Tutorial: Recent Advances in Datalog±

Joint work with Andrea Calì, Thomas Lukasiewicz, Michael Kifer et al.

Georg Gottlob, Michael Morak & Andreas Pieris

Univ. of Oxford, UK and TU Wien, Austria
Structure of the Tutorial

PART 1, presented by Georg Gottlob

- Datalog (basics)
- Motivation for Datalog±, applications
- Ontological reasoning and query answering (QA)
- Main decidable fragments: guarded Datalog±, sticky Datalog±
- Complexity and expressive power
- Weakly guarded Datalog±
- Other extensions (quick overview)
- Further Applications (SPARQL querying, conceptual modeling)
PART 2, presented by Michael Morak

- Introduction/quick recap

- Other features of Datalog± (Overview)

- **Feature: Disjunction**
  * What is disjunction? Why do we need it?
  * Semantics of Disjunction (disj. chase, QA with disjunction, etc.)
  * Complexity of guarded Datalog± with Disjunction

- **Feature: Negation**
  * What is negation? Why do we need it?
  * Different semantics (classical, stratified, stable)
  * Why not simply classical FO negation?
  * Complexity of guarded Datalog+- with Stable Negation

- Combining Disjunction and Negation?

- Conclusion
What is Datalog± and What are its Advantages?

- A rule-based formalism that combines the advantages of LP in Datalog with features for expressing ontological knowledge
- A general KR language
What is $Datalog^\pm$ and What are its Advantages?

- A rule-based formalism that combines the advantages of LP in Datalog with features for expressing ontological knowledge
- A general KR language

But first, let say few words about the good old plain Datalog
Datalog

- Recursive database query language defined in the 1980s
- A simple but useful framework for inductive definitions
- Datalog = Function-free Prolog with fully declarative semantics
- Simple A*(Horn+) syntax and clear semantics
- Well-understood
- Complexity: EXPTIME-complete; PTIME-complete data complexity
- Various methods for optimisation
Datalog: Another Success Story of LiCS

- Important recursive query language
- Benchmark for other query languages
- Has influenced SQL3 standard
- Successfully used in many recent applications, e.g., code querying, web data extraction, business process, modeling and automation…
- Large projects and some companies are “Datalog-based”
Datalog: Another Success Story of LiCS

• Important recursive query language

• Benchmark for other query languages

• Has influenced SQL3 standard

• Successfully used in many recent applications, e.g., code querying, web data extraction, business process, modeling and automation…

• Large projects and some companies are “Datalog-based”

RENEWED INTEREST

Datalog 2.0 workshop
Datalog - An Example

**EDB**
- emp(john)
- emp(mary)
- emp(hilda)
- emp(ann)
- reports(john, ann)
- reports(mary, ann)
- reports(ann, hilda)
- reports(hilda, eve)

**IDB**
- reports(X, Y) → emp(X)
- reports(X, Y) → emp(Y)
- emp(X) → person(X)
- reports(X, Y) → mgs(Y, X)
- mgs(X, Y), mgs(Y, Z) → mgs(X, Z)

emp(eve), mgs(ann, john), mgs(ann, mary), mgs(hilda, ann)
mgs(eve, hilda), mgs(hilda, john), mgs(hilda, mary),
mgs(eve, ann), mgs(eve, john), mgs(eve, mary)
Datalog - An Example

Unique Name Assumption implicitly made, i.e., john ≠ mary etc.
Datalog Semantics

\[ \forall X \forall Y \text{ reports}(X,Y) \rightarrow \text{emp}(X) \land \]
\[ \forall X \forall Y \text{ reports}(X,Y) \rightarrow \text{emp}(Y) \land \]
\[ \forall X \text{ emp}(X) \rightarrow \text{person}(X) \land \]
\[ \forall X \forall Y \text{ reports}(X,Y) \rightarrow \text{mgs}(Y,X) \land \]
\[ \forall X \forall Y \forall Z \text{ mgs}(X,Y), \text{mgs}(Y,Z) \rightarrow \text{mgs}(X,Z) \]

\[ \models \]

\[ \text{emp}(eve), \text{mgs}(ann,john), \text{mgs}(ann,mary), \text{mgs}(hilda,ann) \]
\[ \text{mgs}(eve,hilda), \text{mgs}(hilda,john), \text{mhs}(hilda,mary), \]
\[ \text{mgs}(eve,ann), \text{mgs}(eve,john), \text{mgs}(eve,mary) \]

\[ = \text{LFP} \]
Datalog and Conjunctive Query Answering

\[ Q_1 : \exists X \exists Y \exists Z \ (\text{emp}(X) \land \text{mgs}(X,Y) \land \text{mgs}(Y,Z)) \quad \text{Yes!} \]

\[ Q_2 : \exists X \exists Y \exists Z \ (\text{mgs}(X,Y) \land \text{mgs}(Y,X)) \quad \text{No!} \]
Datalog and CTL

**Theorem:** Datalog with stratified negation captures PTIME over ordered structures

\[ E(\neg P \cup Q) \]

Q(X) \rightarrow H(X)

not P(X), E(X,Y), H(Y) \rightarrow H(X)

Q \land AFP

not P(X), E(X,Y) \rightarrow T(X,Y)

T(X,Y), not P(Y) E(Y,Z) \rightarrow T(X,Z)

T(X,X) \rightarrow S(X)

not P(X), E(X,Y), S(Y) \rightarrow S(X)

Q(X), not S(X) \rightarrow H(X)

Kripke structure, no sink

there is a path from X to Y on which “not P” globally holds

X is on an infinite path on which “not P” globally holds
Datalog LITE: A deductive query language with linear time model checking

GEORG GOTTLOB
Vienna University of Technology
ERICH GRÄDEL
RWTH Aachen
and
HELMUT VEITH
Vienna University of Technology

We present Datalog LITE, a new deductive query language with a linear time model checking algorithm, i.e., linear time data complexity and program complexity. Datalog LITE is a variant of Datalog that uses stratified negation, restricted variable occurrences and a limited form of universal quantification in rule bodies.

Despite linear time evaluation, Datalog LITE is highly expressive: It encompasses popular modal and temporal logics such as CTL or the alternation-free $\mu$-calculus. In fact these formalisms have natural presentations as fragments of Datalog LITE. Further Datalog LITE is equivalent to the alternation-free portion of guarded fixed point logic. Consequently, linear time model checking algorithms for all mentioned logics are obtained in a unified way.

The results are complemented by inexpressibility proofs to the effect that linear time fragments of stratified Datalog have too limited expressive power.
The Goal of the Datalog± Project

Extend plain Datalog with key modelling features:

- existential quantification in rule heads
- constraints of the form “body → X = Y”
- constraints of the form “body → ⊥”

for query answering over databases...

while possibly maintaining the good data complexity of Datalog
Why?

- Data exchange
- Data extraction
- Ontology querying (DL-Lite, EL, etc)
- Automated product configuration
- Querying the semantic web (RDF graphs)
- Conceptual Modeling (e.g., UML)
Why?

- Data exchange
- Data extraction
- Ontology querying (DL-Lite, EL, etc)
- Automated product configuration
- Querying the semantic web (RDF graphs)
- Conceptual Modeling (e.g., UML)
Data Exchange

employee(Lastname, Firstname, Address) → person(Firstname, Lastname, Birthdate)

employee(X,Y,Z) → ∃W person(Y,X,W)
Data Exchange

- Deep theory has been developed
  [Fagin, Kolaitis, Miller & Popa, 2003]; [Arenas et al., 2014]

- One looks for a finite model $\rightarrow$ null values
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>PRICE</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
we need object creation...

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
### Data Extraction

- `table(T_1)`
- `table(T_2)`
- `sameColor(T_1, T_2)`
- `isNeighbourRight(T_1, T_2) → ∃T tablebox(T), contains(T, T_1), contains(T, T_2)`

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba_Protege_cx</td>
<td>480</td>
</tr>
<tr>
<td>Dell_25416</td>
<td>360</td>
</tr>
<tr>
<td>Dell_23233</td>
<td>470</td>
</tr>
<tr>
<td>Acer_78987</td>
<td>390</td>
</tr>
</tbody>
</table>
## Ontology Querying

**DL-Lite family** of description logics

<table>
<thead>
<tr>
<th>DL-Lite TBox</th>
<th>Datalog± Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DL-Lite\textsubscript{core}</strong></td>
<td></td>
</tr>
<tr>
<td>$c \sqsubseteq d$</td>
<td>$c(X) \rightarrow d(X)$</td>
</tr>
<tr>
<td>$c \sqsubseteq \exists p$</td>
<td>$c(X) \rightarrow \exists Y \ p(X,Y)$</td>
</tr>
<tr>
<td>$\exists p \sqsubseteq c$</td>
<td>$p(X,Y) \rightarrow c(X)$</td>
</tr>
<tr>
<td>$c \sqsubseteq \neg d$</td>
<td>$c(X), \ d(X) \rightarrow \bot$</td>
</tr>
<tr>
<td><strong>DL-Lite\textsubscript{R} (OWL 2 QL)</strong></td>
<td></td>
</tr>
<tr>
<td>$p \sqsubseteq q$</td>
<td>$p(X,Y) \rightarrow q(X,Y)$</td>
</tr>
<tr>
<td><strong>DL-Lite\textsubscript{F}</strong></td>
<td></td>
</tr>
<tr>
<td>$\text{funct}(p)$</td>
<td>$p(X,Y), \ p(X,Z) \rightarrow Y=Z$</td>
</tr>
</tbody>
</table>
**Ontology Querying**

ELHI$^-$ - an extension of the key description logic EL

<table>
<thead>
<tr>
<th>ELHI$^-$ TBox</th>
<th>Datalog$^{\pm}$ Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b \sqsubseteq c \sqsubseteq d$</td>
<td>$b(X), c(X) \rightarrow d(X)$</td>
</tr>
<tr>
<td>$c \sqsubseteq \exists p.d$</td>
<td>$c(X) \rightarrow \exists Y \ p(X,Y), d(Y)$</td>
</tr>
<tr>
<td>$\exists p.c \sqsubseteq d$</td>
<td>$p(X,Y), c(Y) \rightarrow d(X)$</td>
</tr>
<tr>
<td>$c \sqsubseteq \neg d$</td>
<td>$c(X), d(X) \rightarrow \bot$</td>
</tr>
<tr>
<td>$p \sqsubseteq q$</td>
<td>$p(X,Y) \rightarrow q(X,Y)$</td>
</tr>
<tr>
<td>$p \sqsubseteq \neg q$</td>
<td>$p(X,Y), q(X,Y) \rightarrow \bot$</td>
</tr>
</tbody>
</table>
Every component of type $\alpha$ must be connected via port 1 to a component of type $\beta$

component($X$), type($X,\alpha$) $\rightarrow$ $\exists Y$ component($Y$), type($Y,\beta$), connected($X,Y,1$)
Which is the Main Reasoning Service in Datalog\(^\pm\)?
Which is the Main Reasoning Service in Datalog\(^\pm\)?

Query answering under certain-answer semantics

- Undecidable problem

- Expressive decidable fragments - field of intense research
  (e.g., IBM Almaden, Montpellier, Dresden, Calabria, Oxford, Rome …)
Which are the Main Decidable Datalog$^\pm$ Languages?

- **Weak-acyclicity**: acyclicity condition on the dependency graph
  [Fagin et al., 2003]

- Introduced in the context of data exchange

- Guarantees the termination of the **chase** procedure

- Several interesting extensions
  [Marnette, 2009]; [Cuenca Grau et al., 2013] etc.
Which are the Main Decidable Datalog\(\pm\) Languages?

- **Guardedness**: there exists a body-atom that contains all the \(\forall\)-variables
  \[
  \text{supervisorOf}(S,E), \text{emp}(E) \rightarrow \text{emp}(S)
  \]

- **Weak-guardedness**: guard only variables that may unify with null values (more details later)

- **Linearity**: there exists only one atom in the body
  \[
  \text{person}(P) \rightarrow \exists F \text{ fatherOf}(F,P) \\
  \text{fatherOf}(F,P) \rightarrow \text{person}(F)
  \]

[Calì, G. & Kifer, 2008]
Which is the main tool for proving decidability and complexity results for weakly acyclic and guarded Datalog\(^\pm\) Languages?

The Chase
The Chase: Example

\[ D = \{ p(a,b), p(c,b), p(a,d), q(a), q(d) \} \quad \text{dom}(D) = \{a,b,c,d\} \]

\[ \Sigma = \{ p(x,y) \land q(y) \rightarrow \exists z \ r(y,z), \quad r(x_1,y) \land r(x_2,z) \rightarrow y = z \} \]

\[ \text{Chase}(D, \Sigma) = ? \]
\( D = \{ p(a,b), p(c,b), p(a,d), q(b), q(d) \} \)

\( \text{dom}(D) = \{a,b,c,d\} \)

\( \Sigma = \{ p(x,y) \land q(y) \rightarrow \exists z\ r(y,z), \quad r(x_1,y) \land r(x_2,z) \rightarrow y = z \} \)

\( \text{Chase}(D, \Sigma) = D \cup \{ r(b,u) \} \)
\[ D = \{ p(a,b), p(c,b), p(a,d), q(b), q(d) \} \]
\[ \text{dom}(D) = \{ a, b, c, d \} \]
\[ \Sigma = \{ p(x,y) \& q(y) \rightarrow \exists z \ r(y,z), \quad r(x_1,y) \& r(x_2,z) \rightarrow y = z \} \]

\[ \text{Chase}(D, \Sigma) = D \cup \{ r(b,u) \} \]
\[ \Sigma = \{ \text{p}(x,y) \land \text{q}(y) \rightarrow \exists z \text{ r}(y,z), \quad \text{r}(x_1,y) \land \text{r}(x_2,z) \rightarrow y = z \} \]

\[
\text{Chase}(D, \Sigma) = D \cup \{ \text{r}(b,u), \text{r}(d,v) \}
\]

\[
D = \{ \text{p}(a,b), \text{p}(c,b), \text{p}(a,d), \text{q}(b), \text{q}(d) \}, \quad \text{dom}(D) = \{a,b,c,d\}
\]
\[ \Sigma = \{ \ p(x, y) \land q(y) \implies \exists z \ r(y, z), \quad r(x_1, y) \land r(x_2, z) \implies y = z \} \]

\[ \text{Chase}(D, \Sigma) = D \cup \{ \ r(b, u), r(d, v) \} \]

\[ D = \{ p(a, b), p(c, b), p(a, d), q(b), q(d) \} \]

\[ \text{dom}(D) = \{ a, b, c, d \} \]
\[ \Sigma = \{ \quad p(x,y) \& q(y) \rightarrow \exists z \ r(y,z), \quad r(x_1,y) \& r(x_2,z) \rightarrow y = z \quad \} \]

\[ \text{Chase}(D, \Sigma) = D \cup \{ r(b,u), r(d,u) \} \]

No further rule applicable. STOP.
Facts about the Chase

• Depends on the order of rule application

\[ D = \{ p(a) \} \quad \Sigma = \{ p(x) \rightarrow \exists y \ q(y), \quad p(x) \rightarrow q(x) \} \]

\[ \text{Solution1} = \{ p(a), \ q(u), \ q(a) \} \]
Facts about the Chase

• Depends on the order of rule application

\[ D = \{ p(a) \} \quad \Sigma = \{ p(x) \rightarrow \exists x \ q(x), \quad p(x) \rightarrow q(x) \} \]

Solution1 = \{ p(a), q(u), q(a) \}

Solution2 = \{ p(a), q(a) \}

We will assume a canonical ordering
Facts about the Chase

- Can be infinite

\[ D = \{ p(a,b) \} \quad \Sigma = \{ p(x,y) \rightarrow \exists z \ p(y,z) \} \]

Solution =
\{p(a,b),
p(b,u_1),
p(u_1,u_2),
p(u_2,u_3),
p(u_3,u_4),
p(u_3,u_4),
\ldots.
\ldots.
\ldots.\}
Chase(D, Σ) is a universal model

For each other model M, there is a homomorphism Chase(D,Σ) → M
For each other model $M$, there is a homomorphism $\text{Chase}(D, \Sigma) \rightarrow M$. 

$\text{Chase}(D, \Sigma)$ is a universal model.
For each other model $M$, there is a homomorphism $\text{Chase}(D, \Sigma) \rightarrow M$.

$\text{Chase}(D, \Sigma)$ is a universal model (initial model)

$C = \text{Chase}(D, \Sigma)$

For each other model $M$, there is a homomorphism $\text{Chase}(D, \Sigma) \rightarrow M$
For each other model $M$, there is a homomorphism $\text{Chase}(D, \Sigma) \rightarrow M$.
In particular, different chases are homomorphically equivalent

This extends unique LFP property from Datalog to Datalog[∃,=]
But now: unique up to “bi-homomorphisms”
In particular, different chases are homomorphically equivalent.

But what if such chases are infinite???
How can we answer queries ???
Data Complexity  (program & query fixed)

**Theorem:** Conjunctive query answering with Datalog[∃;G] is PTIME complete.
**Theorem:** Conjunctive query answering with Datalog[$\exists;G$] is PTIME complete.

**Proof sketch:**

[Diagram showing a tree-like structure connecting to a database (DB) and various labels such as h(Q), t(Q), P(x,y,a), P(u,y,b), and an ALOGSPACE Procedure box.]
**Theorem:** Conjunctive query answering with Datalog[∃;G] is PTIME complete.

**Proof sketch (membership):**
Sketch of Alternative Proof

**Theorem:** Guarded Datalog[∃] is PTIME-complete in data complexity

Proof (in PTIME):

- Guarded Datalog[∃] enjoys the bounded guard-depth property

- Construct in PTIME the finite part $C$ of the guarded chase forest

- Evaluate the given query over $C$

[Cali, Gottlob & Lukasiewicz, Journal of Web Semantics 2012]
Bounded Guard-Depth Property

- Chase graph

\[ D = \{ R(a,b), S(b) \} \]

\[ P = \begin{cases} 
\rho_1 = R(X,Y), S(Y) \rightarrow \exists Z R(Z,X) \\
\rho_2 = R(X,Y) \rightarrow S(X) 
\end{cases} \]
Bounded Guard-Depth Property

- Guarded chase forest

\[
R(a,b) \quad S(b) \\
\downarrow \\
R(z_1,a) \quad S(a) \\
\downarrow \\
R(z_2,z_1) \quad S(z_1) \\
\downarrow \\
R(z_3,z_2) \quad S(z_2)
\]

resticted to guards and their children

\[
R(a,b) \quad S(b) \quad 0 \\
\downarrow \\
R(z_1,a) \quad S(a) \quad 1 \\
\downarrow \\
R(z_2,z_1) \quad S(z_1) \quad 2 \\
\downarrow \\
R(z_3,z_2) \quad S(z_2) \quad 3
\]
Bounded Guard-Depth Property

guarded chase forest of $D$ w.r.t. $P$

$P(D) \models Q \Rightarrow C \models Q$
Which are the Main Decidable Datalog\(^\pm\) Languages?

- **Stickiness:** join-variables *stick* to the inferred atoms

\[
\begin{align*}
q(X, Y), &\quad p(Y, Z) \rightarrow \exists W \ t(X, Y, W) \\
t(X, Y, Z) &\rightarrow \exists W \ s(Y, W) \\
t(X, Y, Z) &\rightarrow \exists W \ s(X, W)
\end{align*}
\]

[Calì, G. & Pieris, 2010]
Stickiness: Further examples

\[ r(X,Y,Y,Z) \rightarrow s(X,Z) \quad \text{OK} \quad (not \ \text{really \ sticky \ but \ ‘sticky-join’}) \]

\[ m(X) \& e(Y) \rightarrow s(X,Y) \quad \text{OK} \quad \text{Cartesian Product [Rudolph]} \]

\[ r(U,V), \ r(V',W) \rightarrow a(U,V,V',W) \quad \text{OK} \]

\[ r(X,Y) \& r(Y,Z) \rightarrow r(X,Z) \quad \text{NOT OK} \]

\[ r(U,V), \ r(V',W) \rightarrow a(U,V,V',W) \quad \text{NOT OK} \]

\[ a(X,Y,Y,Z) \rightarrow r(X,Z) \quad \text{NOT OK} \]
Extensions of Stickiness (just a glimpse)

• Sticky-join Datalog$^\pm$: mild extension; $p(x,x) \rightarrow q(x)$

• Tame Datalog$^\pm$: combines guarded and sticky

• Weakly sticky: lose join-variables that can host finitely many terms

• Weakly tame: ‘weak’ version of tame
Which are the Main Decidable Datalog± Languages?

- Weakly-guarded
- Guarded
- Linear
  - \( DL-\text{Lite}_R \)
  - \( ELHI \)
- Sticky
- Weakly-Sticky
Which are the Main Decidable Datalog$^{\pm}$ Languages?

- Weakly-guarded
- Guarded
- Linear
- DL-Lite$_R$
- ELHI
- Weakly-Sticky
- Sticky
- BTS - Bounded Treewidth Sets
- FUS - Finite Unification Sets

Two abstract decidability paradigms

[Baget et al., 2009]
**Which are the Main Decidable \( \text{Datalog}^{\pm} \) Languages?**

interesting extensions of (weakly-)guarded \( \text{Datalog}^{\pm} \)

- **Frontier-guarded**: guard only the frontier-variables
  
  \[\text{supervisorOf}(S,E), \text{directorOf}(E,D) \rightarrow \exists F \ \text{directorOf}(S,F)\]

- **Weakly-frontier-guarded**: defined analogously

- A set of weakly-frontier-guarded rules is a BTS

[Baget et al., 2009]
**Which are the Main Decidable Datalog\(\pm\) Languages?**

**combining decidability paradigms**

- **Glut-guardedness**: guard only variables that can host infinitely many terms
- **Weak-stickiness**: loose join-variables that can host finitely many terms
- **Tameness**: sticky rules do not interact with guard-atoms
What About Equality and the Falsum in Rule Heads?

\[ \text{reportsTo}(X,Y), \text{reportsTo}(X,Z) \rightarrow Y = Z \]

- **Non-Conflicting condition**: no interaction between \( \exists \)- and \( = \)-rules
- Preliminary check **without** adding complexity

\[ \text{employee}(X), \text{customer}(X) \rightarrow \bot \]

- Check **without** adding complexity - reduction to query answering
### What is the Complexity of the Main Languages?

<table>
<thead>
<tr>
<th>Language</th>
<th>Combined Complexity</th>
<th>Fixed Program</th>
<th>Data Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>PSPACE-c</td>
<td>NP-c</td>
<td>in AC&lt;sub&gt;0&lt;/sub&gt;</td>
</tr>
<tr>
<td>Guarded</td>
<td>2EXPTIME-c</td>
<td>NP-c</td>
<td>PTIME-c</td>
</tr>
<tr>
<td>Weakly-guarded</td>
<td>2EXPTIME-c</td>
<td>EXPTIME-c</td>
<td>EXPTIME-c</td>
</tr>
<tr>
<td>Sticky</td>
<td>EXPTIME-c</td>
<td>NP-c</td>
<td>in AC&lt;sub&gt;0&lt;/sub&gt;</td>
</tr>
<tr>
<td>Weakly-sticky</td>
<td>2EXPTIME-c</td>
<td>NP-c</td>
<td>PTIME-c</td>
</tr>
<tr>
<td>Language</td>
<td>Combined Complexity</td>
<td>Fixed Program</td>
<td>Data Complexity</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Linear</td>
<td>PSPACE-c</td>
<td>NP-c</td>
<td>in AC&lt;sub&gt;0&lt;/sub&gt;</td>
</tr>
<tr>
<td>Guarded</td>
<td>2EXPTIME-c</td>
<td>NP-c</td>
<td>PTIME-c</td>
</tr>
<tr>
<td>Weakly-guarded</td>
<td>2EXPTIME-c</td>
<td>EXPTIME-c</td>
<td>EXPTIME-c</td>
</tr>
<tr>
<td>Sticky</td>
<td>EXPTIME-c</td>
<td>NP-c</td>
<td>in AC&lt;sub&gt;0&lt;/sub&gt;</td>
</tr>
<tr>
<td>Weakly-sticky</td>
<td>2EXPTIME-c</td>
<td>NP-c</td>
<td>PTIME-c</td>
</tr>
</tbody>
</table>
Weakly-guarded Datalog$^\pm$

All body-variables occurring at affected positions only, jointly occur in an atom only unifies with $\exists$-positions or affected positions

\[ p(X,Y), s(Y,Z) \rightarrow \exists W \; t(Y,X,W) \]
\[ t(X,Y,Z) \rightarrow \exists W \; p(W,Z) \]
\[ p(X,Y) \rightarrow \exists Z \; q(X,Z) \]

Affected positions $=$ ?
Weakly-guarded Datalog$^\pm$

All body-variables occurring at **affected positions** only, jointly occur in an atom

only unifies with $\exists$-positions or affected positions

\[
p(X,Y), s(Y,Z) \rightarrow \exists W \ t(Y,X,W) \\
t(X,Y,Z) \rightarrow \exists W \ p(W,Z) \\
p(X,Y) \rightarrow \exists Z \ q(X,Z)
\]

Affected positions = \{ t[3], p[1], q[2] \}
Weakly-guarded Datalog\(^\pm\)

All body-variables occurring at **affected positions** only, jointly occur in an atom

only unifies with \(\exists\)-positions or affected positions

\[
p(X,Y), \ s(Y,Z) \rightarrow \exists W \ t(Y,X,W)
\]

\[
t(X,Y,Z) \rightarrow \exists W \ p(W,Z)
\]

\[
p(X,Y) \rightarrow \exists Z \ q(X,Z)
\]

Affected positions \(= \{ t[3], p[1], q[2]\} \)
Weakly-guarded Datalog$^\pm$

All body-variables occurring at affected positions only, jointly occur in an atom

only unifies with $\exists$-positions or affected positions

\[
p(X,Y), s(Y,Z) \rightarrow \exists W \ t(Y,X,W)
\]
\[
t(X,Y,Z) \rightarrow \exists W \ p(W,Z)
\]
\[
p(X,Y) \rightarrow \exists Z \ q(X,Z)
\]

Affected positions = \{ t[3], p[1], q[2], t[2], p[2], q[1] \}
Weakly-guarded Datalog$^\pm$

All body-variables occurring at affected positions only, jointly occur in an atom only unifies with $\exists$-positions or affected positions

\[
p(X,Y), s(Y,Z) \rightarrow \exists W \ t(Y,X,W)
\]

\[
t(X,Y,Z) \rightarrow \exists W \ p(W,Z)
\]

\[
p(X,Y) \rightarrow \exists Z \ q(X,Z)
\]

Affected positions = \{ t[3], p[1], q[2], t[2], p[2], q[1] \}
Weakly-guarded Datalog$^\pm$

All body-variables occurring at affected positions only, jointly occur in an atom only unifies with $\exists$-positions or affected positions

\[ p(X,Y), s(Y,Z) \rightarrow \exists W \ t(Y,X,W) \]

\[ t(X,Y,Z) \rightarrow \exists W \ p(W,Z) \]

\[ p(X,Y) \rightarrow \exists Z \ q(X,Z) \]

Affected positions  =  {  t[3], p[1], q[2], t[2], p[2], q[1]  }
Expressive Power of Weakly-guarded Datalog\(^{\pm}\)

- **Theorem:** Semipositive weakly-guarded Datalog\(^{\pm}\) captures EXPTIME on ordered databases (but not on arbitrary databases)

- **Theorem:** Weakly-guarded Datalog\(^{\pm}\) with stratified negation captures EXPTIME even w/o an order

[G., Šimkus & Rudolph, 2014]
Expressive Power of Weakly-guarded Datalog$^\pm$

Stratified WG Datalog$^\pm = \text{EXPTIME}$

Guarded

Datalog

the right consolidation of guardedness with plain Datalog
Expressive Power of Weakly-guarded Datalog$^\perp$

Some natural but inherently difficult queries can be expressed, e.g., whether a graph contains a clique of size $n$

The database encodes the graph and the value $n$; $n$ is encoded as a successor-chain of length $n$

\[
\begin{align*}
\text{zero}(X) & \rightarrow \exists Y \text{ism}(Y,X) \\
\text{ism}(X,Y), \text{succ}(Y,Z), \text{node}(W) & \rightarrow \exists U \text{next}(X,W,U), \text{ism}(U,Z), \text{map}(U,Z,W) \\
\text{next}(X,Y,Z), \text{map}(X,U,V) & \rightarrow \text{map}(Z,U,V) \\
\text{less}(X,Y), \text{map}(Z,X,W), \text{map}(Z,Y,U), \text{not edge}(W,U) & \rightarrow \text{noclique}(Z) \\
\text{less}(X,Y), \text{map}(Z,X,W), \text{map}(Z,Y,W) & \rightarrow \text{noclique}(Z) \\
\text{ism}(X,Y), \text{max}(Y), \text{not noclique}(X) & \rightarrow \text{clique}
\end{align*}
\]

[Arenas, G. & Pieris, 2014]
Expressive Power of Weakly-guarded Datalog

Some natural but inherently difficult queries can be expressed, e.g., whether a graph contains a clique of size \( n \):

\[
\begin{align*}
\text{zero}(X) &\rightarrow \exists Y \text{ ism}(Y,X) \\
\text{ism}(X,Y), \text{succ}(Y,Z), \text{node}(W) &\rightarrow \exists U \text{ next}(X,W,U), \text{ism}(U,Z), \text{map}(U,Z,W) \\
\text{next}(X,Y,Z), \text{map}(X,U,V) &\rightarrow \text{map}(Z,U,V) \\
\text{less}(X,Y), \text{map}(Z,X,W), \text{map}(Z,Y,U), \text{not edge}(W,U) &\rightarrow \text{noclique}(Z) \\
\text{less}(X,Y), \text{map}(Z,X,W), \text{map}(Z,Y,W) &\rightarrow \text{noclique}(Z) \\
\text{ism}(X,Y), \text{max}(Y), \text{not noclique}(X) &\rightarrow \text{clique}
\end{align*}
\]
Does Existential Quantification Give Us More Power?

- Guarded and Sticky Datalog\(\pm\) are Datalog rewritable, i.e., the program and the query can be compiled into a Datalog query.

- Thus, we do not add power by allowing existential quantification.

- However, in most applications we’d like to decouple the actual query from the program (or ontology).

- Refined notion of expressive power - program expressive power
  [Arenas, G. & Pieris, 2014]
Program Expressive Power (PEP)

• For a fixed Datalog$^\pm$ program $P$, we define

$$\text{PEP}(P) = \{ (D,Q) \mid D \land P \models Q \}$$

• For a Datalog$^\pm$ language $L$, we define

$$\text{PEP}(L) = \{ \text{PEP}(P) \mid P \in L \}$$

• PEP - it expresses the sets of queries that can be made true over all databases via programs in $L$
**Program Expressive Power (PEP)**

**Theorem:** There exists a Linear Datalog\(^\pm\) program \(P\) such that, there is no plain Datalog program \(P_{DAT}\) such that \(\text{PEP}(P) = \text{PEP}(P_{DAT})\)

**Proof:**

Let \(D = \{\text{zero}(0), \text{one}(1)\}\)

Let \(P = \begin{cases} 
\text{zero}(X) \rightarrow \text{n}(X) \\
\text{n}(X) \rightarrow \exists Y \text{s}(X,Y), \text{n}(Y) \\
\text{zero}(X) \rightarrow \text{bit}(X) \\
\text{one}(X) \rightarrow \text{bit}(X) 
\end{cases}\)

Let \(Q_1 = \exists X \exists Y \exists Z \text{s}(X,Y) \land \text{s}(Y,Z)\) \quad \& \quad Q_2 = \exists X \exists Y \exists Z \text{s}(X,Y) \land \text{s}(Y,Z) \land \text{bit}(Z)\)

For every Datalog program \(R\), \(D \land R \vdash Q_1 \Rightarrow D \land R \vdash Q_2\)

But, \(D \land P \not\vdash Q_1\) while \(Q_2\) is not entailed
Program Expressive Power (PEP)

- \( \text{PEP} \text{(Guarded)} \supset \text{PEP} \text{(Guarded Datalog)} \)
- \( \text{PEP} \text{(Weakly-guarded)} \supset \text{PEP} \text{(Datalog)} \)
- \( \text{PEP} \text{(Sticky)} \supset \text{PEP} \text{(Sticky Datalog)} \)
- \( \text{PEP} \text{(Weakly-sticky)} \supset \text{PEP} \text{(Datalog)} \)
What Other Features are Allowed in Datalog$^\pm$?

- **Negation** under the well-founded and stable-model semantics

- **Disjunctive** rule heads under FO semantics

- **Disjunctive rules + negation** under SMS - ongoing work

→ Second part of Tutorial
What Other Developments Have Been Made?

- **Query rewriting** has been extensively studied
  [Calvanese et al., 2005]; [G., Orsi & Pieris, 2011]; [G. & Schwentick, 2012] [Retc.

- **Probabilistic extension** of Datalog$^\pm$ based on Markov logic networks
  [Gottlob, Lukasiewicz & Simari, 2011]

- **Inconsistency management** in Datalog$^\pm$
  [Lukasiewicz, Martinez & Simari, 2012]

- **Preference-based query answering** in Datalog$^\pm$
  [Lukasiewicz, Martinez & Simari, 2013]

- **Finite query answering** under Datalog$^\pm$ languages
What are Typical Applications of Datalog\textsuperscript{±}? 

- Data exchange
- Data extraction
- Ontology querying (DL-Lite, EL, etc)
- Automated product configuration
- Querying the semantic web (RDF graphs)
- Conceptual Modeling (e.g., UML)
What are Typical Applications of Datalog⁺?

- Data exchange
- Data extraction
- Ontology querying (DL-Lite, EL, etc)
- Automated product configuration
- Querying the semantic web (RDF graphs)
- Conceptual Modeling (e.g., UML)
Querying the Semantic Web

- **RDF** - data model for representing information in the Web

- … in fact, is a finite set of triples \((subject, predicate, object)\) - or a relational database for the schema \{\text{triple}(.,.,..)\}

- **SPARQL** - the standard language for querying RDF data
Some SPARQL Queries

- \( Q = (\?X, \text{name}, \?Y) \) - list of pairs \((o_1, o_2)\) such as \(o_2\) is the name of \(o_1\)

- \( Q = (\?X, \text{name}, B) \) - list of elements that have a name

- \( Q = (\?X, \text{name}, \?Y) \ \text{OPT} \ (\?X, \text{phone}, \?Y) \) - for every subject \(o\), return \(o\), the name of \(o\), and the phone number of \(o\), if the phone number is available; otherwise, return \(o\) and its name
From SPARQL to Datalog

$Q = (?X, \text{name, } ?Y)$ - list of pairs $(o_1, o_2)$ such as $o_2$ is the name of $o_1$

$\text{triple}(X, \text{name, } Y) \rightarrow Q(X, Y)$
Q = (?X, name, B) - list of elements that have a name

triple(X,name,Y) → Q(X)
From SPARQL to Datalog

\[ P = (?X, \text{name}, ?Y) \quad \text{OPT} \quad (?X, \text{phone}, ?Y) \] - for every subject \( o \), return \( o \), the name of \( o \), and the phone number of \( o \), if the phone number is available; otherwise, return \( o \) and its name

list of individuals with phone number

\[ \text{triple}(X, \text{name}, Y), \text{triple}(X, \text{phone}, Z) \rightarrow Q(X, Y, Z), \text{compatible}(Z) \]

the third argument (i.e., the phone no.) is missing

\[ \text{triple}(X, \text{name}, Y), \text{not compatible}(X) \rightarrow Q_3(X, Y) \]
From SPARQL to Datalog

\[ P = (?X, \text{name}, ?Y) \ \text{OPT} \ (?X, \text{phone}, ?Y) \ - \text{for every subject} \ o, \text{return} \ o, \text{the name of} \ o, \text{and the phone number of} \ o, \text{if the phone number is available; otherwise, return} \ o \text{and its name} \]

\[ \text{triple}(X, \text{name}, Y), \text{triple}(X, \text{phone}, Z) \rightarrow P(X,Y,Z), \text{compatible}(Z) \]

\[ \text{triple}(X, \text{name}, Y), \text{not compatible}(X) \rightarrow Q_3(X,Y) \]

Non-guarded rules, and also negation is needed
From SPARQL to Datalog

\[ P = (?X, \text{name}, ?Y) \text{ OPT } (?X, \text{phone}, ?Y) \] - for every subject \( o \), return \( o \), the name of \( o \), and the phone number of \( o \), if the phone number is available; otherwise, return \( o \) and its name

\[
\text{triple}(X, \text{name}, Y), \text{triple}(X, \text{phone}, Z) \rightarrow P(X, Y, Z), \text{compatible}(Z)
\]

\[
\text{triple}(X, \text{name}, Y), \text{not compatible}(X) \rightarrow Q_3(X, Y)
\]

Non-guarded rules, and also negation is needed

\textbf{Stratified Weakly-Guarded Datalog}
Additional Functionalities

- **Reasoning capabilities** - deal with RDFS and OWL vocabularies
- **Navigational capabilities** - exploit the graph structure of RDF data
- **General form of recursion** - express natural queries

**Theorem:** Stratified weakly-guarded Datalog\(^\pm\) is strictly more expressive than SPARQL enriched with the above functionalities (under the OWL 2 QL profile)

**Note:** A tractable fragment (in data complexity) which is suitable for querying RDF graphs exists - mild syntactic restriction

[Arenas, Gottlob & Pieris, 2014], [Gottlob & Pieris, IJCAI 2015]
Company Modeling

\[
\text{Company}(X) \rightarrow \exists Y \text{ Issues}(X,Y)
\]
\[
\text{Stock}(X) \rightarrow \exists Y \text{ Issues}(Y,X)
\]
\[
\text{Stock}(X), \text{Issues}(Y,X), \text{Issues}(Z,X) \rightarrow Y = Z
\]
\[
\text{Stock}(X), \text{Index}(X,Y) \rightarrow \text{Str}(Y)
\]
\[
\text{Stock}(X), \text{getIndex}(X,Y) \rightarrow \text{List}(Y)
\]
Are There any Implemented Systems?

- **Naya**: Able to treat the FO-rewritable languages of the Datalog\(^\pm\) family, namely Linear and Sticky [Virgilio et al., 2012]

- **DLV\(^3\)**: Implements a bottom-up evaluation strategy for a Datalog\(^\pm\) language called Shy inside the well-known ASP system DLV [Leone et al., 2012]

- **Alaska**: Similarly to Nyaya, is able to treat the FO-rewritable Datalog\(^\pm\) languages [König et al., 2012]