Containment of Shape Expression Schemas for RDF

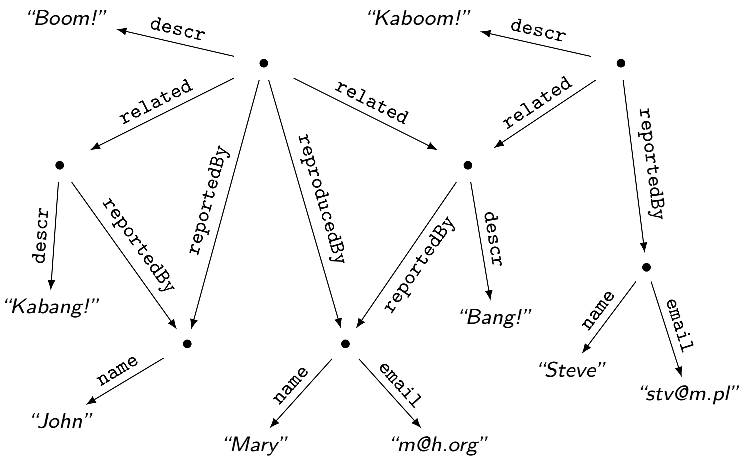
Sławek Staworko
University of Lille & INRIA LINKS
(joint work with P. Wieczorek, University of Wrocław)

Principles of Database Systems 2019

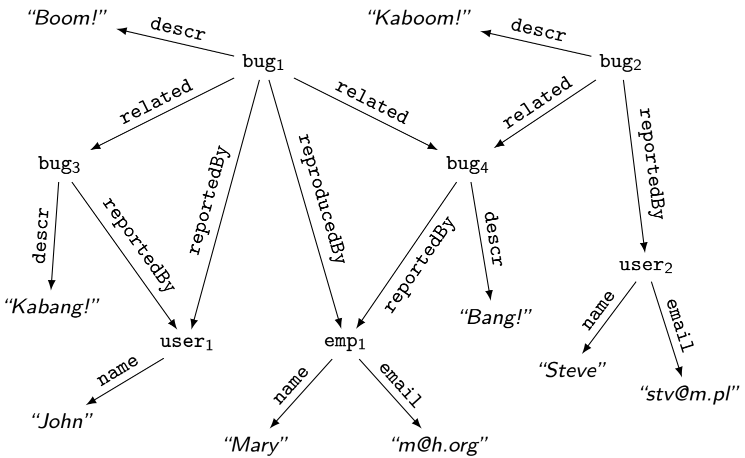
Amsterdam, Netherlands

2 July 2019

What is RDF and does it need schemas?



What is RDF and does it need schemas?



What is RDF and does it need schemas? (cont'd.)

Originally, free-range RDF

- The *driving* technology of Web 3.0
- *"Just publish your data so others can access it!"*
- Intentionally schema-free and ontology oriented (RDF Schema)

Nowadays, industrial-strength RDF

- Produced and consumed by applications (data exchange format)
- Often obtained from exporting data from relational databases (e.g., R2RML)
- Follows a strict structure

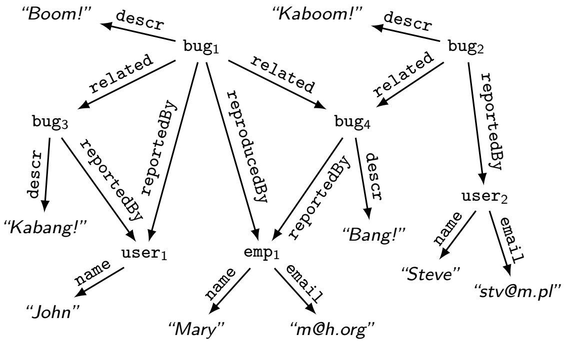
What are schemas for?

- Provide a semantic insight into data
- Capture the structure of the graph (summary)
- Enable validation i.e., checking data conformance

Shape Expression Schema (ShEx)

Syntax

ShEx is a set of rules of the form $\mathit{Type} \to \mathit{RegExp}(\mathit{Predicate} \times \mathit{Type})$

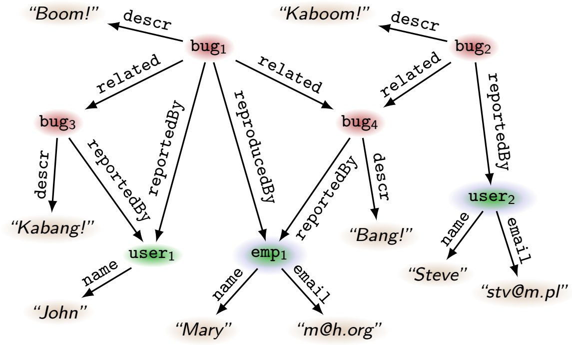


$$
\begin{aligned}
\texttt{Bug} \to{} & \texttt{descr} :: \texttt{str}, \\
& \texttt{reportedBy} :: \texttt{User}, \\
& \texttt{reproducedBy} :: \texttt{Employee}?, \\
& \texttt{related} :: \texttt{Bug}* \\
\texttt{User} \to{} & \texttt{name} :: \texttt{str}, \\
& \texttt{email} :: \texttt{str}? \\
\texttt{Employee} \to{} & \texttt{name} :: \texttt{str}, \\
& \texttt{email} :: \texttt{str}
\end{aligned}
$$

Shape Expression Schema (ShEx)

Syntax

ShEx is a set of rules of the form $\mathit{Type} \rightarrow \mathit{RegExp}(\mathit{Predicate} \times \mathit{Type})$



$$
\begin{aligned}
\mathtt{Bug} &\rightarrow \mathtt{descr} :: \mathtt{str}, \\
&\quad \mathtt{reportedBy} :: \mathtt{User}, \\
&\quad \mathtt{reproducedBy} :: \mathtt{Employee}?, \\
&\quad \mathtt{related} :: \mathtt{Bug}^* \\
\mathtt{User} &\rightarrow \mathtt{name} :: \mathtt{str}, \\
&\quad \mathtt{email} :: \mathtt{str}? \\
\mathtt{Employee} &\rightarrow \mathtt{name} :: \mathtt{str}, \\
&\quad \mathtt{email} :: \mathtt{str}
\end{aligned}
$$

Semantics

Graph satisfies a schema if every node has at least one type

Background information

Shape Expressions Schemas (ShEx)

- Inspired by XML Schema and reminiscent of (tree) automata
- Based on regular expressions under commutative closure
  membership NP-c [Kopczynski&To'10]; containment coNEXP-c [Haase&Hofman'16]
- Envisioned as a potential XSLT-like transformation engine for RDF

ShEx vs SHACL

- ShEx is a schema language with a growing base of users and a host of applications
- SHACL is Shape Constraint Language (e.g., path constraints)
- significant overlap (upcoming paper) but also differences (recursion, negation etc.)
- comparable validation complexity (NP-complete)
- both have been developed under the tutelage of W3C
- SHACL ended up a W3C Recommendation (yay!), ShEx a W3C Community Group Project

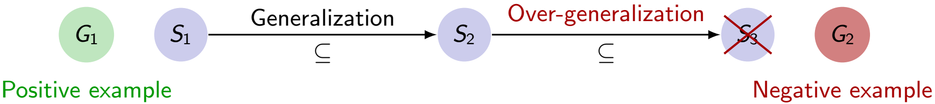
Containment problem

Containment $S_1 \subseteq S_2$

Does every graph that satisfies S_1 also satisfies S_2?

Motivation

- Fundamental problem (static analysis: query optimization, schema minimization etc.)
- Inference of ShEx (work in progress)



The challenge

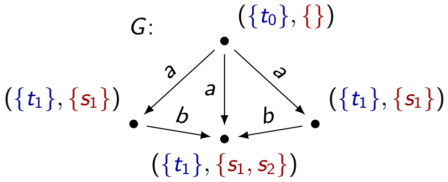
- Commutative (unordered) REs = Presburger Arithmetic (PA)
- $\mathrm{MSO_G} \subsetneq \mathrm{ShEx} \subseteq \mathrm{MSO_G} + \mathrm{PA}$
- $\mathrm{MSO_G}$ with very little arithmetic is undecidable [Elgot&Rabin'66]

Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^* \qquad t_1 \to b :: t_1^?$$

$$\not\subseteq$$

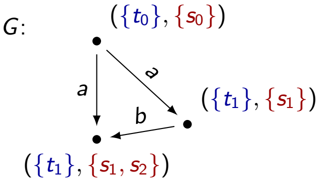
$$S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^* \qquad s_1 \to b :: s_2^? \qquad s_2 \to \epsilon$$



Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^* \qquad\qquad \not\subseteq \qquad\qquad S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^*$$

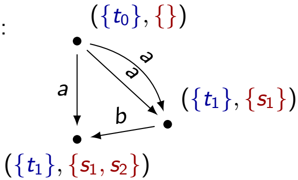
$$\qquad t_1 \to b :: t_1^? \qquad\qquad\qquad\qquad\qquad s_1 \to b :: s_2^? \qquad s_2 \to \epsilon$$



Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^* \qquad t_1 \to b :: t_1^? \qquad \not\subseteq \qquad S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^* \qquad s_1 \to b :: s_2^? \qquad s_2 \to \epsilon$$

G:

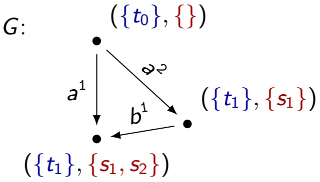


Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^{*} \qquad t_1 \to b :: t_1^{?}$$

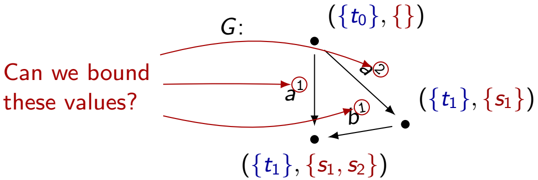
$$\not\subseteq$$

$$S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^{*} \qquad s_1 \to b :: s_2^{?} \qquad s_2 \to \epsilon$$



Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^* \qquad t_1 \to b :: t_1^? \qquad \not\subseteq \qquad S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^* \qquad s_1 \to b :: s_2^? \qquad s_2 \to \epsilon$$

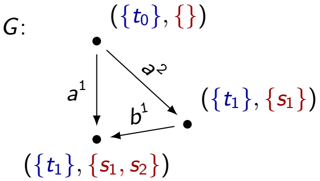


Containment of ShEx is in co2NEXP^{NP}

- The counter-example is a graph with at most exponential number of nodes, one node per (A, B)-kind
- A PA formula that describes the multiplicities
- PA enjoys an upper bound $O(|\varphi|^{3|\bar{x}|^k})$ on minimal solutions [Weispfenning'90]
- Double exponential upper bound on the size of a counter-example

Decidability of Containment

$$S_1: \quad t_0 \to a :: t_1^* \qquad t_1 \to b :: t_1^? \qquad \not\subseteq \qquad S_2: \quad s_0 \to a :: s_1 \mid (a :: s_1,\ a :: s_2)^* \qquad s_1 \to b :: s_2^? \qquad s_2 \to \epsilon$$

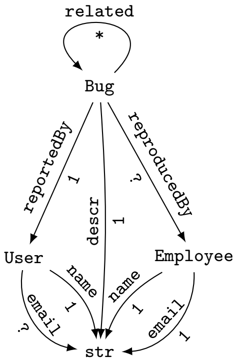


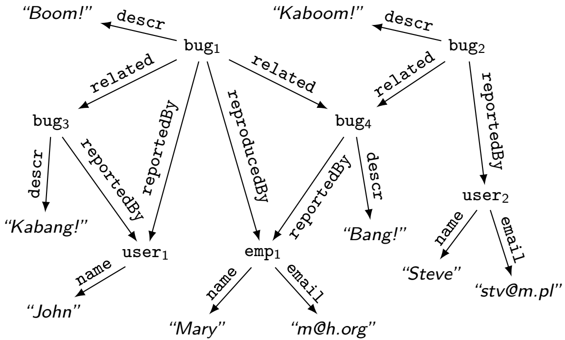
Containment of ShEx is in co2NEXP^{NP} and coNEXP-hard

- The counter-example is a graph with at most exponential number of nodes, one node per (A, B)-kind
- A PA formula that describes the multiplicities
- PA enjoys an upper bound $O(|\varphi|^{3|\bar{x}|^k})$ on minimal solutions [Weispfenning'90]
- Double exponential upper bound on the size of a counter-example
- Containment of commutative REs has recently been shown to be coNEXP-hard [Haase&Hofman'16]

ShEx$_0$

- no disjunction $(a :: t_1 \mid b :: t_2)$ and no grouping $(a :: t_1, b :: t_2)^*$
- Shape Graphs – an equivalent graphical representation





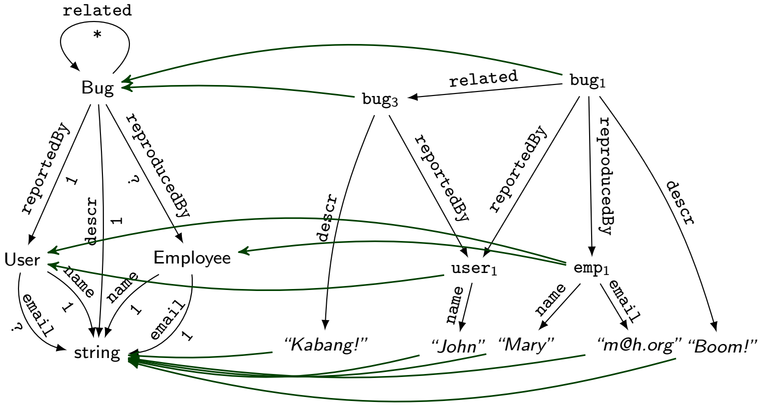
$$\texttt{Bug} \to \texttt{descr} :: \texttt{str},\ \texttt{reportedBy} :: \texttt{User},\ \texttt{reproducedBy} :: \texttt{Employee}^?,\ \texttt{related} :: \texttt{Bug}^*$$

$$\texttt{User} \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}^?$$

$$\texttt{Employee} \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}$$

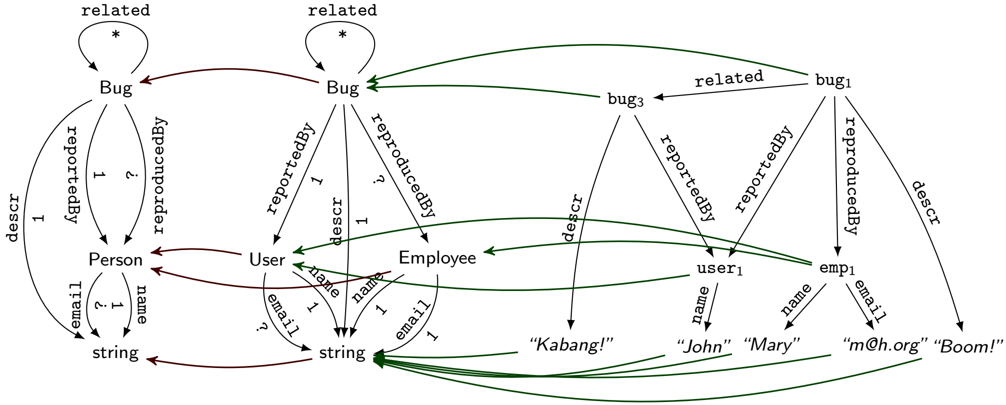
Embeddings

- Generalized simulations (graph morphism with occurrence constraints)
- Capture semantics of $ShEx_0$ by means of structural comparison



Embeddings

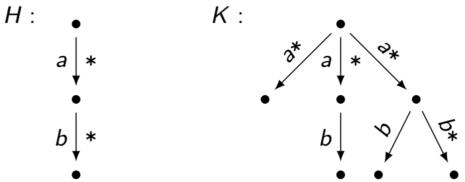
- Generalized simulations (graph morphism with occurrence constraints)
- Capture semantics of $ShEx_0$ by means of structural comparison
- Embeddings generalize naturally to pairs of shape graphs



Properties of embeddings

Embedding and containment

- Embedding implies containment
- In general, the converse does not hold



H cannot be embedded into K ($b :: t^*$ is equivalent to $\epsilon \mid b :: t \mid b :: t^+$)

Theorem

Constructing embeddings is

- in PTIME if only 1, ?, $*$, + are used
- NP-complete if arbitrary occurrence constraints are allowed $a :: t^{[n;m]}$

When does containment implies embedding ?

Determinism

- $\mathsf{DetShEx}_0$ every type uses each predicate symbol at most once
- $\mathsf{DetShEx}_0^-$ no + are allowed and ? must be dominated by $*$

Characterizing graph

For any $H \in \mathsf{DetShEx}_0^-$ there is a polynomially-sized graph G characterizing H under containment i.e.,

$$\forall K \in \mathsf{DetShEx}_0^-.\ G \text{ satisfies } K \Rightarrow H \subseteq K.$$

Theorem

Containment for $\mathsf{DetShEx}_0^-$ is in PTIME

Theorem

Containment for $\mathsf{DetShEx}_0$ is coNP-hard

Two equivalent ShEx$_0$ schemas and their shape graphs

H :

$\texttt{Bug} \to \texttt{descr} :: \texttt{str},\ \texttt{reportedBy} :: \texttt{User},\ \texttt{reproducedBy} :: \texttt{Employee}^?,$
$\quad \texttt{related} :: \texttt{Bug}^*$

$\texttt{User} \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}^?$

$\texttt{Employee} \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}$

K :

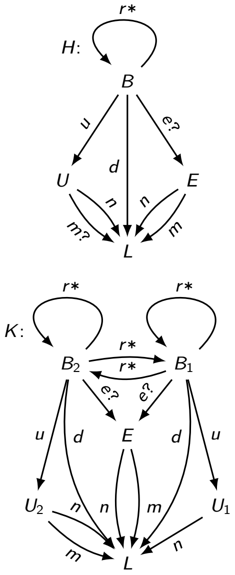
$\texttt{User}_1 \to \texttt{name} :: \texttt{str}$

$\texttt{User}_2 \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}$

$\texttt{Bug}_1 \to \texttt{descr} :: \texttt{str},\ \texttt{reportedBy} :: \texttt{User}_1,\ \texttt{reproducedBy} :: \texttt{Employee}^?,$
$\quad \texttt{related} :: \texttt{Bug}_1^*,\ \texttt{related} :: \texttt{Bug}_2^*$

$\texttt{Bug}_2 \to \texttt{descr} :: \texttt{str},\ \texttt{reportedBy} :: \texttt{User}_2,\ \texttt{reproducedBy} :: \texttt{Employee}^?,$
$\quad \texttt{related} :: \texttt{Bug}_1^*,\ \texttt{related} :: \texttt{Bug}_2^*$

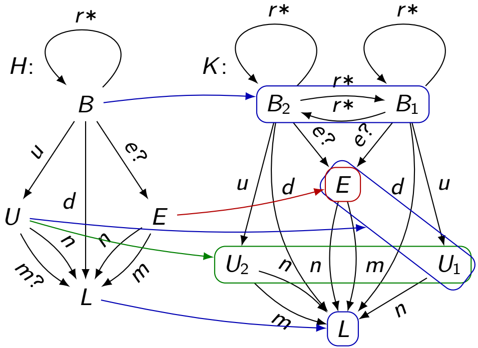
$\texttt{Employee} \to \texttt{name} :: \texttt{str},\ \texttt{email} :: \texttt{str}$



Coverings

Generalization of embeddings

A type t is covered by a set of types $S = \{s_1, \dots, s_k\}$ iff any node satisfying t also satisfies one of the types in S

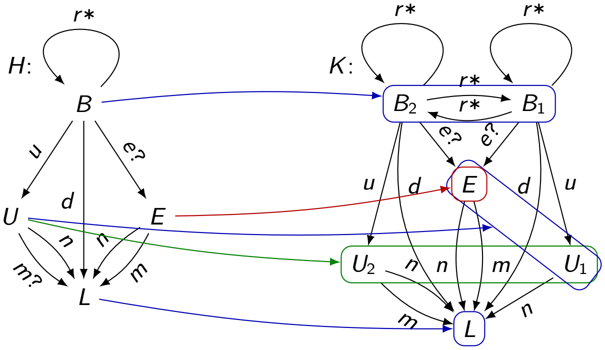


Lemma (Constructing covering)

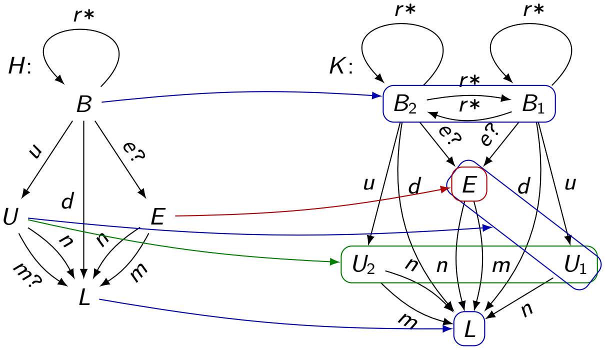
Covering is the maximum relation $R \subseteq \text{Types}(H) \times \mathcal{P}(\text{Types}(K))$ such that

$$\forall (t, S) \in R.\ \text{def}(t) \xrightarrow{\mathit{Unfold}}_R \{\text{def}(s) \mid s \in S\}.$$

Unfolding



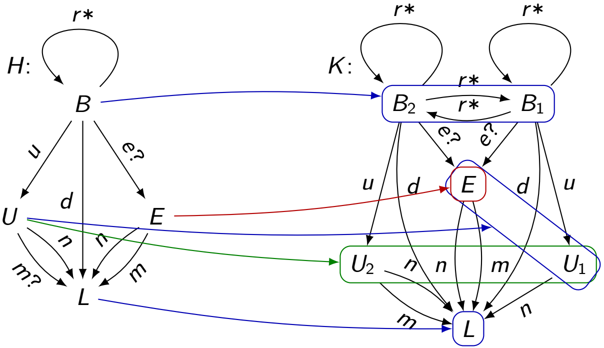
Unfolding



Unfolding U into $\{U_1, U_2\}$

$$U \to n :: L,\ m :: L^? \equiv n :: L,\ (\epsilon \mid m :: L) \equiv (n :: L) \mid (n :: L,\ m :: L) \leftarrow U_1 \mid U_2$$

Unfolding



Unfolding B into $\{B_1, B_2\}$

$$
\begin{aligned}
B &\to r :: B^*,\ u :: U,\ d :: L,\ e :: E^?\\
&\equiv (r :: B^*,\ u :: U_1,\ d :: L,\ e :: E^?) \mid (r :: B^*,\ u :: U_2,\ d :: L,\ e :: E^?)\\
&\equiv (r :: B_1^*,\ r :: B_2^*,\ u :: U_1,\ d :: L,\ e :: E^?) \mid (r :: B_1^*,\ r :: B_2^*,\ u :: U_2,\ d :: L,\ e :: E^?)\\
&\leftarrow B_1 \mid B_2
\end{aligned}
$$

Complexity of ShEx_0

Theorem

Containment for ShEx_0 is in EXP

- Covering is a relation of exponential size
- Covering can be obtained with an iterative refinement process (starting with maximal relation and remove at least one element at each iteration until stabilization)
- At each step unfoldings are constructed and each unfolding is a tree whose size is bounded exponentially

Theorem

Containment for ShEx_0 is EXP-complete

- Reduction from containment for binary tree automata

Conclusions and future work

Summary of results

- Containment for ShEx is decidable
- There is a (arguably practical) class DetShEx_0^- with tractable containment
- ShEx is very different from tree automata and requires novel techniques

ShEx	DetShEx	ShEx_0	DetShEx_0	DetShEx_0^-
coNEXP-h and co2EXP^{NP}	co2EXP	EXP-c	coNP-h	PTIME

Further work

- Since ShEx_0 still can capture (limited) disjunction, can the lower bounds be adapted to ShEx_0 with disjunction?
- How many of our results transfer to SHACL and at what cost?
- What is the precise impact of determinism on complexity of containment?

Questions