Probabilistic logics for defining and using P2P service descriptions

Henrik Nottelmann, Norbert Fuhr

MMGPS Workshop

London, UK

December 16, 2003

Outline

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

Motivation (1)

Scenario: P2P network with large number of (web) services

in our case: library services, e.g.

- search services
- schema mapping services (for queries, for results)
- query modification services

Goal: dynamically 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 (not in this talk)

Motivation (2)

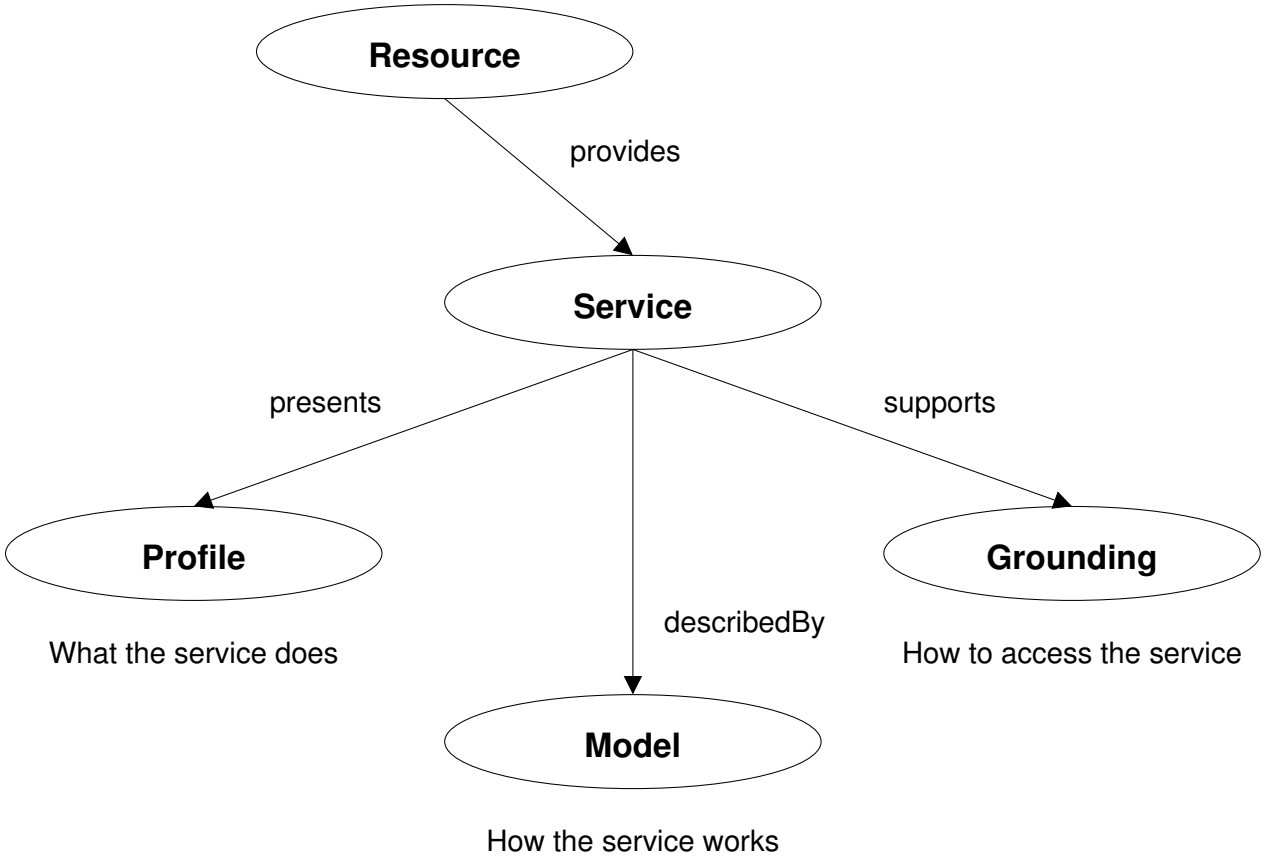
Work in progress: within the DFG/NSF project PEPPER

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

project just started

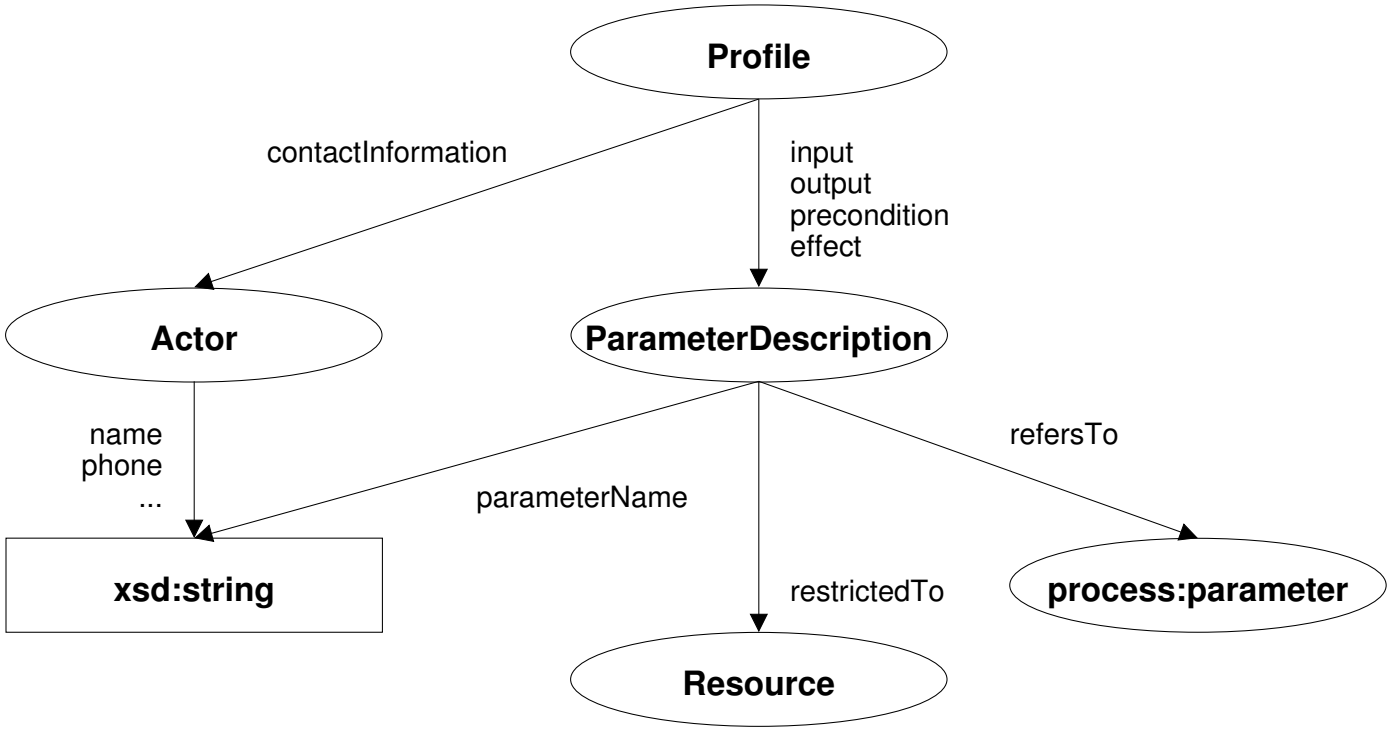
DAML Services (DAML-S)

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



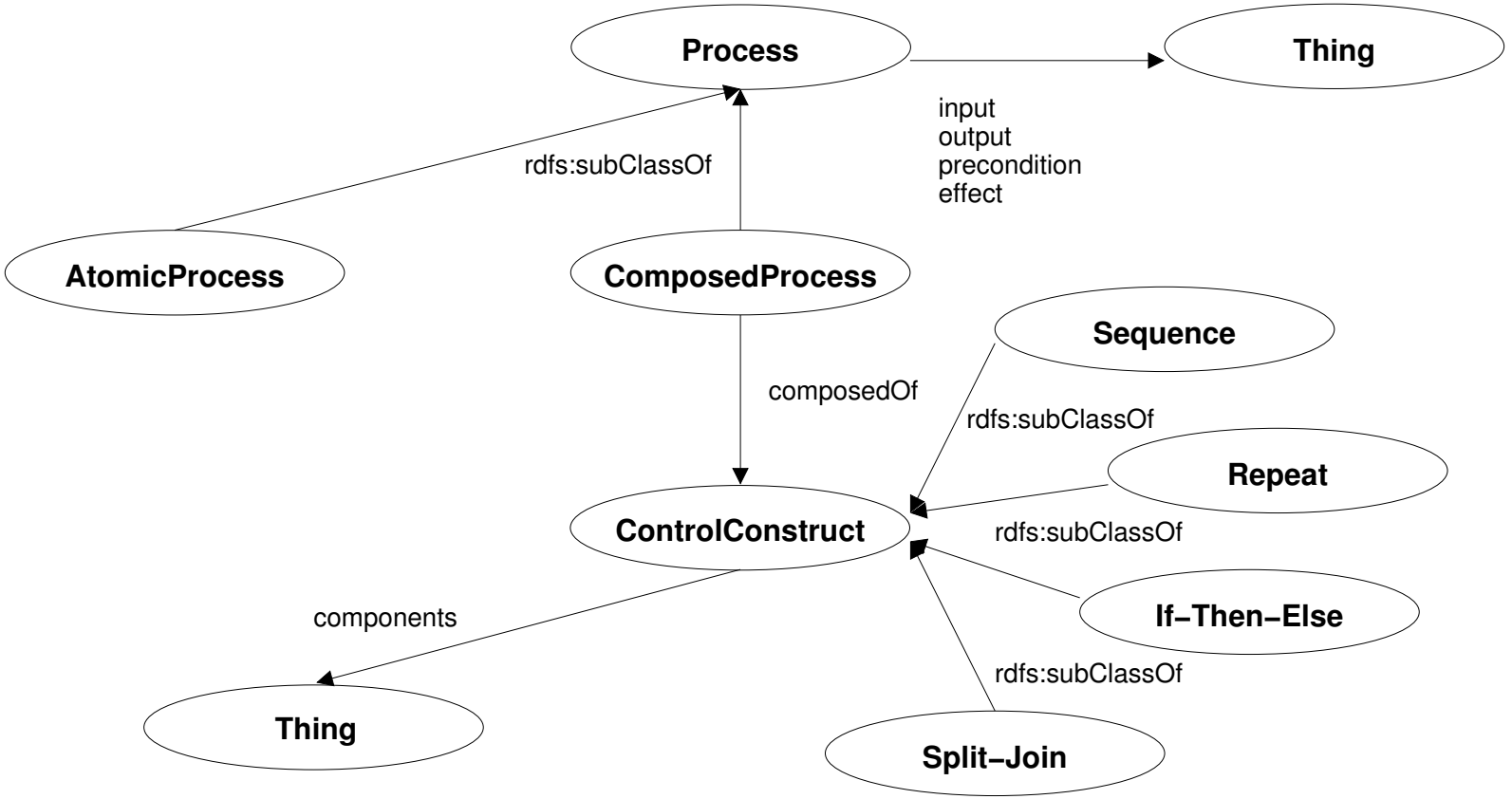
Components of DAML-S: Profile

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)



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:

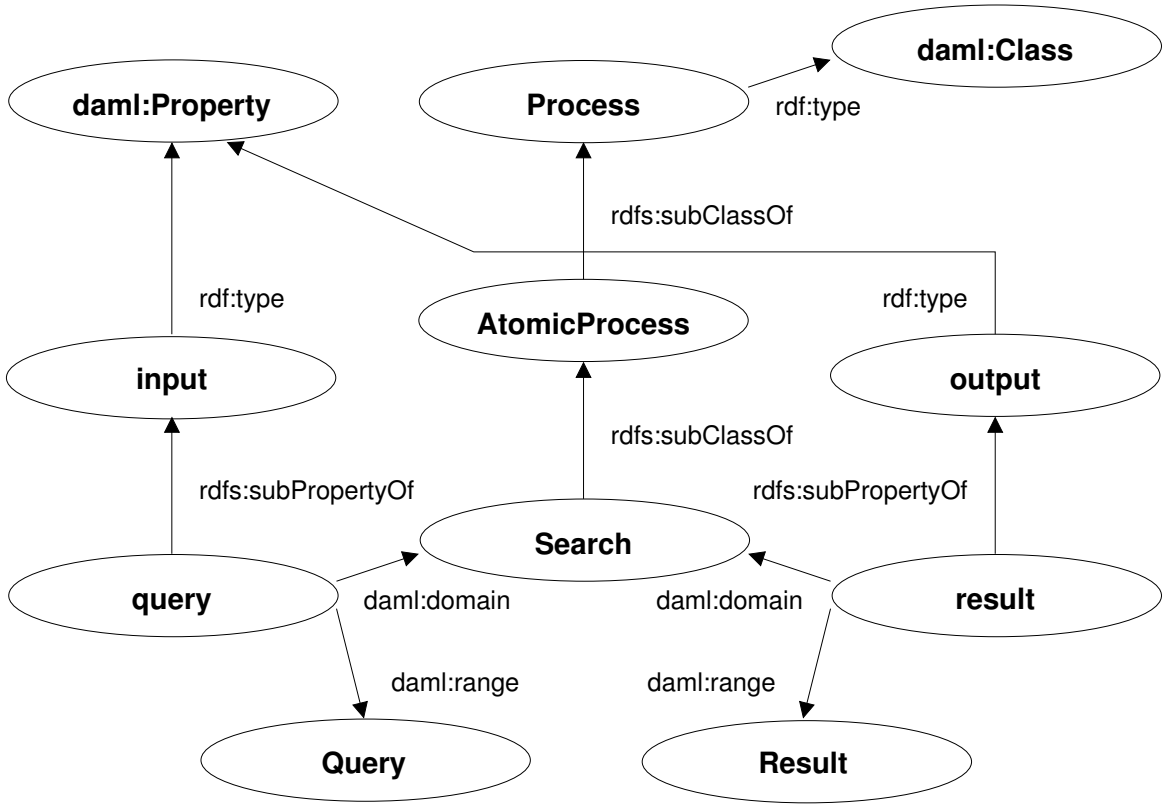
- present lower ontology for library services (profile/process)
- use profiles for match-making
- descriptions of atomic processes shorter than for profiles

  $\Rightarrow$ use corresponding processes only in this talk

  $\Rightarrow$ adaption to profiles easy to do

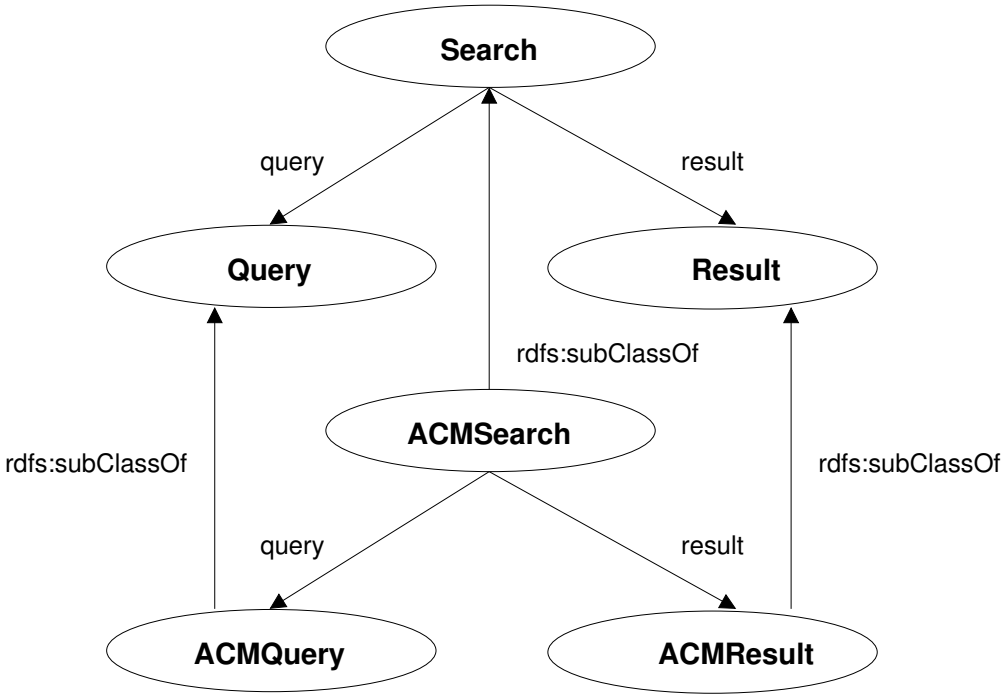
Process ontology: Search services (1)

Ontology: contains definition for generic search services



Process ontology: Search services (2)

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 $\mapsto$ DCResult

Query modification: use relevance judgements

e.g. DCQuery $\times$ DCResult $\mapsto$ DCQuery

Probabilistic Datalog

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

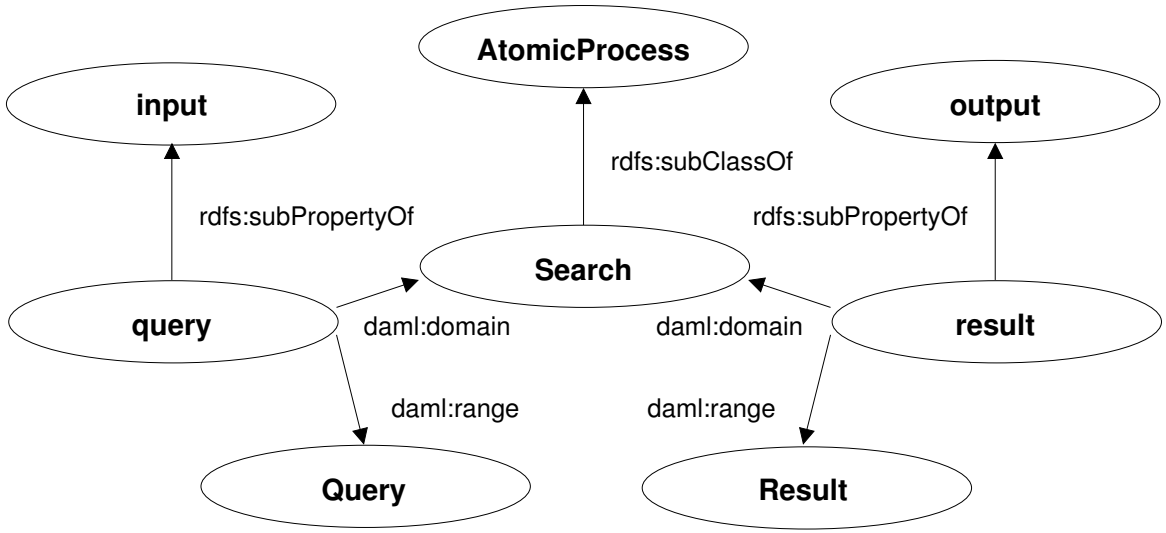
$\Rightarrow$ probabilistic Datalog

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

```
person(paul). man(peter). woman(mary). 0.5 man(jo). 0.8 parent(peter,jo).

father(X,Y) :- parent(X,Y) & man(X).
0.5 man(X) :- person(X).

=> 0.4 father(peter,jo).
   0.5 man(paul).
```

DAML+OIL and Datalog

Example:




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

Match-making rules (1)

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

Service descriptions: here: simplified model

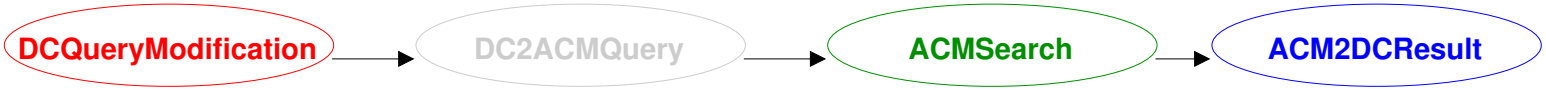
```
# service(name,input,output)

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

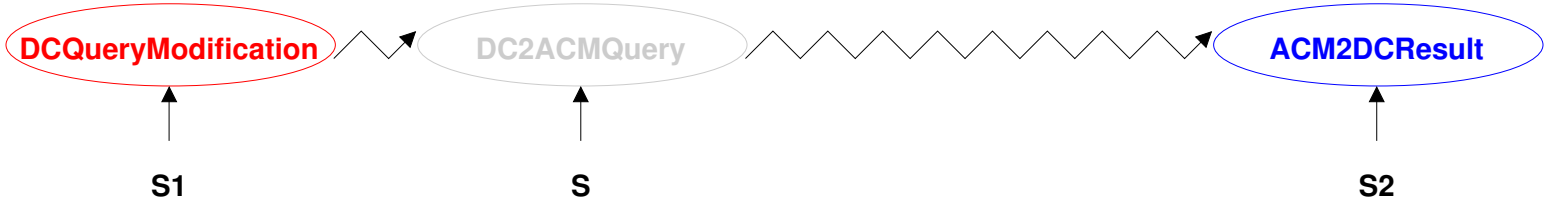
```
match(dl:DCQuery_DCResult,dl:DCQuery).
match(dl:DCQuery_DCResult,dl:DCResult).
match(dl:ACMQuery,dl:ACMQuery).
...
```

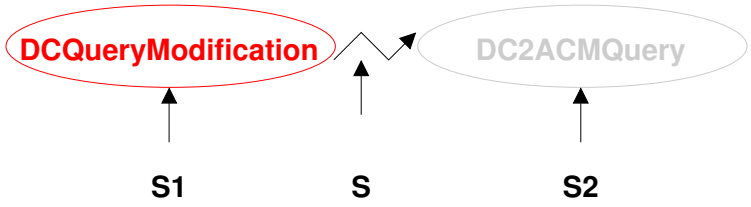
Match-making rules (2)

Goal: execution plan



Idea: `chain(S1,S,S2)` iff there is a chain beginning with `S1`, ending with `S2`, and with `S` in between





Match-making rules (3)

Chaining 2 services: with matching input/output

```
chain(S1,null,S2) :- service(S1,I,SO1) & service(S2,SI2,O) & match(SO1,SI2).

=> chain(dl:ACMSearch, null, dl:ACM2DCResult).
=> ...
```

Chaining >2 services: transitive closure

```
chain(S1,null,S2) :- chain(S1,S11,S) & chain(S,S22,S2).

=> chain(dl:DCQueryModification, dl:DC2ACMQuery, dl:ACMSearch).
=> chain(dl:DCQueryModification, dl:ACMQuery, dl:ACM2DCResult).
=> chain(dl:DC2ACMQuery, dl:ACMSearch, dl:ACM2DCResult).
```

Match-making rules (4)

Execution plan: filter chains with correct input/output

```
plan(T,S1,S,S2) :- task(T,TI,TO) & chain(S1,S,S2) &
                   service(S1,I,O1) & match(TI,I) &
                   service(S2,O2,O) & match(O,TO).

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

Complete plan: ask for intermediary steps

```
?- chain(dl:DCQueryModification, S, dl:ACMSearch).

=> (dl:DC2ACMQuery).


?- chain(dl:DCQueryModification, S, dl:DC2ACMQuery).

=> (null).
```

Match-making rules (5)

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

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)
- implementation including service grounding