

Deductive Databases and XML

Dietmar Seipel

Julius–Maximilians–Universität Würzburg

Current Revision Date: February 12, 2002

Outline

- Deductive Databases
 - Introduction
 - Representation in XML
- XML and Semi-Structured Data
 - Complex Objects in DDBs
 - Queries and Transformations

The Semantic Web

The challenge of the Semantic Web is to provide

- a language that expresses both data and rules for *reasoning* about the data

and that allows

- rules from any existing *knowledge–representation* system

to be exported onto the Web.

Berners–Lee, Hendler, Lassila (Scientific American)

Knowledge Representation on the Web

- XML, XML-Namespaces
- RDF (Resource Description Framework)
- DAML (DARPA Agent Markup Language)
- OIL (Ontology Inference Layer)
- OWL (OntoWeb Language)
- RuleML (Rule Markup Language)

Knowledge Representation in Logic Programming

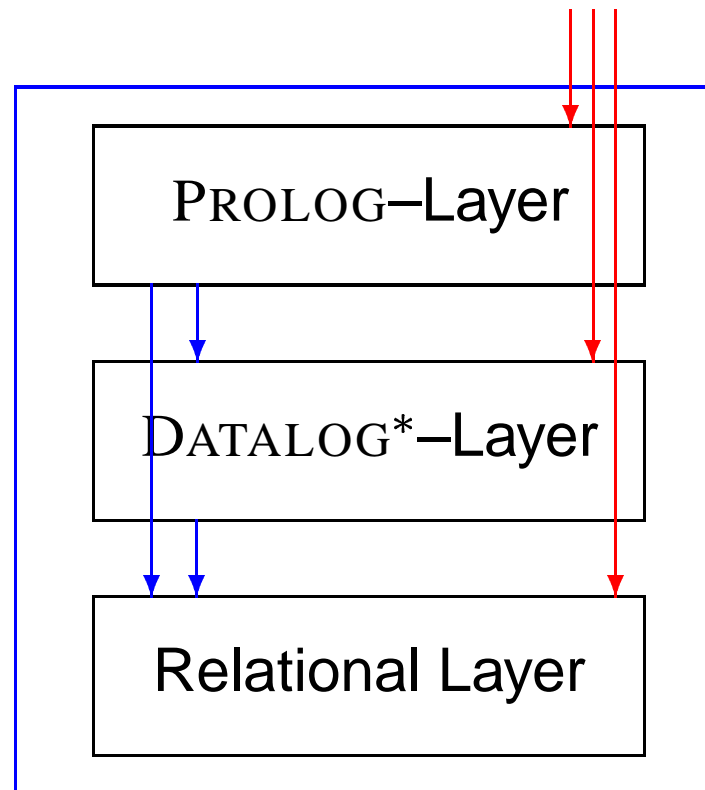
Various **AI-formalisms** can be mapped to extensions of logic programming:

- diagnostic reasoning
- planning under constraints
- reasoning about actions, specifications
- preferential (defeasible) reasoning
- probabilistic reasoning

Part 1: Deductive Databases

- Architecture and Applications
- An Example with Disjunctive Knowledge
- Syntax and Semantics
- Representation in XML

Deductive Databases – Architecture



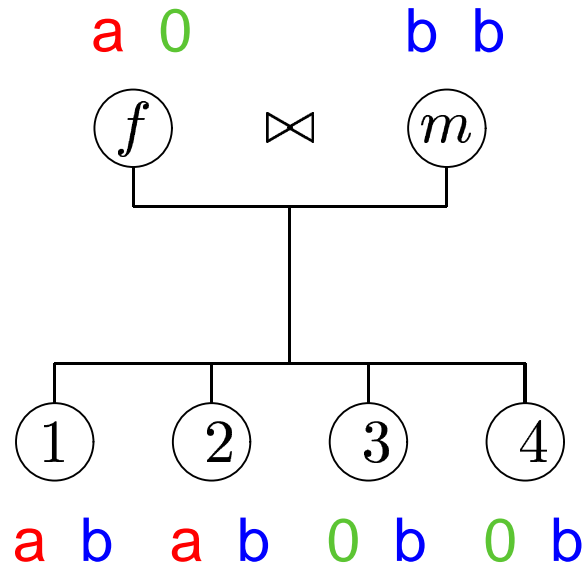
Deductive Databases – Application Areas

Knowledge Representation

- rule-based knowledge
- expert knowledge
- semi-structured knowledge

and Reasoning

Example: Inheritance



$\text{heterozygot}(P, T_1, T_2) \leftarrow \text{genotype}(P, T_1) \wedge \text{genotype}(P, T_2) \wedge T_1 \neq T_2$

$\text{heterozygot}(P) \leftarrow \text{heterozygot}(P, T_1, T_2)$

$\text{homozygot}(P, T) \leftarrow \text{genotype}(P, T) \wedge \text{not heterozygot}(P)$

Inference of Phenotypes

phenotype(P, a) \leftarrow homozygot(P, a)

phenotype(P, a) \leftarrow heterozygot(P, a, 0)

phenotype(P, ab) \leftarrow heterozygot(P, a, b)

phenotype(P, 0) \leftarrow homozygot(P, 0)

Background Knowledge with Disjunction

$\text{genotype}(P, a) \vee \text{genotype}(P, b) \vee \text{genotype}(P, 0) \leftarrow \text{person}(P)$
 $\leftarrow \text{genotype}(P, T_1) \wedge \text{genotype}(P, T_2) \wedge \text{genotype}(P, T_3) \wedge$
 $T_1 \neq T_2 \wedge T_1 \neq T_3 \wedge T_2 \neq T_3$

$\text{genotype}(\text{Parent}, T) \leftarrow$

$\text{parent}(\text{Child}, \text{Parent}) \wedge \text{homozygot}(\text{Child}, T)$

$\text{genotype}(\text{Parent}, T_1) \vee \text{genotype}(\text{Parent}, T_2) \leftarrow$

$\text{parent}(\text{Child}, \text{Parent}) \wedge \text{heterozygot}(\text{Child}, T_1, T_2)$

Syntax of Rules

A **disjunctive rule** r is given by

$$A_1 \vee \dots \vee A_k \leftarrow B_1 \wedge \dots \wedge B_m \wedge \text{not } C_1 \wedge \dots \wedge \text{not } C_n,$$

A_i, B_i, C_i : atoms, “not”: default negation.

$n=0$: positive–disjunctive rule

$n=m=0$: disjunctive fact

$k=1$: normal rule

$k=0$: integrity constraint, denial rule

Logical Terms in XML

$\text{XML}(t) :=$

{	$\langle \text{term constant}="t"/\rangle,$	if t is a constant,
	$\langle \text{term variable}="t"/\rangle,$	if t is a variable,
	$\langle \text{term functor}="f">$	if $t = f(t_1, \dots, t_n)$ is a complex term
	$\text{XML}(t_1) \dots \text{XML}(t_n) ,$	
$\langle / \text{term} \rangle$		

Logical Atoms and Literals in XML

$$\text{XML}(p(t_1, \dots, t_n)) := \begin{cases} \langle \text{atom predicate}="p" \rangle \\ \text{XML}(t_1) \dots \text{XML}(t_n) \\ \langle /atom \rangle \end{cases}$$

$$\text{XML}(\text{not } A) := \begin{cases} \langle \text{negation type}="default" \rangle \\ \text{XML}(A) \\ \langle /negation \rangle \end{cases}$$

Example: A Logical Atom in XML

$p(1, X, f(2, Y)) \mapsto$

```
<atom predicate="p">  
  <term constant="1"/>  
  <term variable="X"/>  
  <term functor="f">  
    <term constant="2"/>  
    <term variable="Y"/>  
  </term>  
</atom>
```

Disjunctive Rules in XML

$\text{XML}(r) :=$
 $\langle \text{rule} \rangle$
 $\langle \text{head type="disjunctive"} \rangle$
 $\text{XML}(A_1) \dots \text{XML}(A_k)$
 $\langle / \text{head} \rangle$
 $\langle \text{body type="conjunctive"} \rangle$
 $\text{XML}(B_1) \dots \text{XML}(B_m)$
 $\text{XML}(\text{not } C_1) \dots \text{XML}(\text{not } C_n)$
 $\langle / \text{body} \rangle$
 $\langle / \text{rule} \rangle$

Semantics of “not” and “ \vee ”

		Default Negation	
		no	yes
Disjunction	no	SLD, $\mathcal{T}_{\mathcal{P}}$	SLDNF, WFS, STABLE
	yes	$\mathcal{T}_{\mathcal{P}}^s, \mathcal{T}_{\mathcal{P}}^I$	WFS ?, STABLE

Cycles

Positive Cycle:

$$path(X, Y) \leftarrow arc(X, Z) \wedge path(Z, Y),$$
$$path(X, Y) \leftarrow arc(X, Y),$$
$$arc(a, b), arc(b, a).$$

Negative Cycle:

$$win(X) \leftarrow move(X, Y) \wedge \text{not } win(Y),$$
$$move(a, b), move(b, a). \quad (\text{Even})$$
$$move(c, c). \quad (\text{Odd})$$

Part 2: XML and Semi-Structured Data

- Complex Objects in XML
- Characteristics of PROLOG
- Complex Objects in Field-Notation
- Queries and Transformations

Complex Objects in XML

```
<rentals>
  <rental start="18.12.2001">
    <movie>
      <title> Something about Mary </title>
      <price_code> new_release </price_code>
      <days_rented> 4 </days_rented>
    </movie>
    ...
  </rental>
  ...
</rentals>
```

Characteristics of PROLOG

- selection of attribute values is by position:

schema:

```
employee(Name,Address,Mgr,Salary)
```

selection of salary:

```
employee( _, _, _, S )
```

- PROLOG can handle tree structures and term structures nicely

Characteristics of PROLOG

- non-destructive assignment of variables:
the update of components within a structure O,

$$O.a = 7,$$

is not possible without creating a new structure O'

- evaluation of arithmetic expressions:

$$X \text{ is } 3 * (4 + 5)$$

Representation of Complex Objects

- attribute names: a_i
- attribute values: v_i

Complex objects O , where $v_i = O.a_i$, $1 \leq i \leq n$, can be represented as association lists in **field notation**:

$$O = [a_1 : v_1, \dots, a_n : v_n].$$

Representation of XML in Extended Field–Notation

```
rentals: [  
  rental: [ start:'18.12.2001' ]: [  
    movie: [ title:'Something about Mary',  
      price_code:new_release, days_rented:4 ],  
    ... ],  
  rental: [ start:'19.12.2001' ]: [  
    movie: [ title:'Bonanza',  
      price_code:regular, length:2, days_rented:7 ],  
    ... ],  
  ... ]
```

Advantages of the Field–Notation

- The sequence of attribute/value–pairs is arbitrary.
- Values can be accessed by attributes rather than by argument positions.
- The database schema can be changed at run time.
- Null values can be omitted, and new values can be added at run time.
- Semi–structured data, such as XML–data, can be represented and queried very elegantly.

Selection of Components

For a complex object $O = [a_1 : v_1, \dots, a_n : v_n]$ we select the value $X = v_i$ of an attribute a_i using “ $X := O^{\wedge}a_i$ ”:

```
?- O = [ a:[ b:2, c:3 ], c:4 ],  
   X := O^a,  
   Y := O^a^b.
```

```
X = [b:2, c:3]
```

```
Y = 2
```

Basic Concepts of the FNPATH–Language

Queries:

- Path Expressions:

$O^a, O^{a^b}, O^{[a_1, a_2, \dots]}, \dots, O?a$

- Aggregation Operators:

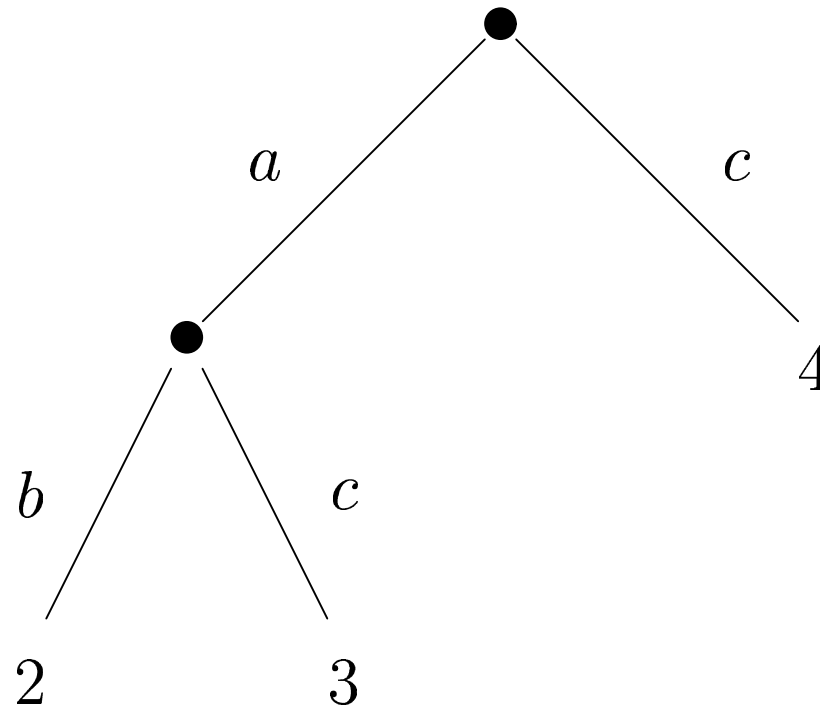
$O\text{-average}^a, O\text{-sum}^a, \dots$

Transformations:

- Modification Operator: $O*[a:v]$

Example: Complex Object

`O = [a:[b:2, c:3], c:4]`



Queries with Variable Path Expressions

```
O = [ a:[ b:2, c:3], c:4 ]
```

```
?- X := O^a^Path.
```

```
X = 2, Path = b ;
```

```
X = 3, Path = c
```

```
?- 2 := O^a^Path.
```

```
Path = b
```

Transformations

```
?- O = [ a:[ b:2, c:3], c:4 ],  
   X := O*[c:5],  
   Y := O*[a^b:1],  
   Z := O^a*[b:1].
```

```
X = [a:[b:2, c:3], c:5]
```

```
Y = [a:[b:1, c:3], c:4]
```

```
Z = [b:1, c:3]
```

Queries with Aggregation

```
?- O = [ a:2, a:3, b:5, a:7 ],  
   X := O-average^a,  
   Y := O-nth(2)^a.
```

```
X = 4
```

```
Y = 3
```

Movie-Query in FNPATH

```
?- O = [ rentals: <Rentals> ],
   findall( rental: [ start:S ]: [ title:T ],
           ( R := O^rentals^rental,
             [T,Ds] := R^movie^[title,days_rented],
             Ds > 4,
             S := R?start ),
           Long_Rentals ),
   Result = long_rentals: Long_Rentals.
```

XML-Query: FLWR-Expression

```
<long_rentals>
  {
    for $r in document("http://www.movies.de")/rentals/rental
    let $m = $r/movie
    where $m/days_rented > 4
    return
      <rental start={ $r/@start }>
        { $m/title }
      </rental>
  }
</long_rentals>
```

Applications to Semi-Structured Data

1. HTML- and XML-documents (XML-PL):
 - data extraction,
 - data transformation.
2. Reasoning in the Semantic Web:
 - The RuleML-initiative is working on a standard for representing rule systems.

Use Cases of FN and FNPATH

- Data Extraction from Web Documents
- Manipulation and Retrieval of Record Structures
- Refactoring Graphical User Interfaces
- Reasoning about XML-Data describing System Behaviour

Related Concepts

- OQL (O₂ 1988/89, ODMG 1994, Lorel 1997)
- F-Logic (SIGMOD 1989)
- XQL (Feb. 1998), XML-QL (Aug. 1998)
- XSL (1999), XPath (Nov. 1999)
- Quilt, XML Query, XQuery (W3C-Standard 2001)
- XML databases (e.g., TAMINO)

Final Remarks

- Semi-structured data can be processed elegantly using deductive databases and PROLOG.
- Practical applications of knowledge bases can benefit from [advanced features](#), such as disjunctive or uncertain information, aggregation, and cardinality constraints.