Static analysis over tree-structured data using graph decompositions

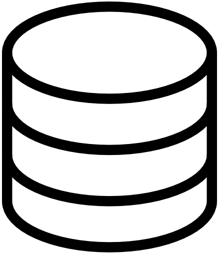
Filip Murlak
University of Warsaw, Poland

Contains joint work with Mikołaj Bojańczyk, Wojciech Czerwiński, Claire David, Filip Mazowiecki, Pawel Parys, and Adam Witkowski.

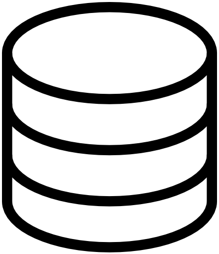
ALCOP 2017
Glasgow, Scotland

Problems

Data



Data

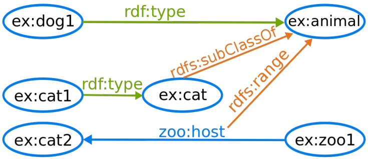


Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182

Activities Table

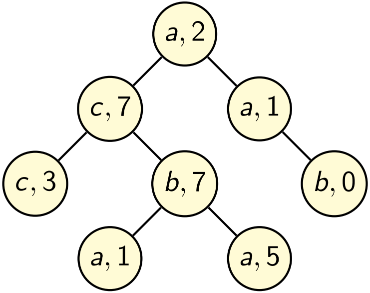
ID*	Activity*	Cost
084	Swimming	$17
084	Tennis	$36
100	Squash	$40
100	Swimming	$17
182	Tennis	$36




```
<xypair>
    <xaxis axistype="independent">
        <property>Wavelength</property>
        <value>254.0</value>
        <unit>nm</unit>
    </xaxis>
    <yaxis axistype="dependent">
        <property>Absorbance</property>
        <value>0.1234</value>
    </yaxis>
</xypair>
```

Data trees

```
<xypair>
    <xaxis axistype="independent">
        <property>Wavelength</property>
        <value>254.0</value>
        <unit>nm</unit>
    </xaxis>
    <yaxis axistype="dependent">
        <property>Absorbance</property>
        <value>0.1234</value>
    </yaxis>
</xypair>
```



trees finite, unranked, ordered

labels $a, b, c, \ldots$ from a finite alphabet (tags)

data values $0, 1, 2, \ldots$ from an infinite data domain (contents)

Schemas describe allowed shapes of data trees

```
<xs:element name="brightstar">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="magnitude" type="xs:decimal"/>
      <xs:element name="distance" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Define several types of trees, each specified (recursively) by

- the label of the root,
- possible sequences of immediate subtree types (regexp);

and choose some of the types as allowed.

Schemas describe allowed shapes of data trees

```
<xs:element name="brightstar">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="magnitude" type="xs:decimal"/>
      <xs:element name="distance" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Define several types of trees, each specified (recursively) by

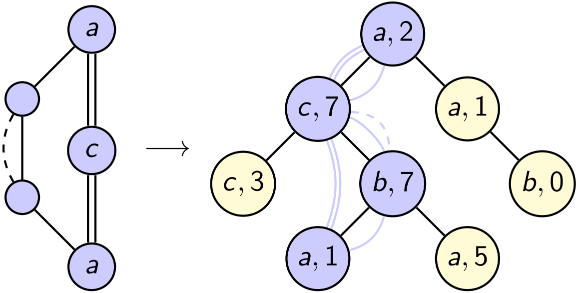
- the label of the root,
- possible sequences of immediate subtree types (regexp);

and choose some of the types as allowed.

Example: *a*-only path from root to leaf, *b*'s elsewhere

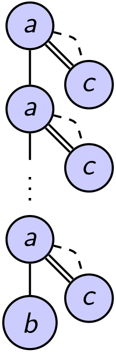
- type τ: root label *a*, immediate subtree types $\sigma^*\tau\sigma^* + \epsilon$;
- type σ: root label *b*, immediate subtree types σ^* ;
- choose: τ .

Conjunctive queries over data trees



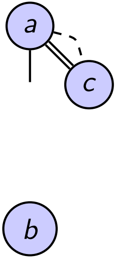
$$\exists x_1 \cdots \exists x_5$$
$$\begin{aligned} child(x_1, x_2) &\wedge child(x_2, x_3) \wedge child(x_3, x_4) \wedge \\ &\wedge desc(x_1, x_5) \wedge desc(x_5, x_4) \wedge \\ &\wedge a(x_1) \wedge a(x_4) \wedge c(x_5) \wedge \\ &\wedge x_2 \sim x_3 \end{aligned}$$

Datalog on data trees



$$p(x) \leftarrow a(x) \land desc(x,y) \land c(y) \land x \sim y \land child(x,z) \land p(z)$$

$$p(x) \leftarrow b(x)$$



extensional predicates *child*, *desc*, $\sim$, *a*, *b*, *c*, ...;

intensional predicates defined recursively using conjunctive queries;

monadic only unary intensional predicates;

linear at most one intensional atom per rule.

Static analysis problems

Satisfiability: Is query P (CQ, UCQ, Datalog, FO, etc.) satisfied in some data tree (conforming to given schema)?

Equivalence: Are queries P, Q equivalent on all data trees?

Containment: Does P imply Q on all data trees?

The staple of data management: query optimization, consistency tests, evaluation modulo constraints, constraint entailment, ...

By Trakhtenbrot's theorem, all undecidable for FO queries.

Static analysis problems

Satisfiability: Is query P (CQ, UCQ, Datalog, FO, etc.) satisfied in some data tree (conforming to given schema)?

Equivalence: Are queries P, Q equivalent on all data trees?

Containment: Does P imply Q on all data trees?

The staple of data management: query optimization, consistency tests, evaluation modulo constraints, constraint entailment, ...

By Trakhtenbrot's theorem, all undecidable for FO queries.

P sat	iff	not $P \Leftrightarrow \bot$	iff	not $P \Rightarrow \bot$
$P \wedge \neg Q$, $Q \wedge \neg P$ unsat	iff	$P \Leftrightarrow Q$	iff	$P \Rightarrow Q$, $Q \Rightarrow P$
$P \wedge \neg Q$ unsat	iff	$P \Leftrightarrow P \wedge Q$	iff	$P \Rightarrow Q$

Problems

Old solutions

New solution

More problems with solutions

Some problems without solutions

Containment of CQs over arbitrary structures

[Chandra, Merlin '77]

Def: $Q \in \mathrm{CQ}$ $\rightsquigarrow$ $\mathbb{A}_Q$: universe $\mathrm{Var}Q$, relations given by atoms of Q

Fact: $\mathbb{A} \models Q$ iff exists $h\colon \mathbb{A}_Q \to \mathbb{A}$

Thm: $P \Rightarrow Q$ iff exists $g\colon \mathbb{A}_Q \to \mathbb{A}_P$

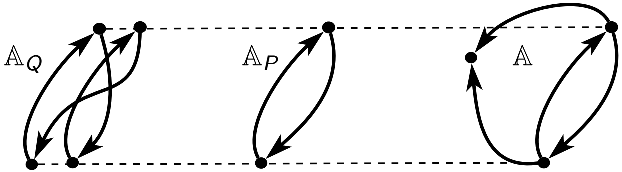
Containment of CQs over arbitrary structures

[Chandra, Merlin '77]

Def: $Q \in \mathrm{CQ}$ $\rightsquigarrow$ $\mathbb{A}_Q$: universe $\mathrm{Var}Q$,
relations given by atoms of Q

Fact: $\mathbb{A} \models Q$ iff exists $h\colon \mathbb{A}_Q \to \mathbb{A}$

Thm: $P \Rightarrow Q$ iff exists $g\colon \mathbb{A}_Q \to \mathbb{A}_P$



($\Leftarrow$) If $g\colon \mathbb{A}_Q \to \mathbb{A}_P$ and $h\colon \mathbb{A}_P \to \mathbb{A}$, then $h \circ g\colon \mathbb{A}_Q \to \mathbb{A}$.

($\Rightarrow$) $\mathbb{A}_P \models P$ and $P \Rightarrow Q$, so $\mathbb{A}_P \models Q$. Exists $h : \mathbb{A}_Q \to \mathbb{A}_P$.

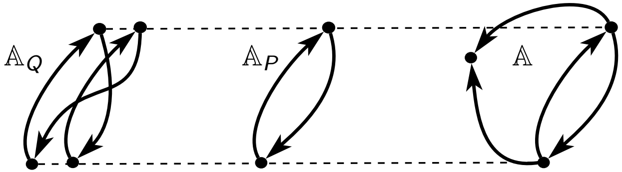
Containment of CQs over arbitrary structures

[Chandra, Merlin '77]

Def: $Q \in \mathrm{CQ}$ $\rightsquigarrow$ $\mathbb{A}_Q$: universe $\mathrm{Var}Q$, relations given by atoms of Q

Fact: $\mathbb{A} \models Q$ iff exists $h\colon \mathbb{A}_Q \to \mathbb{A}$

Thm: $P \Rightarrow Q$ iff exists $g\colon \mathbb{A}_Q \to \mathbb{A}_P$



($\Leftarrow$) If $g\colon \mathbb{A}_Q \to \mathbb{A}_P$ and $h\colon \mathbb{A}_P \to \mathbb{A}$, then $h \circ g\colon \mathbb{A}_Q \to \mathbb{A}$.

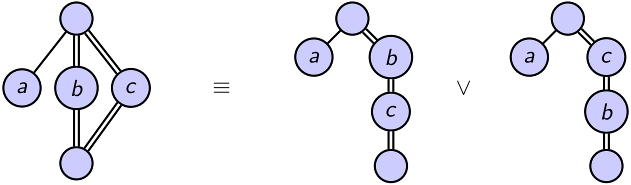
($\Rightarrow$) $\mathbb{A}_P \models P$ and $P \Rightarrow Q$, so $\mathbb{A}_P \models Q$. Exists $h : \mathbb{A}_Q \to \mathbb{A}_P$.

To decide containment, test existence of a homomorphism.

Containment for UCQs over trees without data

[Miklau, Suciu '04]

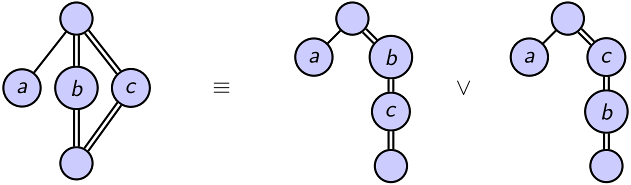
Each UCQ is equivalent to a union of tree-shaped CQs:



Containment for UCQs over trees without data

[Miklau, Suciu '04]

Each UCQ is equivalent to a union of tree-shaped CQs:



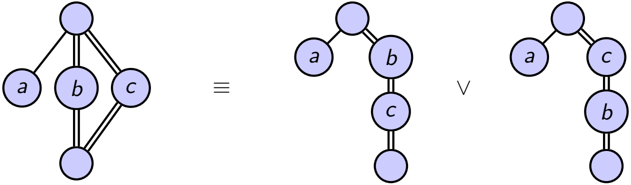
For a tree shaped CQ π build an equivalent tree automaton:

- it computes bottom-up the set of matched subtrees of π;
- knowing which subtrees of π match at the children of node v or strictly below, one can tell which match at v or strictly below.

Containment for UCQs over trees without data

[Miklau, Suciu '04]

Each UCQ is equivalent to a union of tree-shaped CQs:



For a tree shaped CQ π build an equivalent tree automaton:

- it computes bottom-up the set of matched subtrees of π;
- knowing which subtrees of π match at the children of node v or strictly below, one can tell which match at v or strictly below.

Tree automata are effectively closed under Boolean combinations.

Test emptiness of the automaton corresponding to $P \wedge \neg Q$.

Containment for UCQs over data trees

[Björklund, Martens, Schwentick '08]

Can restrict to trees with data values $c_1, \ldots, c_{\|P\|}$ and distinct nulls.

- Let T be a tree satisfying P and not Q.
- P touches $\leq \|P\|$ data values in T; replace with $c_1, \ldots, c_{\|P\|}$.
- In each node not touched by P put a unique fresh data value.
- The resulting tree T' still satisfies P and not Q.

Containment for UCQs over data trees

[Björklund, Martens, Schwentick '08]

Can restrict to trees with data values $c_1, \ldots, c_{\|P\|}$ and distinct nulls.

- Let T be a tree satisfying P and not Q.
- P touches $\leq \|P\|$ data values in T; replace with $c_1, \ldots, c_{\|P\|}$.
- In each node not touched by P put a unique fresh data value.
- The resulting tree T' still satisfies P and not Q.

In such trees, $x \sim y$ holds iff either $x = y$ or $x \sim c_i$ and $y \sim c_i$.

By considering all possibilities, replace P, Q with P', Q' using only $x = y$, $x \sim c_i$, $y \sim c_i$.

Check containment over the finite alphabet $\Sigma \times \{\bot, c_1, \ldots, c_n\}$.

Equivalence for Datalog

Equivalence for Datalog is undecidable:

- with descendant [Abiteboul, Bourhis, Muscholl, Wu 2013]
- for non-linear programs [Mazowiecki, Murlak, Witkowski 2014]
- for non-monadic programs (descendant is easily simulated).

Equivalence for Datalog

Equivalence for Datalog is undecidable:

- with descendant [Abiteboul, Bourhis, Muscholl, Wu 2013]
- for non-linear programs [Mazowiecki, Murlak, Witkowski 2014]
- for non-monadic programs (descendant is easily simulated).

Theorem (Mazowiecki, Murlak, Witkowski 2014)

Equivalence for linear monadic Datalog without desc is decidable.

Can't we restrict reused datavalues like before?

Equivalence for Datalog

Equivalence for Datalog is undecidable:

- with descendant [Abiteboul, Bourhis, Muscholl, Wu 2013]
- for non-linear programs [Mazowiecki, Murlak, Witkowski 2014]
- for non-monadic programs (descendant is easily simulated).

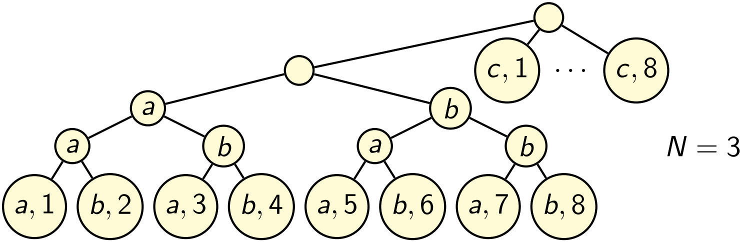
Theorem (Mazowiecki, Murlak, Witkowski 2014)

Equivalence for linear monadic Datalog without desc is decidable.

Can't we restrict reused datavalues like before?

- Let T be a tree satisfying P and not Q.
- Then T satisfies some CQ P_0, an unravelling of P.
- P_0 touches $\leq \|P_0\|$ data values in T, like before,
- but $\|P_0\|$ can be arbitrarily large...

Example



$$
\begin{aligned}
P &\leftarrow DOWN_0(x)\\
DOWN_i(x) &\leftarrow child(x,y) \wedge a(y) \wedge DOWN_{i+1}(y)\\
DOWN_N(x) &\leftarrow UP_N(x) \wedge (N{+}1)\text{-}parent(x,y) \wedge child(y,z) \wedge c(z) \wedge x \sim z\\
UP_i(x) &\leftarrow a(x) \wedge parent(x,y) \wedge child(y,z) \wedge b(z) \wedge DOWN_i(z)\\
UP_i(x) &\leftarrow b(x) \wedge parent(x,y) \wedge UP_{i-1}(y)\\
UP_0(x) &\leftarrow true\\
Q &\leftarrow x \sim y \wedge i\text{-}parent(x,x') \wedge i\text{-}parent(y,y') \wedge a(x') \wedge b(y')
\end{aligned}
$$

Problems

Old solutions

New solution

More problems with solutions

Some problems without solutions

Clique-width

Instead of processing structures, process their hierarchical decompositions (derivations).

Construct (derive) coloured structures using operations:

i – create a new node of colour i;

$R(i_1, \dots, i_r)$ – add to R all tuples of nodes with colours $(i_1, \dots, i_r)$;

$i \mapsto j$ – change colour i to j;

$\oplus$ – take disjoint union of two structures.

clique-width$(\mathbb{A})$ = least number of colours sufficient to construct $\mathbb{A}$

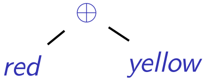
Examples

Linear orders: clique-width 2

yellow

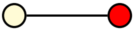
Examples

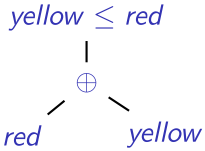
Linear orders: clique-width 2



Examples

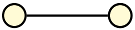
Linear orders: clique-width 2

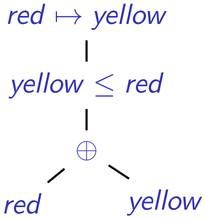




Examples

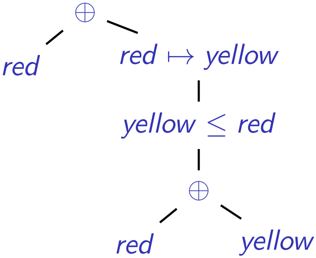
Linear orders: clique-width 2





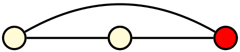
Examples

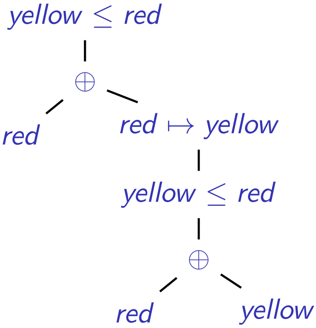
Linear orders: clique-width 2



Examples

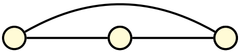
Linear orders: clique-width 2

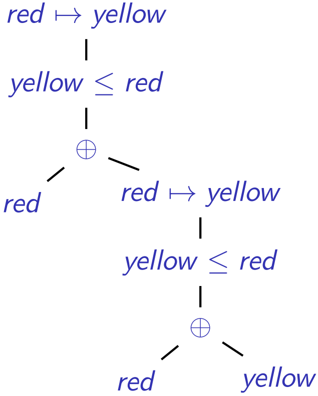




Examples

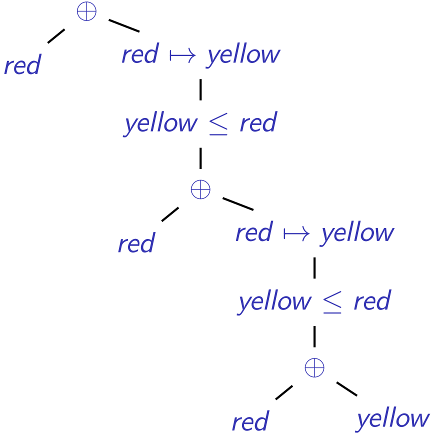
Linear orders: clique-width 2





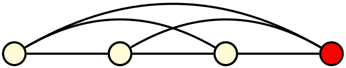
Examples

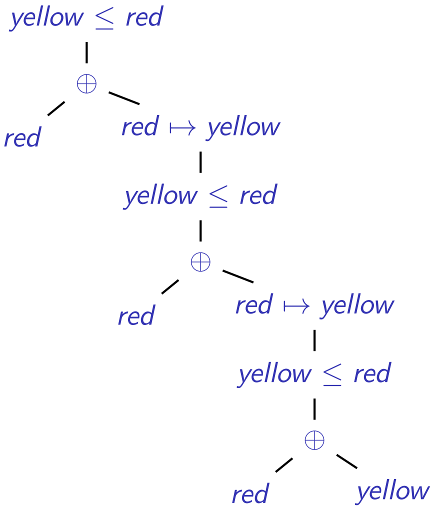
Linear orders: clique-width 2



Examples

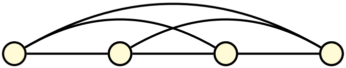
Linear orders: clique-width 2

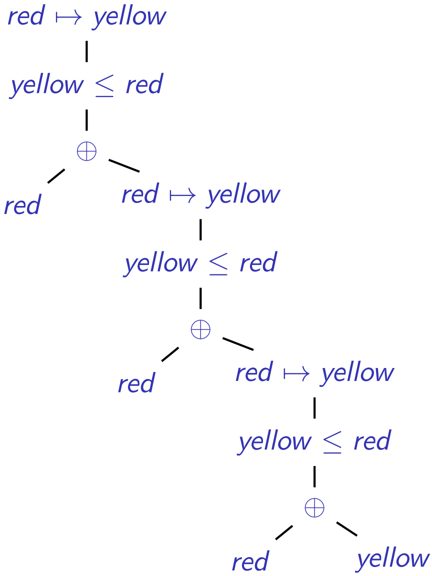




Examples

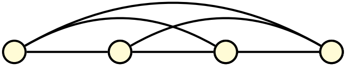
Linear orders: clique-width 2



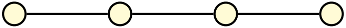


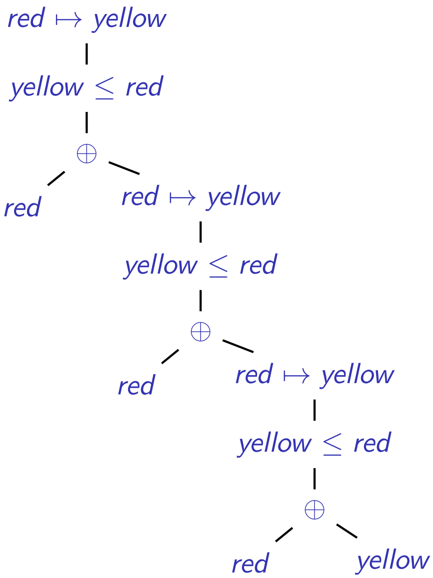
Examples

Linear orders: clique-width 2



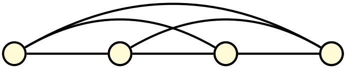
Paths: clique-width 3



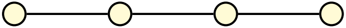


Examples

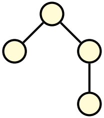
Linear orders: clique-width 2

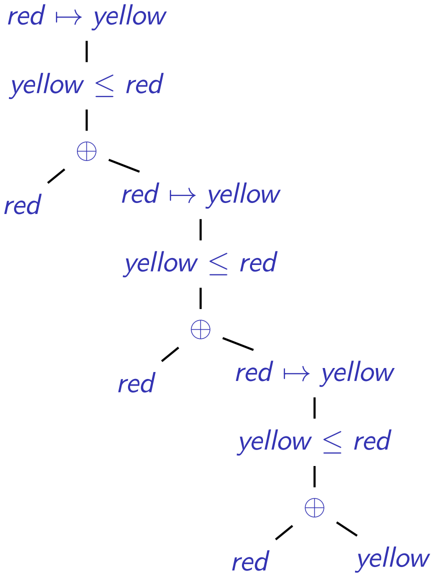


Paths: clique-width 3



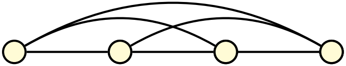
Trees: clique-width 3

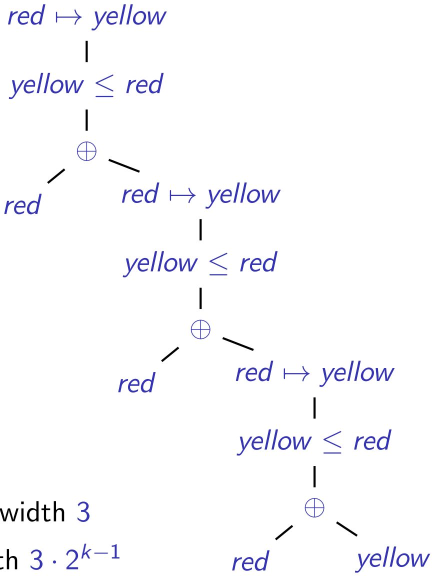




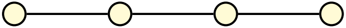
Examples

Linear orders: clique-width 2

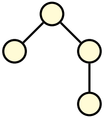




Paths: clique-width 3



Trees: clique-width 3



Cographs: clique-width 2

Distance-hereditary graphs: clique-width 3

Graphs of tree-width k: clique-width $3 \cdot 2^{k-1}$

Bounded clique-width means simple

Many NP-complete problems are in P for graphs of bounded clique-width.

Fixed-parameter tractable with clique-width as parameter: time $f(k) \cdot n^c$ on inputs of size n and clique-width at most k, where f is some function, and c is an absolute constant.

Hamiltonicity

Is there a path in graph G that visits each node exactly once?

3-colorability

Can nodes of the graph G be coloured so that each edge connects nodes of different colours?

Courcelle's theorem

Monadic second order logic (MSO)

$$\varphi, \psi \ ::= \ R(x_1, \ldots, x_r) \ \mid \neg\varphi \ \mid \ \varphi \wedge \psi \ \mid \ \varphi \vee \psi \ \mid \ \exists x\, \varphi \ \mid \ \forall x\, \varphi \ \mid \ \mid \exists X\, \varphi \ \mid \ \forall X\, \varphi \ \mid \ X(x)$$

3-colorability

$$\exists X_1\, \exists X_2\, \exists X_3 \quad \forall x\, X_1(x) \vee X_2(x) \vee X_3(x) \wedge \forall x\, \forall y\, E(x,y) \Rightarrow \bigwedge_i \neg(X_i(x) \wedge X_i(y))$$

Theorem (Courcelle)

For every $k \in \mathbb{N}$ and $\varphi \in MSO$ one can construct an automaton recognizing k-derivations yielding models of φ.

Courcelle's theorem applied to parametrized complexity

Theorem (Courcelle)

For every $k \in \mathbb{N}$ and $\varphi \in MSO$ one can construct an automaton recognizing k-derivations yielding models of φ.

Corollary

Each set of structures definable in MSO can be decided in polynomial time over graphs of bounded cliquewidth.

- Compute k-derivation e for the input structure (poly-time);
- construct the automaton $\mathcal{A}$ for k and the defining formula φ;
- run the automaton $\mathcal{A}$ on e.

Courcelle's theorem applied to static analysis

Theorem (Courcelle)

For every $k \in \mathbb{N}$ *and* $\varphi \in MSO$ *one can construct an automaton recognizing* k*-derivations yielding models of* φ*.*

Corollary

For every $k \in \mathbb{N}$*, it is decidable if given* $\varphi \in MSO$ *has a model of clique-width at most* k*.*

- Construct the automaton $\mathcal{A}$ for k and the formula φ;
- test emptiness of the automaton $\mathcal{A}$ (poly-time).

Datalog containment via bounded clique-width

[Bojańczyk, Murlak, Witkowski '15]

Theorem

Let P, Q be monadic, linear Datalog programs without descendant. If $P \wedge \neg Q$ is satisfiable, it is satisfiable in a data tree of clique-width at most $10 \cdot \|P\|^2$.

Corollary

Containment for linear monadic Datalog programs without descendant is decidable.

- Rewrite monadic programs P, Q into $\varphi_P, \varphi_Q \in MSO$.
- Write $\varphi_{datatree} \in MSO$ saying that the structure is a data tree.
- Test satisfiability of $\varphi_P \wedge \neg\varphi_Q \wedge \varphi_{datatree}$.
- For tight complexity, adjust Courcelle's theorem to Datalog.

Problems

Old solutions

New solution

More problems with solutions

Some problems without solutions

Containment for downward Datalog

[Bojańczyk, Murlak, Witkowski '15]

A monadic Datalog program is downward if in all rules for $S(x)$, all mentioned nodes are descendants of x.

Theorem

Let P, Q be downward Datalog programs. If $P \wedge \neg Q$ is satisfiable, it is satisfiable in a data tree of clique-width at most $5 \cdot \|P\|$.

Corollary

Containment for downward Datalog programs is decidable.

Non-mixing constraints

[Czerwiński, David, Murlak, Parys '16]

In database systems, correctness of data is expressed with integrity constraints:

$$\varphi(\bar{x}) \Rightarrow \alpha_{\sim}(\bar{x}) \quad \text{and} \quad \varphi(\bar{x}) \Rightarrow \alpha_{\nsim}(\bar{x})$$

with $\varphi \in \mathit{UCQ}(\mathit{child}, \mathit{desc}, \Sigma)$, $\alpha_{\sim} \in \mathit{UCQ}(\sim)$, $\alpha_{\nsim} \in \mathit{UCQ}(\nsim)$.

Validity: Does each data tree of schema $\mathcal{S}$ satisfy set Δ of non-mixing constraints?

Entailment: Does each data tree of schema $\mathcal{S}$ that sastisfies Δ also satisfies constraint δ?

Theorem

Both problems allow counter-examples of bounded clique-width.

Problems

Old solutions

New solution

More problems with solutions

Some problems without solutions

Open problems

Containment of Datalog programs

- in the presence of a schema;
- with sibling order.

Non-mixing constraints with

- free use of comparisons with constants;
- Skolem functions.