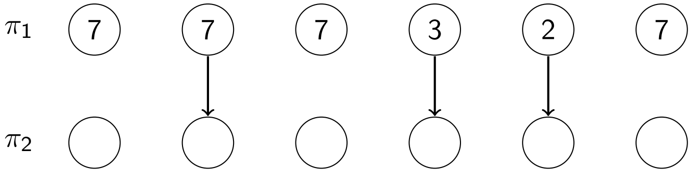
Regular expressions over a tuple of paths (like in ECRPQs)

- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$

Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

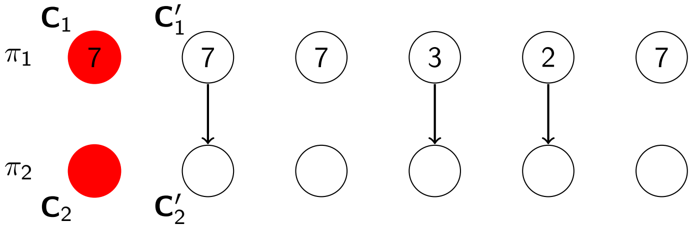
- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

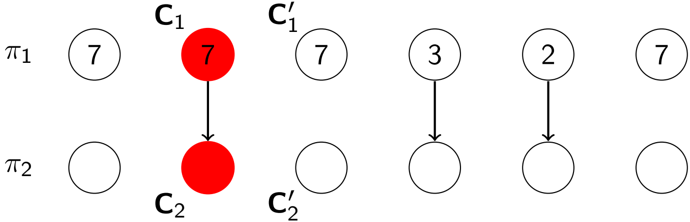
- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

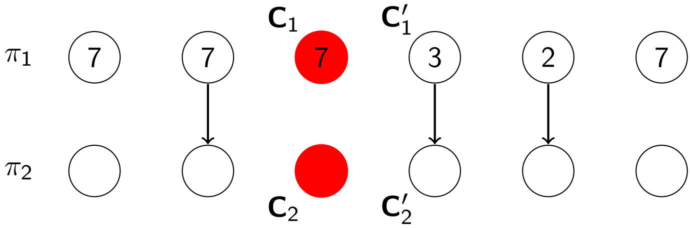
- Here, $\mathbf{C}_i$ is the current node on the *i*-th path
  $\mathbf{C}'_i$ is the next node on the *i*-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7 \rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1 \rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

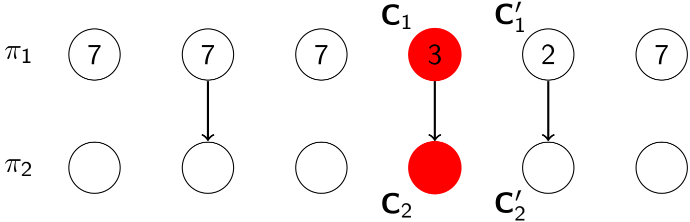
- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

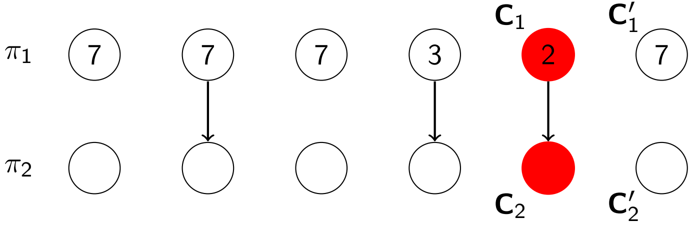
- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

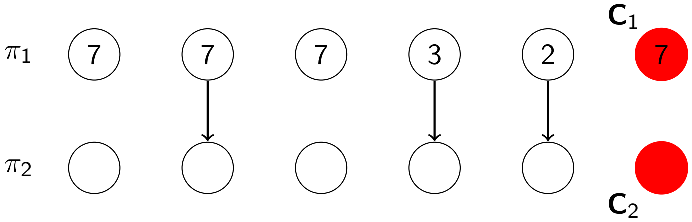
- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}_i'$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Regular constraints

Regular expressions over a tuple of paths (like in ECRPQs)

- Here, $\mathbf{C}_i$ is the current node on the i-th path
  $\mathbf{C}'_i$ is the next node on the i-th path.
- $(\langle \mathrm{type}(\mathbf{C}_1) = 7\rangle + \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1\rangle)^*(\pi_1, \pi_2)$



Building blocks: Arithmetical constraints

Allow to compare boolean combinations of sums along paths:

- $c_1 \Lambda_1 + \ldots + c_j \Lambda_j \leq c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.

Building blocks: Arithmetical constraints

Allow to compare boolean combinations of sums along paths:

- $c_1\Lambda_1 + \ldots + c_j\Lambda_j \le c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.
- Semantics: $\sum_{i=1}^{s} \mathrm{some\_labelling}(\pi_{i_1}[i], \ldots, \pi_{i_k}[i])$.

Building blocks: Arithmetical constraints

Allow to compare boolean combinations of sums along paths:

- $c_1\Lambda_1 + \ldots + c_j\Lambda_j \leq c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.
- Semantics: $\sum_{i=1}^{s} \mathrm{some\_labelling}(\pi_{i_1}[i], \ldots, \pi_{i_k}[i])$.

Examples

- $\mathrm{time}[\pi] \leq 10$ (total time to go over the path π is ≤ 10);

Building blocks: Arithmetical constraints

Allow to compare boolean combinations of sums along paths:

- $c_1\Lambda_1 + \ldots + c_j\Lambda_j \leq c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.
- Semantics: $\sum_{i=1}^{s} \mathrm{some\_labelling}(\pi_{i_1}[i], \ldots, \pi_{i_k}[i])$.

Examples

- $\mathrm{time}[\pi] \leq 10$ (total time to go over the path π is ≤ 10);
- $\mathrm{attr}[\pi] - \mathrm{one}[\pi] \leq 0$ (the average attractiveness of π is ≤ 1);

Building blocks: Arithmetical constraints

Allow to compare boolean combinations of sums along paths:

- $c_1 \Lambda_1 + \ldots + c_j \Lambda_j \leq c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.
- Semantics: $\sum_{i=1}^{s} \mathrm{some\_labelling}(\pi_{i_1}[i], \ldots, \pi_{i_k}[i])$.

Examples

- $\mathrm{time}[\pi] \leq 10$ (total time to go over the path π is ≤ 10);
- $\mathrm{attr}[\pi] - \mathrm{one}[\pi] \leq 0$ (the average attractiveness of π is ≤ 1);
- $\mathrm{time}[\pi_1] - \mathrm{time}[\pi_2] \leq 0$ (path π_1 is faster than π_2);

Building blocks: Arithmetical constraints

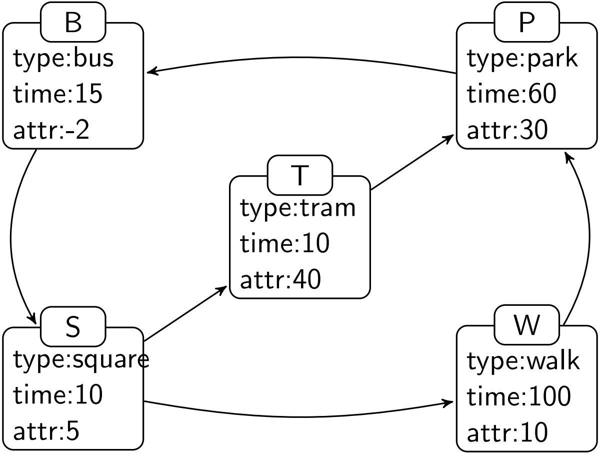
Allow to compare boolean combinations of sums along paths:

- $c_1 \Lambda_1 + \ldots + c_j \Lambda_j \leq c_0$, where c_i are constants and Λ_i is $\mathrm{some\_labelling}[\pi_{i_1}, \ldots, \pi_{i_k}]$.
- Semantics: $\sum_{i=1}^{s} \mathrm{some\_labelling}(\pi_{i_1}[i], \ldots, \pi_{i_k}[i])$.

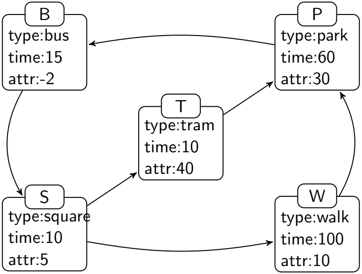
Examples

- $\mathrm{time}[\pi] \leq 10$ (total time to go over the path π is ≤ 10);
- $\mathrm{attr}[\pi] - \mathrm{one}[\pi] \leq 0$ (the average attractiveness of π is ≤ 1);
- $\mathrm{time}[\pi_1] - \mathrm{time}[\pi_2] \leq 0$ (path π_1 is faster than π_2);
- $\mathrm{edge}[\pi_1, \pi_2] \leq 5$ (number of edges between the corresponding places in π_1 and π_2 is ≤ 5).

Example map-representing graph

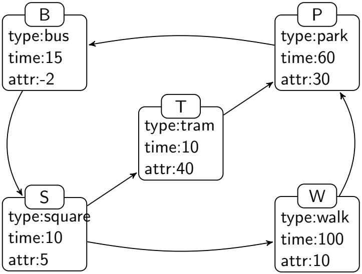


PRA Examples



$$\mathrm{route}(\pi) := \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_1') = 1 \rangle^* \langle \top \rangle (\pi)$$

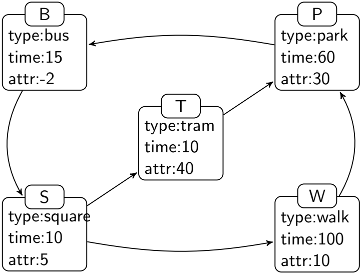
PRA Examples



$$\mathrm{route}(\pi) := \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_1') = 1\rangle^* \langle \top \rangle(\pi)$$

$$\texttt{MATCH NODES } (s,t) \texttt{ SUCH THAT } s \to^{\pi} t \texttt{ WHERE } \mathrm{route}(\pi)$$
$$\texttt{HAVING } \mathrm{time}[\pi] \leq 360 \wedge \mathrm{attr}[\pi] > 100$$

PRA Examples



$$
\begin{aligned}
&\texttt{MATCH NODES } (s,t) \texttt{ SUCH THAT } s \rightarrow^{\pi} t\\
&\texttt{WHERE } \mathrm{route}(\pi) \wedge \langle \mathrm{type}(\mathbf{C}_1) = c_{\mathrm{tram}} \rangle^*(\rho) \wedge\ (\langle \mathrm{type}(\mathbf{C}_1) = c_{\mathrm{bus}} \rangle +\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathrm{type}(\mathbf{C}_1) = c_{\mathrm{walk}} \rangle +\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathrm{type}(\mathbf{C}_1) = c_{\mathrm{tram}} \rangle +\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathrm{edge}(\mathbf{C}_1, \mathbf{C}_2) = 1 \rangle)^*(\pi, \rho)
\end{aligned}
$$

OPRA: defining new labellings

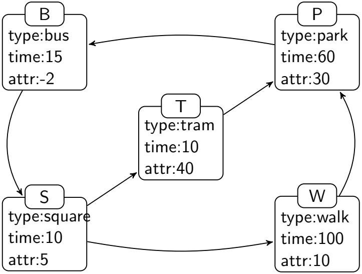
Values for new labellings are specified with *terms*.

$$t(\vec{x}) ::= c \mid \lambda(\vec{y}) \mid [Q(\vec{y})] \mid \min_{\lambda,\pi} Q(\vec{y},\pi) \mid \max_{\lambda,\pi} Q(\vec{y},\pi)$$
$$\mid y = y \mid f(t(\vec{y}),\ldots,t(\vec{y})) \mid f'(\{t(x) \colon t(x,\vec{y})\})$$

where $f, f' \in \{\textsc{Max}, \textsc{Min}, \textsc{Count}, \textsc{Sum}, +, -, \cdot, \leq$ (assuming 0 for false and 1 for true)

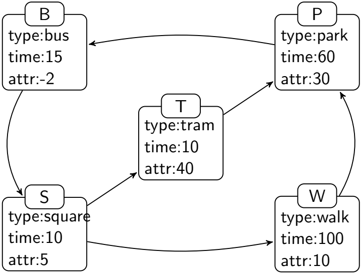
Then, we may write $\texttt{LET}\ \lambda_1{:=}t_1,\ldots,\lambda_n{:=}t_n\ \texttt{IN}\ Q$.

OPRA Example: new labelling



$\texttt{LET}\ \mathrm{walk\_time}(x) :=$
$(\mathrm{type}(x) = c_{\mathrm{walk}}) \cdot \mathrm{time}(x)\ \texttt{IN}$
$\texttt{MATCH NODES}\ (s, t)\ \texttt{SUCH THAT}\ s \to^{\pi} t$
$\texttt{WHERE}\ \mathrm{route}(\pi)\ \texttt{HAVING}\ \mathrm{walk\_time}[\pi] \leq 10$

OPRA Example: nested query



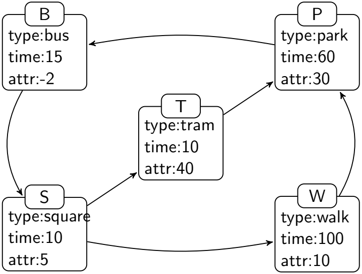
$\texttt{LET}\ \mathrm{crowded}(x) :=$

$\texttt{[MATCH NODES}\ (x)\ \texttt{SUCH THAT}\ x \to^{\pi} y\ \texttt{WHERE}$

$\mathrm{route}(\pi) \wedge \langle \top \rangle^{*} \langle \mathrm{attr}(\mathbf{C}_1) > 100 \rangle (\pi)\ \texttt{HAVING}\ \mathrm{time}[\pi] \leq 10\ \texttt{] IN}$

$\texttt{MATCH PATHS}\ (\pi)\ \texttt{WHERE}\ \mathrm{route}(\pi) \wedge \langle \mathrm{crowded}(\mathbf{C}_1) = 0 \rangle^{*} (\pi)$

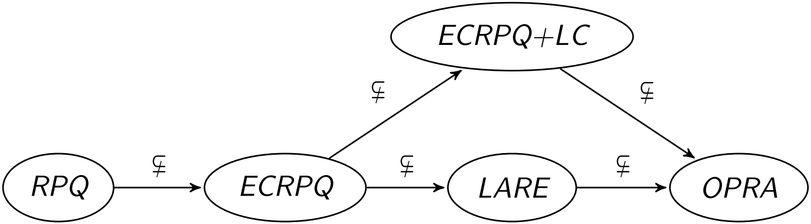
OPRA Example: most attractive but in minimum time



MATCH NODES (s, t) SUCH THAT $s \rightarrow^{\pi} t$ WHERE $\mathrm{route}(\pi)$
HAVING
$(\mathrm{attr}[\pi] = \max_{\mathrm{attr},\rho} Q_{\mathrm{route}}(s, t, \rho)) \wedge$
$(\mathrm{time}[\pi] = \min_{\mathrm{time},\rho} Q_{\mathrm{route}}(s, t, \rho))$

Our results

Theorem (Expressivity)



Theorem (Complexity)

Query answering for OPRA is PSPACE-complete (combined complexity) and NL-complete (data complexity).