Why Rules?

OWL may not suffice for all applications

- There are statements that cannot be expressed in OWL (cf. Lecture 4)
- Modeling constructs of OWL not always adequate or most desirable
- First-order logic in general may be insufficient (e.g. if non-monotonic negation is desired)

→ “Rules” as an alternative paradigm for knowledge modeling
What is a Rule?

- Logical Rules (predicate logic implications)
  - “F → G” (equivalent to “¬F v G”)
  - Logical extension of a knowledge base (static)
  - Open World
  - Declarative (descriptive)

- Procedural Rules (e.g. production rules)
  - “If X then Y else Z”
  - Executable machine directive (dynamic)
  - Operational (meaning = effect on execution)

- Logic Programming (e.g. Prolog, F-Logic)
  - “man(x) :- person(X), not woman(X)”
  - Approximating logical semantics with procedural aspects, built-ins possible
  - Typically Closed World
  - “semi-declarative”

- Deduction rules of a calculus (e.g. rules for RDF semantics, lecture 1)
  - Rules not part of the knowledge base, “meta-rules”
Rule languages are hardly compatible with each other → important to chose adequate rule language

Possible criteria:

- Clear specification of syntax and semantics?
- Support by software tools?
- Which expressive features are needed?
- Complexity of implementation? Performance?
- Compatibility with other formats, e.g. OWL?
- Declarative (describing) or operational (programming)?
- …
Which Rule Language?

- **Logical Rules** (predicate logic implications)
  - Clearly defined, extensively researched, well understood
  - Very well compatible with OWL and RDF
  - Not decidable if unrestricted

- **Procedural Rules** (e.g. production rules)
  - Many independent approaches, vague definition
  - Used like programming languages, relation to RDF and OWL not clear
  - Efficient processing possible

- **Logic Programming** (e.g. Prolog, F-Logic)
  - Clearly defined, but many independent approaches
  - Partly compatible with OWL and RDF
  - Decidability/complexity depends very much on the chosen approach

→ In this lecture: predicate logic rules
  (which are also the basis for logic programming)
Predicate Logic as a Rule Language

- Rules as first-order logic implications (Horn clauses):

\[ A_1 \land A_2 \land \ldots \land A_n \rightarrow H \]  

(“Body → Head”)

Example: “\( \text{Man}(x) \land \text{happilyMarriedWith}(x,y) \rightarrow \text{HappyHusband}(x) \)"

- Constants, variables, function symbols can be used; but no negation
- Quantifiers are omitted: free variables considered universally quantified

**Datalog:** rules without function symbols
  - Originally developed for deductive databases
  - Knowledge bases (“datalog programs”) are sets of function-free Horn clauses
  - Decidable
  - Efficiently implementable for large datasets (overall complexity ExpTime, i.e. “Draughts” or “Chess”)
Semantics of Datalog?

- Semantics follows from first-order logic:
  - Generally well-known
  - Compatible with other first-order fragments, e.g. description logics (→ OWL)

Note: to maintain this relationship, we will prefer FOL syntax “hasCat(x,y)” over Turtle style “x hasCat y” or “triple(x,hasCat,y).”
How can datalog and OWL be combined?

**SWRL – Semantic Web Rule Language** [swrl]

- Proposal for a rule extension for OWL (W3C member submission)
- Idea: datalog rules referring to an OWL ontology
  → Symbols in rules can be OWL identifiers or new symbols
- Various further features and syntactic forms (not relevant here)
Semantics of SWRL

Combined semantics OWL DL + datalog?
→ use first-order mapping of OWL (lecture 2)

In effect:
- OWL individuals are datalog constants
- OWL classes are unary datalog predicates
- OWL properties are binary datalog predicates

→ A first-order interpretation can at the same time be a model for an OWL ontology and a datalog program
→ Entailment over OWL+datalog (SWRL) well-defined
Example: SWRL knowledge base

For readability, we abbreviate URIs strongly here; the datalog syntax does not follow a formal specification (SWRL XML is too verbose here)

(1) \text{Vegetarian}(x) \land \text{Fishproduct}(y) \rightarrow \text{dislikes}(x, y)
(2) \text{ordered}(x, y) \land \text{dislikes}(x, y) \rightarrow \text{Unhappy}(x)
(3) \text{ordered}(x, y) \rightarrow \text{Dish}(y)
(4) \text{dislikes}(x, z) \land \text{Dish}(y) \land \text{contains}(y, z) \rightarrow \text{dislikes}(x, y)
(5) \rightarrow \text{Vegetarian}(\text{markus})
(6) \text{Happy}(x) \land \text{Unhappy}(x) \rightarrow

(7) \text{markus} \text{ rdf:type } [
   \text{rdf:type} \text{ owl:Restriction;}
   \text{owl:onProperty} \text{ ordered;}
   \text{owl:someValuesFrom} \text{ ThaiCurry ]}
(8) \text{ThaiCurry} \text{ rdfs:subClassOf } [
   \text{rdf:type} \text{ owl:Restriction;}
   \text{owl:onProperty} \text{ contains;}
   \text{owl:someValuesFrom} \text{ Fishproduct ]}
Example: SWRL knowledge base

OWL DL can be translated to first-order logic:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Vegetarian(x) \land Fishproduct(y) \rightarrow dislikes(x,y)</td>
</tr>
<tr>
<td>(2)</td>
<td>ordered(x,y) \land dislikes(x,y) \rightarrow Unhappy(x)</td>
</tr>
<tr>
<td>(3)</td>
<td>ordered(x,y) \rightarrow Dish(y)</td>
</tr>
<tr>
<td>(4)</td>
<td>dislikes(x,z) \land Dish(y) \land contains(y,z) \rightarrow dislikes(x,y)</td>
</tr>
<tr>
<td>(5)</td>
<td>\rightarrow Vegetarian(markus)</td>
</tr>
<tr>
<td>(6)</td>
<td>Happy(x) \land Unhappy(x) \rightarrow</td>
</tr>
<tr>
<td>(7)</td>
<td>\exists y. ordered(markus,y) \land ThaiCurry(y)</td>
</tr>
<tr>
<td>(8)</td>
<td>\forall x. ThaiCurry(x) \rightarrow (\exists y. contains(x,y) \land FishProduct(y))</td>
</tr>
</tbody>
</table>

\rightarrow Semantics completely defined
\rightarrow Expected conclusion: Unhappy(markus)

Note: empty rule heads correspond to “false” (rule body must never be true)
empty rule bodies correspond to “true” (rule head must always be true)
How Hard is SWRL?

- Deduction for OWL DL is NExpTime-complete
- Deduction for OWL 2 DL is N2ExpTime-complete
- Deduction in datalog is ExpTime-complete

→ How hard is deduction for SWRL?

Deduction for SWRL is undecidable
(for OWL and thus for OWL 2, even for OWL EL)
Undecidability of SWRL

**SWRL is undecidable:**
There is no algorithm that can draw *all* logical conclusions from *all* SWRL knowledge bases, even with unlimited time and resources.

**Practically possible:**
- Algorithms that draw *all* conclusions for *some* SWRL knowledge bases
- Algorithms that draw *some* conclusions from *all* SWRL knowledge bases

→ Both trivial if “some” refers to very few things
Decidable Fragments of SWRL

Which classes of SWRL knowledge bases allow for complete inference algorithms?

– All SWRL knowledge bases consisting only of OWL (2) axioms
– All SWRL knowledge bases only consisting of datalog rules
– Every fixed finite class of SWRL knowledge bases

→ Which more interesting decidable fragments exist?
  – Description Logic Rules
  – DL-safe Rules
Observation:
Some SWRL-rules can already be expressed in OWL 2.

- Identifying all such Description Logic Rules leads to a decidable fragment
- Goal: Exploit “hidden” expressivity of OWL 2
- Implementation directly by OWL 2 tools
Simple Rules in OWL 2: Examples

- Simple OWL 2 axioms correspond to rules:

  \[
  \text{Class1} \quad \text{rdfs:subClassOf} \quad \text{Class2} . \\
  \text{Property1} \quad \text{rdfs:subPropertyOf} \quad \text{Property2} .
  \]

  correspond to

  \[
  \text{Class1}(x) \rightarrow \text{Class2}(x) \\
  \text{Property1}(x,y) \rightarrow \text{Property2}(x,y) .
  \]
Simple Rules in OWL 2: Examples

Some classes can be decomposed in rules:

\[
\begin{align*}
[ & \text{owl:intersectionOf ( Happy Unhappy ) } ] \\
& \text{rdfs:subClassOf} \quad \text{owl:Nothing .} \\
[ & \text{rdf:type} \quad \text{owl:Restriction;} \\
& \text{owl:onProperty} \quad \text{livesIn;} \\
& \text{owl:someValuesFrom [ rdf:type owl:Restriction;} \\
& \text{owl:onProperty locatedIn;} \\
& \text{owl:someValuesFrom EUCountry ]} \\
& ] \text{rdfs:subClassOf EUCitizen .}
\end{align*}
\]

correspond to

\[
\begin{align*}
\text{Happy}(x) \land \text{Unhappy}(x) \rightarrow \\
\text{livesIn}(x,y) \land \text{locatedIn}(y,z) \land \text{EUCountry}(z) \rightarrow \text{EUCitizen}(x)
\end{align*}
\]
Simple Rules in OWL 2: Examples

• Property chains provide further rule-like axioms:

```owl
hasUncle owl:PropertyChainAxiom ( hasParent hasBrother ).
```

correspond to

```owl
hasParent(x,y) ∧ hasBrother(y,z) → hasUncle(x,z)
```

→ In all examples, mapping can also be inverted (expressing rules in OWL 2)
More Rules (I)

What about the following?

\[ \text{dislikes}(x,z) \land \text{Dish}(y) \land \text{contains}(y,z) \rightarrow \text{dislikes}(x,y) \]

- Rule head with two variables $\rightarrow$ not representable by subclass axiom
- Rule body contains class expression $\rightarrow$ not representable by subproperty axiom

Yet, this rule can be encoded using OWL 2!
More Rules (2)

Simpler example:

\[ \text{Man}(x) \land \text{hasChild}(x,y) \rightarrow \text{fatherOf}(x,y) \]

Idea: replace \text{Man}(x) by a property expression to encode rule as property chain

- Self can be used to transform classes to properties:
  - Auxiliary property \( P_{\text{Man}} \)
  - Auxiliary axiom:

\[
\text{Man} \text{owl:equivalentClass} [ \text{rdf:type} \text{owl:Restriction};
\text{owl:onProperty} P_{\text{Man}}; \text{owl:hasSelf} “true”^^\text{xsd:boolean} ].
\]

- “Men are exactly those things that have an \( P_{\text{Man}} \) relation to themselves.”

We can now encode the rule as follows:

\[
\text{fatherOf} \text{owl:propertyChainAxiom} ( P_{\text{Man}} \text{hasChild} ).
\]
More Rules (3)

\[ dislikes(x, z) \land Dish(y) \land contains(y, z) \rightarrow dislikes(x, y) \]

becomes

\[ Dish \text{ owl:equivalentClass } [ \text{ rdf:type owl:Restriction; } \\
\text{ owl:onProperty P}_{\text{Dish}}; \text{ owl:hasSelf "true"^^xsd:boolean } ]. \]

\[ dislikes \text{ owl:propertyChainAxiom } \\
( dislikes \text{ [owl:inverseOf contains]} \ P_{\text{Dish}} ). \]
More Rules (4)

Not so simple:

\[ \text{Vegetarian}(x) \land \text{Fishproduct}(y) \rightarrow \text{dislikes}(x, y) \]

Idea: use `owl:topObjectProperty` for linking unconnected rule body parts

- Self can be used to transform classes to properties:
  - Auxiliary properties `PVegetarian` and `PFishproduct`
  - Axioms:

\[
\begin{align*}
\text{Vegetarian} & \text{owl:equivClass} [ \\
& \text{rdf:typ} \text{owl:Restriction}; \text{owl:onProperty} \text{PVegetarian}; \\
& \text{owl:hasSelf} "true"^^\text{xsd:boolean} ]. \\
\text{Fishproduct} & \text{owl:equivClass} [ \\
& \text{rdf:typ} \text{owl:Restriction}; \text{owl:onProperty} \text{PFishproduct}; \\
& \text{owl:hasSelf} "true"^^\text{xsd:boolean} ]. \\
\text{dislikes} & \text{owl:propertyChainAxiom} \\
& ( \text{PVegetarian} \text{owl:topObjectProperty} \text{PFishproduct} ).
\end{align*}
\]
Limits of Description Logic Rules

Not all rules can be encoded like this! Example:

\[ \text{ordered}(x,y) \land \text{dislikes}(x,y) \rightarrow \text{Unhappy}(x) \]

Overview of possible transformations:

- Inverting properties
- “Rolling-up” side branches, e.g.

\[ \text{locatedIn}(x,y) \land \text{EUCountry}(y) \rightarrow \exists! \text{Unhappy}(x) \]

becomes

\[
[ \text{rdf:type} \quad \text{owl:Restriction};
\text{owl:onProperty} \quad \text{locatedIn};
\text{owl:someValuesFrom} \quad \text{EUCountry} ]
\]

- Replacing concepts by properties (using hasSelf)
- Turn property conjunctions into chains
Defining Description Logic Rules

• **Preparation: Normalise rule**
  
  – For each occurrence (!) of an individual name \( n \) in the rule:
    
    Use a fresh variable \( x \), add \([\text{owl:oneOf } n](x)\) to rule body, and replace the occurrence of \( n \) by \( x \).
  
  – Replace every atom \( P(x, x) \) by
    
    \([\text{rdf:type } \text{owl:Restriction}; \text{owl:onProperty } P; \text{owl:hasSelf } "true"^^\text{xsd:boolean}](x)\).

• **Dependency graph of a rule:** undirected graph where
  
  – Nodes = variables of a rule
  
  – Edges = property atoms of a rule (direction is ignored)

• **A SWRL rule is a Description Logic Rule if:**
  
  – All rule atoms use OWL 2 class and property names only
  
  – The normalised rule's dependency graph has no cycles
Example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>( \text{Vegetarian}(x) \land \text{Fishproduct}(y) \rightarrow \text{dislikes}(x,y) )</td>
</tr>
<tr>
<td>(2)</td>
<td>( \text{ordered}(x,y) \land \text{dislikes}(x,y) \rightarrow \text{Unhappy}(x) )</td>
</tr>
<tr>
<td>(3)</td>
<td>( \text{ordered}(x,y) \rightarrow \text{Dish}(y) )</td>
</tr>
<tr>
<td>(4)</td>
<td>( \text{dislikes}(x,z) \land \text{Dish}(y) \land \text{contains}(y,z) \rightarrow \text{dislikes}(x,y) )</td>
</tr>
<tr>
<td>(5)</td>
<td>( \rightarrow \text{Vegetarian}(\text{markus}) )</td>
</tr>
<tr>
<td>(6)</td>
<td>( \text{Happy}(x) \land \text{Unhappy}(x) \rightarrow )</td>
</tr>
</tbody>
</table>

Note: Restrictions like regularity and simplicity must still be satisfied after the translations.
Transforming DL Rules to OWL 2

**Input:** A Description Logic Rule

1. Normalise the rule.
2. For every variable $z$ in the head: if $z$ is not in body, add `owl:Thing(z)` to body.
3. For every pair of variables $x$ and $y$:
   - If $x$ is not reachable from $y$ in the dependency graph, insert `owl:topObjectProperty(x,y)` into the rule
4. The rule head must have a form $D(z)$ or $S(z,z')$.
   - For every atom $R(x,y)$ in the rule body:
     - If the path from $z$ to $y$ is shorter than the path from $z$ to $x$ in the dependency graph, then replace $R(x,y)$ with `[owl:inverseOf R](y,x)`.
5. **While** the rule body contains an atom $R(x,y)$ such that $y$ does not occur in any other binary predicate of the rule do:
   - If the body contains $n$ unary atoms $C_1(y),...,C_n(y)$ then let $E$ denote the expression `[owl:intersectionOf (C_1 ... C_n)]` and remove $C_1(y),...,C_n(y)$ from the body. Otherwise define $E$ to be `owl:Thing`.
   - Replace $R(x,y)$ by: `[rdf:type owl:Restriction; owl:onProperty R; owl:someValuesFrom E](x)`
Transforming DL Rules to OWL 2

The rule can now be expressed in OWL 2:

- If the rule head is unary, then the rule has the form \( C_1(x) \land \ldots \land C_n(x) \rightarrow D(x) \) then replace it by

\[
[\text{owl:intersectionOf}(C_1, \ldots, C_n)] \text{ rdfs:subClassOf } D
\]

- If the rule head is binary
  - For every unary atom \( C(z) \) in the rule body:
    Create a new axiom

\[
C \text{ owl:equivalentClass } [\text{ rdf:type owl:Restriction;}
\text{ owl:onProperty } R; \text{ owl:hasSelf } "true"^^xsd:boolean ].
\]

and replace \( C(z) \) by \( R(z, z) \), where \( R \) is a fresh property.

- Rule now has the form \( R_1(x, x_2) \land \ldots \land R_n(x_n, y) \rightarrow S(x, y) \). Replace it by:

\[
S \text{ owl:propertyChainAxiom ( } R_1 \ldots R_n ) .
\]
Notes on the Transformation

- Replacing a SWRL rule in a SWRL knowledge base by the resulting set of OWL 2 axioms does not affect satisfiability (introduced auxiliary symbols do not occur elsewhere, of course)
- The given algorithm is not optimised: it may produce overly complex axioms in some cases
**Observation:** Datalog is decidable since rules can be applied in only finitely many ways: variables represent only constants.

- Variables in SWRL might represent arbitrarily many inferred individuals
- Goal: Make rules “safe” by restricting possible variable assignments
- **DL-safe rules** as another decidable fragment of SWRL
DL-safe Rules: Definition

Rules now may also include non-OWL predicates:

- A **datalog atom** is an atom with a predicate symbol that does not occur as a class or property in any OWL axiom.

A SWRL rule is **DL-safe** if:

- Every variable in the rule head occurs in a datalog atom in the body.

→ Only constant symbols relevant when considering variable assignments in datalog atoms.
Enforcing DL-Safety

• Example:

\[ \text{ordered}(x,y) \land \text{dislikes}(x,y) \rightarrow \text{Unhappy}(x) \]

→ not DL-safe if \textit{ordered} or \textit{dislikes} occur in OWL axioms

• Enforcing DL-safety by restricting rules to \textbf{named} individuals:

\[ \text{ordered}(x,y) \land \text{dislikes}(x,y) \land \text{O}(x) \land \text{O}(y) \rightarrow \text{Unhappy}(x) \]

where a fact \( \rightarrow \text{O}(a) \) is added for all individuals \( a \).

→ Rule only applicable to \textbf{named} OWL individuals
DL-Safe Rules in Practice

- OWL 2 with DL-safe rules is decidable
- Naïve implementation: each rule expressible by finitely many DL rules where all variables are replaced by individual symbols in all possible ways (very inefficient)
- No increase in worst-case complexity

Implementations:
- Basic support in some reasoners (KAON2, Pellet)
- Implementation in tableau-based tools complicated
A Combined Example

OWL 2 + Description Logic Rules + DL-safe rules still decidable:

(1) \( \text{Vegetarian}(x) \land \text{Fishproduct}(y) \rightarrow \text{dislikes}(x,y) \)
(2) \( \text{ordered}(x,y) \land \text{dislikes}(x,y) \land O(x) \land O(y) \rightarrow \text{Unhappy}(x) \)
(3) \( \text{ordered}(x,y) \rightarrow \text{Dish}(y) \)
(4) \( \text{dislikes}(x,z) \land \text{Dish}(y) \land \text{contains}(y,z) \rightarrow \text{dislikes}(x,y) \)
(5) \( \rightarrow \text{Vegetarian}(\text{markus}) \)
(6) \( \text{Happy}(x) \land \text{Unhappy}(x) \rightarrow \)
(7) \( \text{markus} \text{ rdf:type [} \)
\( \quad \text{rdf:type} \quad \text{owl:Restriction;} \)
\( \quad \text{owl:onProperty} \quad \text{ordered;} \)
\( \quad \text{owl:someValuesFrom ThaiCurry } \]
(8) \( \text{ThaiCurry} \text{ rdfs:subClassOf [} \)
\( \quad \text{rdf:type} \quad \text{owl:Restriction;} \)
\( \quad \text{owl:onProperty} \quad \text{contains;} \)
\( \quad \text{owl:someValuesFrom Fishproduct } \]
(9) \( \rightarrow O(\text{markus}) \)

\( \rightarrow \text{We cannot conclude} \text{Unhappy(}\text{markus}\text{)} \)
A Combined Example

Explicitly use named individual:

1. \( \text{Vegetarian}(x) \land \text{Fishproduct}(y) \rightarrow \text{dislikes}(x,y) \)
2. \( \text{ordered}(x,y) \land \text{dislikes}(x,y) \land O(x) \land O(y) \rightarrow \text{Unhappy}(x) \)
3. \( \text{ordered}(x,y) \rightarrow \text{Dish}(y) \)
4. \( \text{dislikes}(x,z) \land \text{Dish}(y) \land \text{contains}(y,z) \rightarrow \text{dislikes}(x,y) \)
5. \( \text{ordered}(x,y) \rightarrow \text{Vegetarian}(markus) \)
6. \( \text{Happy}(x) \land \text{Unhappy}(x) \rightarrow \)
7. \( \text{markus} \quad \text{ordered} \quad \text{RedThaiCurry} . \)
8. \( \text{RedThaiCurry} \ rdfs:Type \ \text{ThaiCurry} . \)
9. \( \text{ThaiCurry} \ rdfs:SubClassOf [ \text{rdf:type} \quad \text{owl:Restriction}; \text{owl:onProperty} \quad \text{contains}; \text{owl:someValuesFrom} \quad \text{Fishproduct} \ ] \)

\( \rightarrow \text{Now we can conclude Unhappy}(markus) \)
Summary: Rules

• SWRL (“OWL+ datalog”) is undecidable

• Description Logic Rules:
  – SWRL fragment expressible in OWL 2
  – Supported indirectly by OWL 2 reasoners
  – Definition and translation based on dependency graph

• DL-safe rules:
  – SWRL fragment where variables can only assume concrete values
  – Support by some OWL reasoners
  – DL-safety can be enforced (also done implicitly in some tools)

• Combination OWL 2 + DL Rules + DL-safe rules possible
Rules for the Semantic Web?

• Standards and best-practices for rules still missing
• SWRL syntax most widely used in applications
• W3C RIF (Rule Interchange Format) Working Group
  – Standard for various rule languages, also SWRL-like rules
  – Various new features, e.g. syntax from Frame Logic
  – Official specification expected by end 2009
• Many studies on interfacing Logic Programming and OWL
• OWL 2 RL: a profile that can be translated to datalog rules
  (note: inverse direction of Description Logic Rules)
  \( \rightarrow \) enables some interoperability OWL 2 \( \leftrightarrow \) RIF
• Operational “inference rules” or “production rules” supported by some RDF-stores (e.g. Jena)
Outlook: Semantic Web Research

This lecture covered basic technologies – many challenges remain:

- **How to create ontologies?**
  Engineering aspects, tools, automatic annotation, ontology learning, NLP, evaluation, …

- **How to deploy ontologies?**
  Applications in specific fields, e.g. e-science, life sciences, engineering, e-government, Web Services, …

- **How to process large amounts of semantic information?**
  Scalable reasoning, search and querying, distribution, algorithms, …

- **How to gather and combine semantic data?**
  Information integration, ontology mapping, crawling, …

- **How to explain all this to normal people?**
  User interfaces, deployment processes, tool integration, …

- … contributions are welcome!
"I didn't come here to tell you how this is going to end. I came here to tell you how it's going to begin...

Where we go from there is a choice I leave to you.
Further Reading

- P. Hitzler, S. Rudolph, M. Krötzsch. **Foundations of Semantic Web Technologies.** CRC Press, 2009. (Chapter 6 closely related to this lecture; this also contains more references on types of rules not discussed here)


- S. Abiteboul, R. Hull, V. Vianu. **Foundations of Databases.** Addison Wesley, 1994. (the “Alice book” is an excellent resource on datalog)

Selected research articles:

- M. Krötzsch, S. Rudolph, P. Hitzler. **Description Logic Rules.** In Proc. 18th European Conf. on Artificial Intelligence (ECAI 2008), IOS Press, 2008. (original reference on Description Logic Rules)


- B.N. Grosof, I. Horrocks, R. Volz, S. Decker. **Description logic programs: combining logic programs with description logic.** In Proc. 12th Int. World Wide Web Conference (WWW-03), ACM, 2003. (original paper introducing DLP, a description logic that can be translated to datalog; closely related to OWL 2 RL)