Exploiting interaction contexts in P2P ontology mapping

Paolo Besana, Dave Robertson, Micheal Rovatosos

p.besana@sms.ed.ac.uk

Centre for Intelligent System and their Applications
School of Informatics
University of Edinburgh
Introduction

- Open and Peer-to-Peer networks can form the bases for the ascent of virtual communities populated by agents.
- Agents, acting on the behalf of their owner, will enter and exit these communities.
A community of readers and text suppliers
Protocol-Based Coordination

- Most of the interactions between agents follow a predetermined path
- *Protocols* can be conveniently exploited for the interactions
- Lightweight Coordination Calculus (LCC) describes dialogues among different agents
LCC

LCC is based on process calculus, expressed with horn clauses:

- sending and receiving messages are the basic behaviours
- more complex behaviours are expressed with connectives:
  - sequences (then)
  - choices (or)
  - parallelisation (par)
- each step can have preconditions and postconditions
example: reader protocol

\[
\begin{align*}
\text{book\_searcher} & : \text{L,NinLst} \\
\text{likes(Tp,SA)} & \\
\text{tx\{ask(suggestion,Tp,SA,NL),} \\
& \quad \text{a(lib\_suggest,L)} \\
\text{rx\{suggest(T,A),} \\
& \quad \text{a(lib\_suggest,L)} \\
\text{not read(T,A)} & \\
\text{a(book\_searcher(L,NL),R)} & ::= \\
& \text{ask(suggestion,Tp,SA,NL)} \\
& \quad \Rightarrow \text{a(lib\_suggest,L)} \\
& \quad \Leftarrow \text{likes(Tp,SA)} \\
& \text{then} \\
& \quad \text{suggest(T,A)} \Leftarrow \text{a(lib\_suggest,L)} \\
& \quad \text{then} \\
& \quad \text{a(borrower(T,A,L),R)} \Leftarrow \text{not read(T,A)} \\
& \quad \text{or} \\
& \quad \text{a(book\_searcher(L,T | NL,R)}.
\end{align*}
\]
example: library protocol

lib_suggest  L

rx{ask(suggestion, Tp, SA, NL), a(book_searcher(_,_), R)}

satisfy(Tp, SA, NL, T, A)

tx{suggest(T, A), book_searcher(_,_), R}}

a(lib_suggest, L)::=
    ask(suggestion, Tp, SA, NL)
    <= a(book_searcher(_,_), R)
    then
        suggest(T, A)
        => a(book_searcher(_,_), R)
        <-- satisfy(Tp, SA, NL, T, A)
example: message exchange

likes(fiction, orwell)

reader

library
example: message exchange

likes(fiction, orwell)
ask(suggest, novel, orwell, ())

reader
library
example: message exchange

likes(fiction, orwell)

ask(suggest, novel, orwell, ())

suggest('1984', 'G. Orwell')

satisfy(...)
example: message exchange

likes(fiction, orwell)

read(...) → ask(suggest, novel, orwell, ())

suggest('1984', 'G. Orwell') → satisfy(...) → suggest('1984', 'G. Orwell')

ask(suggest, novel, orwell, ('1984'))
example: message exchange

reader

likes(fiction, orwell)

read(...) → ask(suggest, novel, orwell, ())

suggest(‘1984’, ’G. Orwell’)

read(...) → ask(suggest, novel, orwell, (’1984’))

suggest(‘animal farm’, ’G. Orwell’)

library

satisfy(...)
Ontology Mismatch

- Each agent will likely have different ontologies that:
  - reflect different views of their developers
  - mirror the different needs and interests of their owner
- Problems of sharing a common ontology:
  - who imposes it? Why should others accept it?
  - differences can make hard to create a consistent ontology
  - difficult to keep track of the evolution of an ontology
Ontology Mismatch - example 1

Reader Ontology
Ontology Mismatch - example 2

Library Ontology
A more flexible approach:
- finding mappings between the ontologies

Most mapping processes align complete ontologies:
- S-Match
- MAFRA
- QOM
- ...

but...
Ontology mapping in MAS

- In open MAS peers interact with many different peers:
  - it is impossible to foresee all the possible combinations of ontologies
  - interactions are often required to be rapid and can occur simultaneously
  - agents may have ontologies that cover dissimilar domains, with only parts overlapping
Proposed Approach

- A complete mapping is not a requirement for interactions.
- Agents need to share only the parts of their knowledge contextual to the interaction in which they are involved.
- We will present a framework that tries to exploit this to improve the efficiency of ontology mapping.
During an interaction $k$, the framework receives a sequence of external terms $t_j$ and returns the most specific mappings $b_{t_j,k}$.

For each term $t$, it executes an iterative process, composed by 3 steps:

- **Generate Hypotheses**: creates a set of hypotheses
- **Filter Hypotheses**: excludes hypotheses unlikely to be verified
- **Select Hypothesis**: collects and combines evidences in support of the hypotheses, and chooses the strongest one
Framework

Exploiting interaction contexts in P2P ontology mapping – p.15/31
Generate Hypotheses

Initially it generates the most generic bridge for $t$ (e.g. $t \sqsubseteq \text{thing}$)

Then traverses the ontology breadth first

At each iteration $i$ the function $\text{GENERATE-HP}$ receives:

- the bridge $b_{tk(i-1)}$ proved in the previous iteration

returns a set of hypotheses $\Omega$ about the most generic mappings that imply $b_{tk(i-1)}$
Generate Hypotheses

The generated set $\Omega_2$ is:

$$\Omega_2 = \left\{ \langle \subseteq, \supseteq, \equiv \rangle, \text{novel, classification} \right\}, \quad \langle \subseteq, \supseteq, \equiv \rangle, \text{novel, licence} \right\}, \quad \langle \subseteq, \supseteq, \equiv \rangle, \text{novel, storage} \right\}, \quad \ldots \right\}$$
Filter Hypotheses

- Receives a set of hypotheses $\Omega$ and returns a subset $\Omega'$
- If none of the filtered hypotheses is proved, the function may be called again:
  - at each round the function relaxes the filters to obtain a wider set
- In the example, the prefilters may exclude the bridges about the terms licence and storage, as we will see later
Select Best Hypothesis

- *collect evidences*: generate arguments in favour or against the filtered hypotheses using rules

- *combine evidences*: combine arguments to give a confidence level for each hypothesis

- *harvest hypothesis*: the strongest hypothesis is chosen
Rules and prefilters

- Filters reduce the hypotheses that the rules must check.
- Heuristics available improves with the feedback obtained from proved hypotheses.
- The confidence about the result increases and the set of filtered hypotheses narrows.
- In the long run, the prefilters could replace the rules, at least for some external terms, making the mapping process quicker.
Contexts help to focus the search of correct mappings:

- an external term is unlikely to be mapped to a term unrelated from the context of the interaction
Contexts

- Contexts are possible patterns in interactions:
  - some terms tend to appear together in dialogues about similar topics
  - some terms are contextual to the topic of the conversation (*document, download,...*): they do not appear in other conversations
  - other terms are auxiliary to any kind of conversation (*ask, inform,...*)
Contexts - Example

member grade grade of the term to the context

book,0.7
title,0.3
suggestion,0.8
fiction,0.5
essay,0.4
...detective,0.1
science,0.2

license,0.3
title,0.5
download,0.9
member,0.9
...author,0.4
isbn,0.2
loan,0.7

purchase,0.4
license,0.8
price,0.4
download-limit,0.4
title,0.4
...
Contexts - Use

- Contexts are used to classify dialogues as they are performed.

- After a new term is mapped during the interaction, the system reclassifies the dialogue:
  - it searches the contexts that maximise the membership function of the terms encountered in the dialogue up to now

- The new terms that appear in the dialogue are first matched with the terms in the context
parsing message:
ask(suggestion, fiction, orwell, [])

Contexts - Use example

{a} licence, 0.3
   title, 0.5
   download, 0.9
   membership, 0.9

{b} book, 0.7
   title, 0.3
   suggestion, 0.8
   fiction, 0.5
   essay, 0.4
   detective, 0.1
   science, 0.2

{c} purchase, 0.4
   licence, 0.8
   price, 0.4
   download-limit, 0.4
   title, 0.4
   ...
parsing message:
ask (suggestion, fiction, orwell, [])
parsing message: ask(suggestion, fiction, orwell, [])

selected context
the next terms in the dialogue will be matched with the terms in it
Past Mapping Experience

Another possible filter exploits past experiences.

Some external terms may always have been mapped to the same internal terms.

- For example, the external term novel may have always matched with the term fiction defined in the library ontology.
Past Mappings - Use

- This prefilter keeps the hypotheses implied by the past mappings, and discards the others.
Past Mappings - Use

This prefilter keeps the hypotheses implied by the past mappings, and discards the others.
Past Mappings - Use

This prefilter keeps the hypotheses implied by the past mappings, and discards the others.
Past Mappings - Use

- This prefilter keeps the hypotheses implied by the past mappings, and discards the others.

```
  thing
   `-- classification
       `-- content_type
           `-- fiction
           `-- essay

  thing
   `-- document
       `-- text
           `-- novel
           `-- essay
```

subsumed: fiction: essay

equivalent: novel: essay
Consistency

- There is no issue about inconsistency:
  - conflicting past mappings are used as suggestions about the order in which the hypotheses should be checked
  - conflicting hypotheses are tolerated by collecting evidence in favour or against them.
Related Work

- The QOM project (Quick Ontology Mapping) addresses the problem of trading quality for efficiency.
- Possible mapping candidates are filtered using different strategies in order to reduce the time spent computing similarities between unrelated terms.
- It is oriented toward mapping whole ontologies: there is no concern about the contexts of interactions.
- Filters are only based on the ontologies themselves (hierarchy, node labels, etc).
Conclusion

- Presented a framework for mapping ontologies dynamically in open environments
- Only the relevant portions of ontologies are mapped:
  - terms are mapped when encountered in dialogues
  - mapping candidates are filtered using the context of the interaction
- Still a lot of work to do
Thank you for the attention