Probabilistic logics for defining and using P2P service descriptions

Henrik Nottelmann, Norbert Fuhr

QMIR Seminar

London, UK

December 18, 2003
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
4. DAML+OIL and Datalog, pDAML+OIL
5. Match-making rules
6. Service cost estimation
7. Conclusion and outlook
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
4. DAML+OIL and Datalog, pDAML+OIL
5. Match-making rules
6. Service cost estimation
7. Conclusion and outlook
Motivation (1)

Scenario: P2P network with large number of (library) services, e.g.
  - search services
  - schema mapping services (for queries, for results)
  - query modification services

Goal: services dynamically join and leave
  ⇒ compute execution plan for services for a given task
  1. textual service descriptions (DAML-S)
  2. match-making (probabilistic Datalog)
  3. cost estimation and decision
Motivation (2)

Work in progress: within the DFG/NSF project PEPPER (together with CMU)

- distributed Digital Libraries in peer-to-peer networks
- resource selection (search services)
- service selection
- heterogeneous collections

project just started
Outline

1. Motivation

2. DAML-S

3. Lower service ontology for library services

4. DAML+OIL and Datalog, pDAML+OIL

5. Match-making rules

6. Service cost estimation

7. Conclusion and outlook
DAML Services (DAML-S)

**DAML-S**: upper ontology for describing services, based on DAML+OIL

![Diagram of DAML-S](chart)

- **Resource**: provides
- **Service**: presents supports
- **Profile**: What the service does
- **Grounding**: How to access the service
- **Model**: How the service works

Henrik Nottelmann, Norbert Fuhr
DAML Services (DAML-S)

DAML-S: upper ontology for describing services, based on DAML+OIL

- Resource
  - provides
  - Service
  - presents
  - Profile
    - What the service does
  - Model
    - How the service works
  - Grounding
    - How to access the service
  - supports
  - describesBy

Henrik Nottelmann, Norbert Fuhr
**DAML Services (DAML-S)**

**DAML-S:** upper ontology for describing services, based on DAML+OIL

```
Resource
   provides

Service
   presents
   supports

Profile
   describesWhat the service does

Grounding
   describesHow to access the service

Model
   describesHow the service works
```

Henrik Nottelmann, Norbert Fuhr
DAML Services (DAML-S)

**DAML-S:** upper ontology for describing services, based on DAML+OIL

- **Resource**
  - provides
  - describesBy
  - supports

- **Service**
  - presents
  - describesBy
  - supports

- **Profile**
  - What the service does

- **Model**
  - How the service works

- **Grounding**
  - How to access the service
Components of DAML-S: Profile

**Profile**: describes what the services does (for match-making)
Components of DAML-S: Profile

Profile: describes what the services does (for match-making)
Components of DAML-S: Profile

Profile: describes what the services does (for match-making)
Components of DAML-S: Profile

Profile: describes what the services does (for match-making)
Components of DAML-S: Model

**Process Model:** describes how the service works (for in-depth analysis)
Components of DAML-S: Model

**Process Model**: describes how the service works (for in-depth analysis)
Components of DAML-S: Model

(Process) Model: describes how the service works (for in-depth analysis)
Components of DAML-S: Model

(Process) Model: describes how the service works (for in-depth analysis)
Components of DAML-S: Model

(Process) Model: describes how the service works (for in-depth analysis)
Components of DAML-S: Model

(Process) Model: describes how the service works (for in-depth analysis)
Components of DAML-S: Model

(Process) Model: describes how the service works (for in-depth analysis)
DAML-S for library services (1)

Profile: used by the match-making algorithm

Process:

- currently: only atomic processes, 1:1 relationship to profiles
- in future: maybe consider also composite processes for match-making
- match-making result could be expressed as composite process
  (not in this talk)

Grounding: used for calling the selected processes, e.g. via WSDL
  (not in this talk)
DAML-S for library services (2)

DAML-S: provides upper service ontology (vocabulary for defining arbitrary services)

In this talk:

- present lower ontology for library services (profile/process)
- use profiles for match-making
- descriptions of atomic processes shorter than for profiles
  ⇒ use corresponding processes only in this talk
  ⇒ adaption to profiles easy to do
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
4. DAML+OIL and Datalog, pDAML+OIL
5. Match-making rules
6. Service cost estimation
7. Conclusion and outlook
Process ontology: Search services (1)

Ontology: contains definition for generic search services
Process ontology: Search services (1)

**Ontology:** contains definition for generic search services
Ontology: contains definition for generic search services
Concrete search services: subclasses with specialised input/output
Process ontology: Other services

Query transformation: for heterogeneous schemas
  e.g. DCQuery $\mapsto$ ACMQuery

Result transformation: for heterogeneous schemas
  e.g. ACMResult $\mapsto$ DCRresult

Query modification: use relevance judgements
  e.g. DCQuery $\times$ DCRresult $\mapsto$ DCQuery
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
   4. DAML+OIL and Datalog, pDAML+OIL
4. Match-making rules
5. Service cost estimation
6. Conclusion and outlook
Probabilistic Datalog

Definition of match-making rules: not possible in DAML+OIL

⇒ probabilistic Datalog

• variant of predicate logic based on function-free Horn clauses
• negation is allowed (restricted use)
• probabilistic facts and rules
Probabilistic Datalog

Definition of match-making rules: not possible in DAML+OIL

⇒ probabilistic Datalog

• variant of predicate logic based on function-free Horn clauses
• negation is allowed (restricted use)
• probabilistic facts and rules

\[
\begin{align*}
\text{person}(\text{paul}) & . \quad \text{man}(\text{peter}) . \quad \text{woman}(\text{mary}) . \quad 0.5 \quad \text{man}(\text{jo}) . \quad 0.8 \quad \text{parent}(\text{jo}, \text{peter}). \\
\text{father}(X, Y) & : - \quad \text{parent}(X, Y) \; \& \; \text{man}(X) . \\
0.5 \quad \text{man}(X) & : - \quad \text{person}(X) . \\
\Rightarrow \quad & 0.4 \quad \text{father}(\text{jo}, \text{peter}). \\
& 0.5 \quad \text{man}(\text{paul}).
\end{align*}
\]
Four-valued Probabilistic Datalog

**Now 4 truth values:** true, false, inconsistent, unknown

0.5/0.3/0.1/0.1 man(jo).

**Open-world assumption:** replaces CWA in pDatalog

**Evaluation:** mapped onto pDatalog, two predicates for positive/negative knowledge

\[
\begin{align*}
\text{woman}(X) & : - !\text{man}(X). \\
!\text{woman}(X) & : - \text{man}(X). \\
\text{p\_woman}(X) & : - !\text{p\_man}(X) \& \text{n\_man}(X). \\
\text{n\_woman}(X) & : - \text{p\_man}(X) \& !\text{n\_man}(X).
\end{align*}
\]
DAML+OIL and Datalog (1)

Basic idea:

```
subClassOf(dl:Search, process:AtomicProcess).
subPropertyOf(dl:query, process:input).
domain(dl:query, dl:Search).
range(dl:query, dl:Query).
...```

Henrik Nottelmann, Norbert Fuhr
DAML+OIL and Datalog (1)

Basic idea:

```
subClassOf(dl:Search, process:AtomicProcess).
subPropertyOf(dl:query, process:input).
domain(dl:query, dl:Search).
range(dl:query, dl:Query).
...
```
Basic idea:

\[
\begin{align*}
\text{subClassOf}(\text{dl:Search}, \text{process:AtomicProcess}). \\
\text{subPropertyOf}(\text{dl:query}, \text{process:input}). \\
\text{domain}(\text{dl:query}, \text{dl:Search}). \\
\text{range}(\text{dl:query}, \text{dl:Query}). \\
\end{align*}
\]
DAML+OIL and Datalog (1)

Basic idea:

subClassOf(dl:Search, process:AtomicProcess).
subPropertyOf(dl:query, process:input).
domain(dl:query, dl:Search).
range(dl:query, dl:Query).

...
DAML+OIL and Datalog (1)

Basic idea:

subClassOf(dl:Search, process:AtomicProcess).
subPropertyOf(dl:query, process:input).
domain(dl:query, dl:Search).
range(dl:query, dl:Query).
...

Henrik Nottelmann, Norbert Fuhr
DAML+OIL and Datalog (2)

Slight changes in constant syntax: syntactic sugar only

- URIs
- strings in quotes

Open-world assumption: suitable for DAML+OIL

“We cannot say anything about statements we don’t have in our knowledge base.”
DAML+OIL and Datalog (2)

Slight changes in constant syntax: syntactic sugar only

- URIs
- strings in quotes

Open-world assumption: suitable for DAML+OIL

“We cannot say anything about statements we don’t have in our knowledge base.”

General purpose relation: allows for writing generic rules

\[
\text{stat}(C, \text{rdf:type, daml:Class}) :- \text{class}(C).
\]
\[
\text{stat}(O, \text{rdf:type, C}) :- \text{type}(O, C).
\]
DAML+OIL and Datalog (3)

**DAML+OIL modelling primitives:** expressed as rules

**Transitive properties:** use general purpose relation \(\text{stat} \)

\[\text{TransitiveProperty(ancestor).}\]

\[\text{stat}(X,P,Z) :- \text{stat}(X,P,Y) \& \text{stat}(Y,P,Z) \& \text{TransitiveProperty}(P).\]
DAML+OIL and Datalog (3)

DAML+OIL modelling primitives: expressed as rules

Transitive properties: use general purpose relation stat

\[ \text{TransitiveProperty(ancestor)}. \]

\[ \text{stat}(X,P,Z) := \text{stat}(X,P,Y) \& \text{stat}(Y,P,Z) \& \text{TransitiveProperty}(P). \]

Subclasses:

\[ \text{subClassOf}(\text{dl:Search,process:AtomicProcess}). \]

\[ \text{type}(O,\text{C1}) := \text{subClassOf}(\text{C2,}\text{C1}) \& \text{type}(O,\text{C2}). \]
**DAML+OIL and Datalog (4)**

**Disjoint classes:** require use of four-valued probabilistic Datalog

```prolog
disjointWith(flight:Airport, travel:City).

!type(0, C1) :- disjointWith(C1, C2) & type(0, C2).
!type(0, C2) :- disjointWith(C1, C2) & type(0, C1).
```
DAML+OIL and Datalog (4)

**Disjoint classes:** require use of four-valued probabilistic Datalog

\[
\text{disjointWith(flight:Airport,travel:City)}.
\]

\[
!\text{type}(0,C1) :- \text{disjointWith}(C1,C2) \land \text{type}(0,C2).
\]

\[
!\text{type}(0,C2) :- \text{disjointWith}(C1,C2) \land \text{type}(0,C1).
\]

**Cardinality descriptions:**

\[
\text{different}(P,\text{"1"}) :- \text{stat}(X,P,Y1).
\]

\[
\text{different}(P,\text{"3"}) :- \text{stat}(X,P,Y1) \land \text{stat}(X,P,Y2) \land \text{stat}(X,P,Y3) \land
\]

\[
\langle Y1,Y2 \rangle \land \langle Y1,Y3 \rangle \land \langle Y2,Y3 \rangle.
\]

\[
!\text{maxCardinality}(R,N) :- \text{onProperty}(R,P) \land \text{type}(R,daml:Restriction) \land
\]

\[
\text{different}(P,NN) \land \text{add}(NN,N,\text{"1"}).
\]

\[
!\text{minCardinality}(R,N) :- \text{onProperty}(R,P) \land \text{type}(R,daml:Restriction) \land
\]

\[
!\text{different}(P,N).
\]
Excursion: pDAML+OIL

Probabilistic version of DAML+OIL: by mapping onto pDatalog

<Man rdf:ID="jo" pdaml:prob="0.5/0.3/0.1/0.1"/>

0.5/0.3/0.1/0.1 man(jo).

Semantics of DAML+OIL primitives: given by underlying rules

0.5 subClassOf(#A,#B).

⇒ The probability that an instance of #A is also an instance of #B is 50%.

probabilistic weights do not make sense for all cases:

0.5 type(t:Man,rdfs:Class).

Rules in pDAML+OIL: map onto corresponding pDatalog rules
Outline

1. Motivation

2. DAML-S

3. Lower service ontology for library services

4. DAML+OIL and Datalog, pDAML+OIL

5. Match-making rules

6. Service cost estimation

7. Conclusion and outlook
Match-making rules (1)

**Match-making rules:** use facts derived from DAML-S directly

**Service descriptions:** here: simplified model

```
# service(name,input,output)

service(dl:DCQueryModification,dl:DCQuery_DCResult,dl:DCQuery).
service(dl:DC2ACMQuery,dl:DCQuery,dl:ACMQuery).
service(dl:ACMSearch,dl:ACMQuery,dl:ACMResult).
service(dl:ACM2DCResult,dl:ACMResult,dl:DCResult).

task(mytask,dl:DCQuery_DCResult,dl:DCResult).
```

**Input/output type matching:** first argument is super-set of second argument

```
match(dl:DCQuery_DCResult,dl:DCQuery).
match(dl:DCQuery_DCResult,dl:DCResult).
match(dl:ACMQuery,dl:ACMQuery).
...
```
Match-making rules (2)

**Goal:** execution plan

- DCQueryModification
- DC2ACMQuery
- ACMSearch
- ACM2DCResult
Match-making rules (2)

Goal: execution plan

Idea: $\text{chain}(S_1, S, S_2)$ iff there is a chain beginning with $S_1$, ending with $S_2$, and with $S$ in between
**Match-making rules (2)**

**Goal:** execution plan

**Idea:** $\text{chain}(S_1, S, S_2)$ iff there is a chain beginning with $S_1$, ending with $S_2$, and with $S$ in between
Chaining 2 services: with matching input/output

chain(S1, null, S2) :- service(S1, I1, O1) & service(S2, I2, O2) & match(O1, I2).

=> chain(dl:DCQueryModification, null, dl:DC2ACMQuery).
=> chain(dl:DC2ACMQuery, null, dl:ACMSearch).
=> ...
Match-making rules (3)

**Chaining 2 services:** with matching input/output

\[
\text{chain}(S1, \text{null}, S2) : \text{service}(S1, I1, O1) \land \text{service}(S2, I2, O2) \land \text{match}(O1, I2).
\]

\[\Rightarrow \text{chain} (\text{dl:DCQueryModification}, \text{null}, \text{dl:DC2ACMQuery}).\]
\[\Rightarrow \text{chain} (\text{dl:DC2ACMQuery}, \text{null}, \text{dl:ACMSearch}).\]
\[\Rightarrow \ldots\]

**Chaining >2 services:** transitive closure

\[
\text{chain}(S1, S, S2) : \text{chain}(S1, S11, S) \land \text{chain}(S, S22, S2).
\]

\[\Rightarrow \text{chain} (\text{dl:DCQueryModification}, \text{dl:DC2ACMQuery}, \text{dl:ACMSearch}).\]
\[\Rightarrow \text{chain} (\text{dl:DCQueryModification}, \text{dl:ACMQuery}, \text{dl:ACM2DCResult}).\]
\[\Rightarrow \text{chain} (\text{dl:DC2ACMQuery}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).\]
Match-making rules (4)

Execution plan: filter chains with correct input/output

\[
\text{plan}(T, S_1, S, S_2) \leftarrow \text{task}(T, T_I, T_O) \land \text{chain}(S_1, S, S_2) \land \\
\text{service}(S_1, I, O_1) \land \text{match}(T_I, I) \land \\
\text{service}(S_2, O_2, O) \land \text{match}(O, T_O).
\]

\[\Rightarrow \text{plan}(\text{dl:DCQueryModification}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).\]

\[\Rightarrow \text{plan}(\text{dl:DC2ACMQuery}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).\]

Henrik Nottelmann, Norbert Fuhr
Match-making rules (4)

Execution plan: filter chains with correct input/output

\[
\text{plan}(T, S_1, S, S_2) :\neg \text{task}(T, T_I, T_O) \& \text{chain}(S_1, S, S_2) \& \\
\text{service}(S_1, I, O_1) \& \text{match}(T_I, I) \& \\
\text{service}(S_2, O_2, O) \& \text{match}(O, T_O).
\]

\[
\Rightarrow \text{plan}(\text{dl:DCQueryModification}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).
\]

\[
\Rightarrow \text{plan}(\text{dl:DC2ACMQuery}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).
\]

Complete plan: ask for intermediary steps

\[
?- \text{chain}(\text{dl:DCQueryModification}, S, \text{dl:ACMSearch}).
\]

\[
\Rightarrow (\text{dl:DC2ACMQuery}).
\]
**Match-making rules (4)**

**Execution plan:** filter chains with correct input/output

\[
\text{plan}(T,S1,S,S2) :\text{ task}(T,TI,TO) \& \text{ chain}(S1,S,S2) \& \\
\text{ service}(S1,I,O1) \& \text{ match}(TI,I) \& \\
\text{ service}(S2,O2,O) \& \text{ match}(O,TO).
\]

\[
\Rightarrow \text{ plan}(\text{dl:DCQueryModification}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).
\]

\[
\Rightarrow \text{ plan}(\text{dl:DC2ACMQuery}, \text{dl:ACMSearch}, \text{dl:ACM2DCResult}).
\]

**Complete plan:** ask for intermediary steps

\[
?- \text{ chain}(\text{dl:DCQueryModification}, S, \text{dl:ACMSearch}).
\]

\[
\Rightarrow (\text{dl:DC2ACMQuery}).
\]

\[
?- \text{ chain}(\text{dl:DCQueryModification}, S, \text{dl:DC2ACMQuery}).
\]

\[
\Rightarrow (\text{null}).
\]
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
4. DAML+OIL and Datalog, pDAML+OIL
5. Match-making rules
6. Service cost estimation
7. Conclusion and outlook
Cost estimation with match-making rules

**Probabilistic variant:** use probabilities as primitive kind of cost estimation (quality of a service)

weight < 0.5: quality loss, weight > 0.5: quality improvement

0.7 service($dl:DCQueryModification,dl:DCQuery_DCRresult,dl:DCQuery$).
0.4 service($dl:DC2ACMQuery,dl:DCQuery,dl:ACMQuery$).
0.8 service($dl:ACMSearch,dl:ACMQuery,dl:ACMResult$).
0.4 service($dl:ACM2DCResult,dl:ACMResult,dl:DCResult$).
Cost estimation with match-making rules

Probabilistic variant: use probabilities as primitive kind of cost estimation (quality of a service)

weight < 0.5: quality loss, weight > 0.5: quality improvement

0.7 service(dl:DCQueryModification, dl:DCQuery_DCResult, dl:DCQuery).
0.4 service(dl:DC2ACMQuery, dl:DCQuery, dl:ACMQuery).
0.8 service(dl:ACMSearch, dl:ACMQuery, dl:ACMResult).
0.4 service(dl:ACM2DCResult, dl:ACMResult, dl:DCResult).

Execution plan: weight specifies quality of plan, then normalise (0.5^|services|)

=> 0.0896 plan(dl:DCQueryModification, dl:ACMSearch, dl:ACM2DCResult).
=> 0.1280 plan(dl:DC2ACMQuery, dl:ACMSearch, dl:ACM2DCResult).

normalise: 1.4336 dl:DCQueryModification, 1.024 dl:DC2ACMQuery
Cost estimation: Decision-theoretic framework (1)

Idea: cost-optimum search service selection

- assign retrieval costs $C(s, q)$ when $s$ documents are retrieved
- compute for each library number of documents to retrieve, so that overall expected costs are minimised

Costs: different sources with user-specific weighting (different policies)

Effectiveness: $\#rel \cdot C^{rel} + \#nonrel \cdot C^{\neg \text{rel}}$

Time: affine linear function (computation and communication time)

Money: linear costs (charges per retrieved document)
Estimating the number of relevant documents in first ranks: requires some metadata about collection, e.g. moments of indexing weights.

Retrieval model: uncertain inference, linear model

\[
\begin{align*}
0.7 \text{ match}(D,q) & \quad \text{:- text\_contains}(D, "ir"). \\
0.3 \text{ match}(D,q) & \quad \text{:- text\_contains}(D, "filtering").
\end{align*}
\]

Probabilities of inference \( P_r(q \leftarrow d) \) mapped onto probabilities of relevance \( P_r(\text{rel} | q, d) \) with mapping function (ECIR 2003, IRJ 6(4), 2003).
Indexing weights: approximated by normal distribution

\[ p(x, \mu, \sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \]

Probabilities of inference: also follow normal distribution with

\[ \mu = \sum_{t \in q} Pr(q \leftarrow t) \cdot \mu_t \]

\[ \sigma = \sqrt{\sum_{t \in q} (Pr(q \leftarrow t) \cdot \sigma_t)^2} \]
Outline

1. Motivation
2. DAML-S
3. Lower service ontology for library services
4. DAML+OIL and Datalog, pDAML+OIL
5. Match-making rules
6. Service cost estimation
7. Conclusion and outlook
Conclusion and outlook

Conclusion:

• use DAML-S and lower ontology for describing library services
• map DAML-S models onto probabilistic Datalog
• apply match-making rules for creating execution plan

Outlook:

• create detailed lower ontology for library services
• model execution plan in DAML-S (composite process)
• find general cost model for services
• implementation including service grounding