Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres

Elena Nardini    Mirko Viroli    Emanuele Panzavolta

Alma Mater Studiorum – Università di Bologna
{elena.nardini,mirko.viroli}@unibo.it
{emanuele.panzavolta}@studio.unibo.it

SAC 2010

Sierre, Switzerland
March 26, 2010
1 Introduction

2 Towards Semantic Tuple Centres

3 Conclusions

4 Bibliography
Today’s software systems like pervasive systems, internet applications, and Web-service-based systems, are mainly characterised by two main features:

- **distribution** (of control, spatial, and temporal)
- **openness**—dynamism and heterogeneity

**Linda tuple-space model** as basic coordination abstraction [Gelernter, 1985].

- Tuple space as a coordination medium.
- Communication based on **tuples**, **templates**, and a **tuple matching mechanism**.
- Operations **out**, **in**, and **rd** as coordination language.
Towards Semantic Tuple Spaces

- Current research trends in the area of coordination middleware propose **semantic tuple space computing** which enriches tuple spaces semantically to cope with heterogeneity of the structure of the exchanged **tuples** [Nixon et al., 2008].

- **Semantic description of information** – tuple content – through an **ontology language**.

- **Logical reasoning** over such descriptions to support information matching—matching between tuples and tuple templates.
Outline

1. Introduction
2. Towards Semantic Tuple Centres
3. Conclusions
4. Bibliography
Aim

- Take **tuple centre** [Omicini and Denti, 2001b] as a coordination model since it provides programmable tuple spaces.

- Differently from the current approaches [Nixon et al., 2008], to enrich the tuple centre model semantically maintaining the model identity without any assumption about the application domain.

- Implement semantic tuple centres in **TuCSoN** [Omicini and Zambonelli, 1999]—a coordination infrastructure providing tuple centres distributed over the network.
Semantic Tuple Centre Model

- **Domain ontology** allowing to interpret the semantics associated to the knowledge (set of tuples) stored into a tuple centre.

- **Domain objects** – represented by tuples – described so that they can be interpreted in a semantic way, by means of the domain ontology.

- **Semantic tuple templates** as descriptions of a domain object set.

- **Semantic tuple matching mechanism** providing the domain objects – tuples – described through templates.
Ontologies and Individuals in Tuple Centres

- **SHOIN(D)** Description Logic formalism [Baader et al., 2003, Horrocks et al., 2003] to describe domain ontologies and objects.
  - Good compromise between expressiveness and complexity.
  - Theoretical counterpart of **OWL DL**, that is one of the three species of W3C OWL. OWL being a standard, it well fits the openness requirement.
Domain Ontology I

- Described in the form of a **terminology** through a so called **TBox**—a set of **concept** and **role** descriptions.
- Through a set of constructors described in [Horrocks et al., 2003] (Union $C \sqcap D$, Intersection $C \sqcup D$, Negation $\neg C$, Exists $\exists R.C$, etc.).
- Through the operators $\sqsubseteq$ (Inclusion) and $\equiv$ (Equality), in order to define a taxonomy of concepts or roles.
Domain Ontology II

- Each tuple centre is associated to a specific TBox describing the semantics of stored information, i.e. of tuples.
- For the TBox definition in tuple centres an **OWL-DL** ontology document [Horrocks et al., 2003].
Domain Individuals I

- Described through a so called ABox—a set of assertions about the individuals and roles, in terms of the terminology defined through the TBox.
- ABox defines two kinds of assertion: $C(a)$ and $R(b,c)$. 
Each tuple stored in a tuple centre is described as an ABox individual specifying the following information:

- name of the individual we want to describe
- concept to which the individual belongs
- set of roles in which the individual is involved
A possible SHOIN(D)-like description language for semantic tuples:

Individual ::= Iname : Descr

Descr ::= Cname | Cname (F)

F ::= Pname : V | Pname in ( Vset ) | F , F

Vset ::= V | V , Vset

V ::= Iname | N | S

Cname ::= <atom , in Prolog style >

Iname ::= <atom , in Prolog style >

Pname ::= <atom , in Prolog style >

N ::= <number >

S ::= <atom , in Prolog style >

We can obtain the following semantic tuple:

f550 : ‘Car’ (hasMaker : ferrari, hasMaxSpeed : 285, hasColour in (red, black))
Tuple templates become specifications of set of domain individuals described by the domain ontology.

⇒ A tuple template becomes a description, in TBox formalism, of the set of individuals one is interested in.

In order to describe a tuple template in a semantic way, we need a SHOIN(D)-like description language to express a tuple template as a description in the TBox formalism.
Semantic Tuple Templates II

- A possible SHOIN(D)-like description language for semantic tuple templates:

```
C ::= all | none | Cname | C , C | C ; C |
    not C | R | {Iset} | Cname ( R ) |
    ( C )
R ::= F | exists F | only F | M
F ::= P in C | P : Iname | P : D
M ::= # Msymb PosInt : P
D ::= Msymb N | = S
P ::= Pname | Pname / Binding
Msymb ::= > | < | >= | <= | =
Iset ::= Iname | Iname , Iset

Cname ::= <atom, in Prolog style >
Iname ::= <atom, in Prolog style >
Pname ::= <atom, in Prolog style >
Binding ::= <variable, in Prolog style >
N ::= <number >
PosInt ::= <positive integer number >
```

‘Car’ ; ‘Vehicle’ (hasMaker in (hasCountry : italy),
hasPrice < 15000)
Semantic Tuple Matching Mechanism

- Semantic matching mechanism amounts to look for the individuals (in the ABox) which are instances of the given concept, namely, which tuples match the semantic template.
- Description Logic reasoners can be used to perform this kind of reasoning.
In a semantic view, tuple centre primitives (in, rd, and out) represent the language whereby system components can read, consume, and write knowledge described by means of a domain ontology.

Each primitive can fail in case of non-consistency with the TBox.

⇒ Differently from the original tuple centre semantic, the out can fail in case the related tuple is not consistent with the domain ontology.
Extending TuCSoN I

- Tuple centres are provided in each TuCSoN node by a container.

- The container represents the manager of the tuple centre life-cycle and provides the API to create, access, and use them.

  ➔ The container has to provide the API to:
  - create a tuple centre associated with a specified TBox and a reasoner
  - set and obtain the TBox related to a particular tuple centre
We adopted the Pellet OWL reasoner [Sirin et al., 2007] because:

- it is easy to integrate it with TuCSoN since it is open-source and it is written in java
- it is a complete OWL-DL reasoner with good performance
- it is based on the expressive SPARQL query language

When the container creates a new tuple centre:

- by exploiting the Pellet API, a new instance of the ontology is created from a specified OWL file and provided to the tuple centre
- a new instance of the Pellet reasoner is created and provided to the tuple centre
Extending TuCSoN III

- In face of an `out` primitive:
  - the individual expressed by the received semantic tuple is interpreted
  - by exploiting the Pellet reasoner, the individual consistency with the ontology is checked
  - the individual is inserted in the ABox

- In face of a `in` or `rd` primitive:
  - the concept specification expressed by the received semantic tuple template is interpreted
  - by exploiting the Pellet reasoner the concept specification is checked against the ontology
  - the concept specification is interpreted and converted in SPARQL query
  - the first individual obtained by the reasoner is converted in a tuple
  - the obtained individual is unified with the tuple template
Besides semantic tuples, tuple templates, and tuple matching mechanism, in a semantic tuple centre it is useful to preserve the possibility of using standard syntactic tuples and templates.

⇒ No semantic tuples, tuple templates, and tuple matching mechanism are useful to realise coordination mechanisms in tuple centres.
When an operation like `out(entered_user(user1))` is executed on the `user_preferences_tc` tuple centre, for example the command `write_display(ka:‘Car’(hasMaker=ford, hasMaxSpeed=165, hasPrice=8000))` is executed.
Conclusions

- Semantic tuple centres was implemented in TuCSoN; a prototype was realised as an open source branch of TuCSoN in http://tucson.svn.sourceforge.net/viewvc/tucson/branches/.

- Future work:
  - Study the semantic tuple centre performance.
  - Test the semantic tuple centre model in pervasive computing applications.
  - Extend the semantic tuple centre model in order to support a fuzzy semantic matching.
Bibliography I

*The Description Logic Handbook: Theory, Implementation, and Applications.*
Cambridge University Press.

Semantic resource matching for pervasive environments: The approach and its evaluation.
Technical Report ECSTR-IAM08-001, Southampton, UK.

Generative communication in Linda.
*ACM Transactions on Programming Languages and Systems,* 7(1):80–112.


Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres

Elena Nardini    Mirko Viroli    Emanuele Panzavolta

ALMA MATER STUDIORUM – Università di Bologna
{elena.nardini,mirko.viroli}@unibo.it
{emanuele.panzavolta}@studio.unibo.it

SAC 2010

Sierre, Switzerland
March 26, 2010