Semantic and Fuzzy Coordination Through Programmable Tuple Spaces

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

Cesena - May 31, 2011

Outline

Referring Context

- Future and emerging software systems – like Internet-based, pervasive and self-organising systems – are mainly [Omicini and Viroli, 2011]:
  - Distributed in control, space and time
  - Open in terms of heterogeneity and dynamism
  - Knowledge-intensive—most of the activities are knowledge-based
- Interaction is one of the main sources of complexity in the software system engineering [Wegner, 1997]

→ Coordination is the activity of managing/costraining the interactions occurring dynamically among software components

→ Coordination models and languages are a set of approaches for the engineering of a system interaction space based on the following abstractions [Papadopoulos and Arbab, 1998, Busi et al., 2001]:

- Coordination media enabling interaction among
- coordinables – system components to be coordinated – by means of suitable
- coordination laws

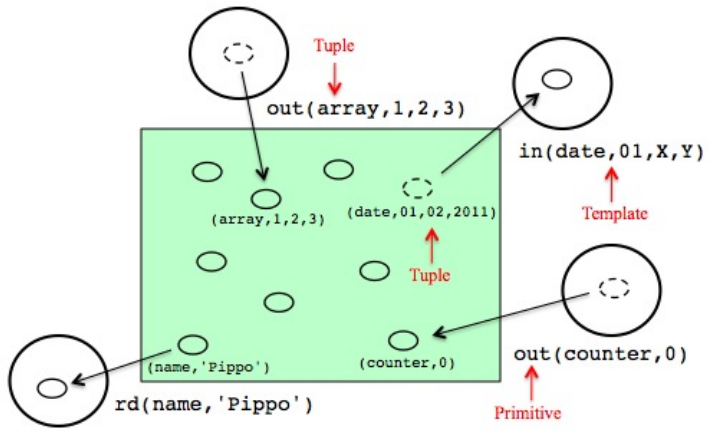
and on communication languages – syntax used to express and exchange data structures – and coordination languages—linguistic embodiements of the coordination model

The Tuple Space Model

- A growing interest in the tuple space model [Rossi et al., 2001]
  - Represents the main class of space-based coordination models [Papadopoulos and Arbab, 1998]
    * Coordination media: tuple space
    * Coordination laws: defined by the primitive semantic
    * Communication language: tuples
    * Coordination language: primitives *out*, *in* and *rd*
  - Generative communication [Gelernter, 1985]: tuples are permanently written in a tuple space
    * Promotes communication uncoupling

    → Supports openness in distributed systems
  - Associative access: in order to read/consume a tuple, a tuple set description has to be specified
    * Promotes knowledge-based coordination

    → Fits well with knowledge-intensive systems



Model Extensions [Omicini and Viroli, 2011]

- The original formulation of tuple spaces is within the LINDA model [Gelernter, 1985]
- A number of implementations and extensions have been developed and proposed in literature—e.g. Sun's JavaSpaces [Freeman et al., 1999] and GigaSpaces [GigaSpaces, 2007]
- Many other proposals have instead been focussed on extending the tuple-based coordination model beyond its original limitations. In particular, two are notable [Omicini and Viroli, 2011]:
  - Programmability of the behaviour of the tuple-based communication abstraction
  - Enhancing tuple-based communication with semantic

Programmability of the Behaviour

- The behaviour on LINDA tuple spaces is set once and for all by the model [Omicini and Denti, 2001]
  - → It cannot be tailored to the specific application needs
  - → Any coordination policy not directly supported by the coordination abstraction has to be charged upon system components
  - → Growing complexity due the coordination coupling among system coordinables [Omicini and Zambonelli, 1999]
- Several proposals towards the programmability of the tuple space behaviour:
  - Tuple centres [Denti et al., 1997] introduced the very notion of programmable tuple space
  - IBM's T Spaces [Wyckoff et al., 1998]
  - MARS [Cabri et al., 2000]
  - Law-governed Linda (LGL) [Minsky and Leichter, 1995]
  - EgoSpaces [Julien and Roman, 2006]
  - LighTS [Balzarotti et al., 2007]
  - LIME [Murphy et al., 2006]
  - TOTA [Mamei and Zambonelli, 2004]

Semantically Enriched Information

- Associative access in LINDA tuple spaces is based on a tuple matching mechanism which is purely syntactic
- As already observed in contexts like Web applications [Paolucci et al., 2002] and Pervasive Computing [Bandara et al., 2008], syntactic-driven matching leads to several limitations in dealing with open and dynamic scenarios where the exchanged information may have different syntactic structures

  Example: A system component could refers to the concept Car and in the shared spaces there is information about SportCar or CityCar concepts – that are both types of Car –, with a purely syntactic matching mechanism it is not possible to match those concepts

  → Imposes to coordinables a design-time awareness of the structure and content of tuples

  → Information semantic coupling [Nardini et al., 2010]

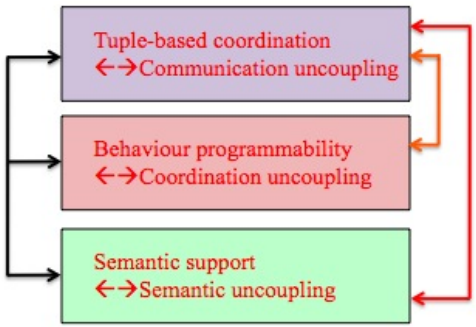
- The advent of the Semantic Web has increased the interest in the use of semantic description of information through
  - an ontology language describing application domain's *concepts*, *individuals* and *relations* among them and
  - logical reasoning over such descriptions to support information matching
- Semantic tuple space computing [Nixon et al., 2008] exploits such approaches in order to augment tuple spaces with semantic:
  - Triple Space Computing (TSC) [Fensel, 2004]
  - Conceptual Spaces (CSpaces) [Martín-Recuerda, 2005]
  - Semantic Web Spaces [Tolksdorf et al., 2008]
  - sTuples [Khushraj et al., 2004]

Research Aim

- Behaviour programmability and semantic support in tuple spaces are both essential requirements for open, distributed and knowledge-intensive systems
- However, none of the tuple-space-based approaches in literature accounts for both

→ **Research objective**:

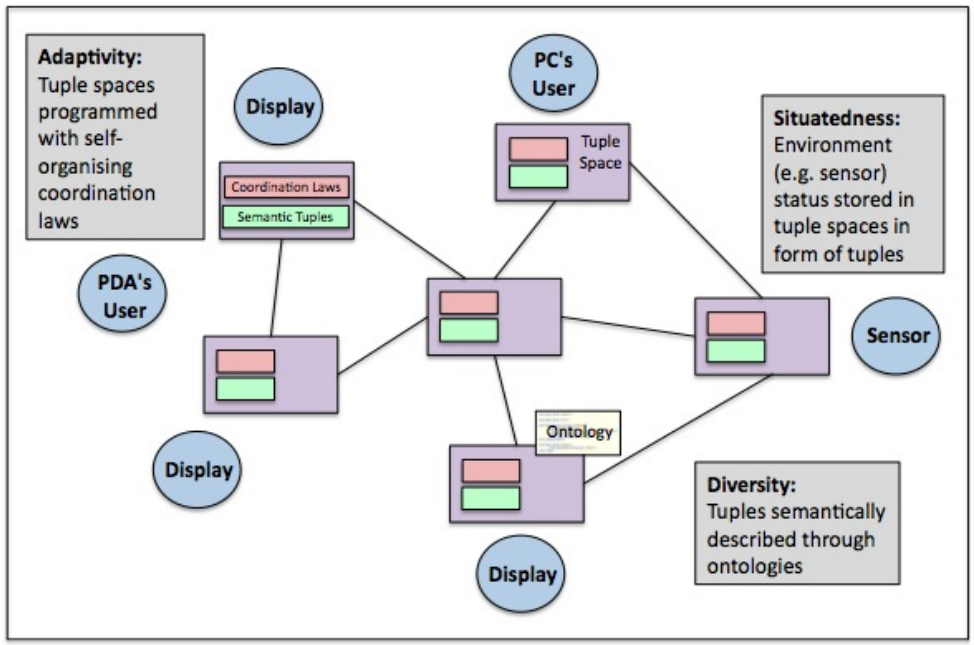
- Design and implement a new coordination abstraction including three key elements:



  supporting a semantic coordination ruled by customisable coordination laws
- No assumptions about the application context and keeping of the conceptual integrity with the original tuple-space model

A Motivating Scenario [Viroli et al., 2011]

- Pervasive service coordination
  - Increasing availability of pervasive sensing and actuating devices like RFID tags, PDAs and localisation devices
  - A new generation of general-purpose adaptive services:
    * Services to coordinate and ease customers' activity
    * Pervasive location-based information services
    * Social services exploiting contextual information
- **Requirements:** situatedness, adaptivity and diversity
- **The idea:**



Outline

Tuple-based Coordination through Tuple Centres

- Tuple centres [Omicini and Denti, 2001]: *tuple spaces whose behaviour can be determined through a specification language defining how a tuple centre should react to incoming/outcoming communication events*
  - A behaviour specification can associate any event possibly occurring in the tuple centre to a set of computational activities called reactions
  - Tuple centres are modelled as runtime runtime first-class abstraction [Ricci and Omicini, 2003] promoting the online engineering

- Tuple centres are implemented in the coordination infrastructure TuCSoN [Omicini and Zambonelli, 1999]
  - Reactions are specified with ReSpecT [Omicini, 2007]—a Turing-equivalent (→ general-purpose), logic-based specification language
  - → TuCSoN tuple centres are logic, thus making it possible to spread intelligence through the system where needed, for example by exploiting cognitive agents [Wooldridge and Jennings, 1995].

→ Tuple centres seem to provide the coordination abstraction closest to the research aim

→ **The idea**: Design the new coordination abstraction as a general-purpose semantic tuple-space
  - Starting from the tuple centre model
  - Semantically enriching the model by exploiting ontology-based techniques

and implement the new coordination model – the semantic tuple-centre model – in TuCSoN

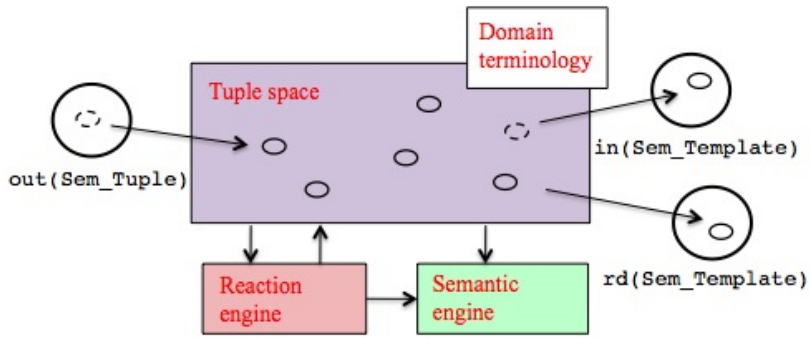
The Semantic Tuple-Centre Model

- Ontology: *a formal explicit specification of a shared conceptualisation in terms of concepts, individuals and of relations among them* [Gruber, 1995]
- Description Logic (DL) [Baader et al., 2003] is a family of formal knowledge representation languages providing a logical formalism for ontologies
  - TBox (terminological box) encoding the domain of the discourse
  - ABox (assertional box) encoding the existence of some individuals of the discourse
  - Reasoning service executing reasoning tasks on TBox and ABox: consistency checking, instance checking, instance retrieval and subsumption checking

→ From an ontological viewpoint a tuple centre has a simple and natural interpretation:
A knowledge repository structured as a set of tuples that can be sees as representing objects of the application domain whose meaning is described by an ontology

→ **Ingredients:**

(1) Domain terminology
(2) Semantic tuples
(3) Semantic templates
(4) Semantic primitives
(5) Semantic reactions
(6) Semantic matching



- Semantic tuple-centre definition through the conceptual framework of SHOIN(D) [Horrocks et al., 2003]
  - A very expressive DL [Baader et al., 2003]
  - Represents the counterpart of OWL DL—a kind of OWL, that is the *W3C* standard for the *Semantic Web* and the standard de-facto for semantic applications in general

Semantic Tuple Centres in TuCSoN—Domain Terminology

- *Describes the domain concepts and their relations attached to a tuple centre*
- Formally defined as SHOIN(D) TBox—describing the terminological axioms: concepts and their relations called roles
- OWL DL as domain terminology language for TuCSoN tuple centres

Example:

(1) $\mathtt{Maker} \sqsubseteq \top$
(2) $\mathtt{Car} \sqsubseteq (=1\ \mathtt{hasMaker})$
(3) $(=1\ \mathtt{hasMaker}) \sqsubseteq \mathtt{Car}$
(4) $\top \sqsubseteq \forall \mathtt{hasMaker}.\mathtt{Maker}$
(5) $\mathtt{CityCar} \sqsubseteq \mathtt{Car}$

```
<owl:Class rdf:ID="Maker"/>

<owl:Class rdf:ID="Car"/>

<owl:ObjectProperty rdf:ID="hasMaker">
        <rdf:type rdf:resource=
          "http://www.w3.org/2002/07/owl\#FunctionalProperty"/>
        <rdfs:domain rdf:resource="\#Car"/>
        <rdfs:range rdf:resource="\#Maker"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="CityCar">
        <rdfs:subClassOf rdf:resource="\#Car"/>
</owl:Class>
```

Semantic Tuple Centres in TuCSoN—Semantic Tuples

- *Represent domain individuals semantically interpreted by the domain terminology*
- Formally defined through a SHOIN(D) ABox—defining the axioms to assert specific domain objects (*C(a)*) and their roles (*R(a,b)*)

TuCSoN **semantic tuple grammar:**

```
Individual ::= iname ':' C
C ::= cname | cname '(' R ')'
R ::= rname ':' V | rname 'in' '(' Vset ')' | R ',' R
Vset ::= V | V ',' Vset
V ::= iname | string | int | float
```

Example:

```
(1)  Car(f550)
(2)  hasMaker(f550,ferrari)
(3)  hasMaxSpeed(f550,285)
(4)  hasColour(f550,red)
(5)  hasColour(f550,black)
```

```
f550 : 'Car' (hasMaker : ferrari, hasMaxSpeed : 285,
              hasColour in (red, black))
```

Semantic Tuple Centres in TuCSoN—Semantic Templates

- *Consist in specifications of sets of domain individuals described by the domain terminology*
- Formally defined as SHOIN(D) TBox axioms—concepts and role descriptions

TuCSoN **semantic template grammar:**

```
C ::= 'All' | 'None' | cname | C 'and' C | C 'or' C | 'not' C | D |
      '{' [ iname { ',' , iname } ] '}' | C '(' D ')' | '(' C ')' |
      'String' | 'Int' | 'Float'
D ::= F | 'exists' F | 'only' F | M
F ::= R 'in' C | R ':' I | R ':' I | R ':' Msymb N |
      R ':' 'eq' string | R
M ::= '#' R Msymb N
R ::= rname | rname '/' vname
Msymb ::= 'gt' | 'lt' | 'geq' | 'leq' | 'eq'
```

Semantic Tuple Centres in TuCSoN—Semantic Templates

Template-SHOIN(D) grammar mapping:

SHOIN(D)	*Language Expression*
$\top$	`All`
$\bot$	`None`
$C \sqcap D$	`C and D`
$C \sqcup D$	`C or D`
$\neg C$	`not C`
$\forall R.C$	`only R in C`
$\exists R.C$	`exists R in C`
$> n\ R, \leq n\ R$	`# R lt n, # R leq n`
$> n\ R, \geq n\ R$	`# R gt n, # R geq n`
$= n\ R$	`# R eq n`
$\{a_i,\ldots,a_n\}$	$\{\text{iname}_i,\ldots,\text{iname}_n\}$
concrete domains	`String, Int, Float`
concrete domain operators	`String: eq. Int, Float: eq, lt, leq, gt, geq.`

Example:

$\text{Car} \sqcap \exists \text{hasMaker.ford}$

```
'Car' and (exists hasMaker : ford)
'Car' and (exists hasMaker / X in 'Maker')
```

Semantic Tuple Centres in TuCSoN—Semantic Primitives

- *Represent the language whereby system components can read (rd), consume (in) and write (out) tuples described by the domain terminology*
- Require to revisit the semantic of the basic primitives
  - Is needed to check whether the relations and concepts associated to the individuals and concepts exist in the domain terminology
  - While an *out* in a tuple centre always succeeds, in a semantic tuple centre may fail. In particular, in face of the primitive has to be checked:
    * The individual already exists in the tuple centre
    * The consistency of the tuple to be written with the domain ontology—TBox plus ABox

    → Such kind of checks can be performed through the DL reasoner service
- The TuCSoN primitive syntax has to be extended in order to:
  - Include both semantic and syntactic primitives
  - Obtain an individual as a result from a semantic template

Example:

```
out(semantic fiat500: 'CityCar'( hasMaker : fiat))
in(semantic Result matching ('CityCar' and (exists hasMaker : fiat)))
```

Semantic Tuple Centres in TuCSoN—Semantic Reactions

- *Sets of computational activities within a tuple centre defined through a reaction specification language*
- TuCSoN exploits ReSpecT as a reaction specification language
  - A ReSpecT reaction is a *logic term* of kind `reaction(E, (G, R))`
    - `E` describing the set of communication events `Ev` for which a reaction has to be executed
    - `G` describing a set of conditions to be satisfied in order to execute a reaction
    - `R` describing the computation associated to a reaction

Semantic Tuple Centres in TuCSoN—Semantic Reactions

- In a semantic view:
  - E represents a specification of a coordination primitive related to a concept description (*in*/*rd*) or to a domain individual (*out*)
  - → E should contain a concept description expressed in terms of the semantic template language

Example:

```
reaction( in(semantic 'Car'), ..., ... )
in(semantic Result matching ('CityCar' and (exists hasMaker : fiat)))

reaction( out(semantic 'Car' and (exists hasMaker : fiat)), ..., ...)
out(semantic fiat500: 'CityCar'( hasMaker : fiat))
```

  - G represents a set of constraints on the communication event Ev
  - → G is extended so that the guard could contain a concept description expressed in terms of the semantic template language

Example:

```
reaction( out(semantic 'CityCar'), semantic (exists hasMaker : fiat), ...)
```

  - R can read, remove and write tuples from/to the tuple centre
  - → R can contain semantic primitives

Example:

```
reaction( out(semantic 'CityCar'), semantic (exists hasMaker : ford), (
          rd (semantic ford 'Maker' and (exists hasCars / N in Int)),
          N1 is N + 1,
          out(semantic ford : 'Maker'(hasCars : N1)) ))
```

Semantic Tuple Centres in TuCSoN—Semantic Matching Mechanism

- *Represents the algorithm checking the relationships between individuals and concepts described by tuples and templates or among concepts, in the execution of coordination primitives and reactions*
  - **Primitives**: When a semantic *reading/consuming* primitive is performed, the matching mechanism should identify and retrieve an individual matching the provided semantic template
  - **Reactions**: The semantic matching mechanism should work in two ways:
    * `E` describes a *writing* event: Matching `E` with `Ev` consists in checking:

      (1) if `Ev` is a writing event

      (2) if the individual in `Ev` belongs to the concept described in `E` and `G`

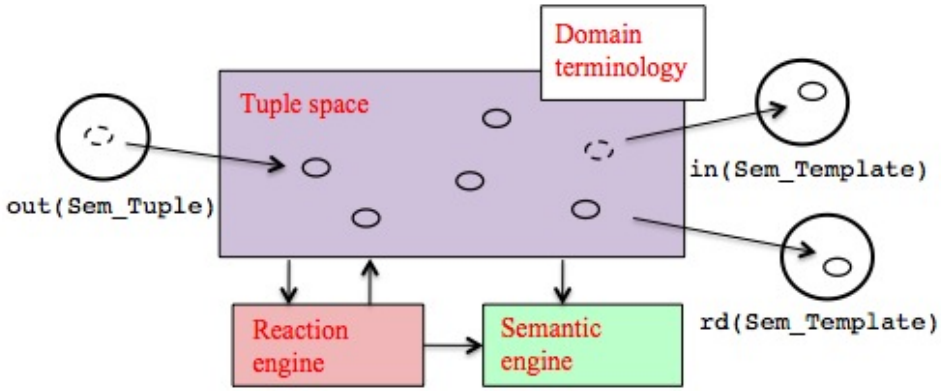
    * `E` describes a *consuming/reading* event: Matching `E` with `Ev` consists in checking:

      (1) if `Ev` is a consuming/reading event

      (2) if the concept description in `Ev` is a sub-concept of the concept in `E` and `G`

- The matching mechanism can be easily obtained by exploiting the reasoning service:
  - Instance retrieval: finds the individuals that are instance of a given concept
  - Instance checking: verifies whether a given individual is an instance of a specified concept
  - Subsumption checking: verifies the subsumption of two concepts

The Semantic Tuple-Centre Architecture



- The semantic engine provides the semantic support
  - Manages the domain terminology related to a tuple centre
  - Provides a reasoning service (internal or external) executing reasoning tasks on TBox and ABox: consistency checking, instance checking, instance retrieval and subsumption checking
  - Interacts with both the tuple space and the reaction engine providing the following operations:

    (1) Individual assertion: insert a new individual a in the ABox and *checks* if the ABox *is consistent* with the new individual (exploited for the primitive *out*)
    $\longrightarrow$ The ABox can or not coincide with the tuple space; it depends if we exploit an internal reasoner engineered specifically for the tuple space or an external DL reasoner with its own ABox representation

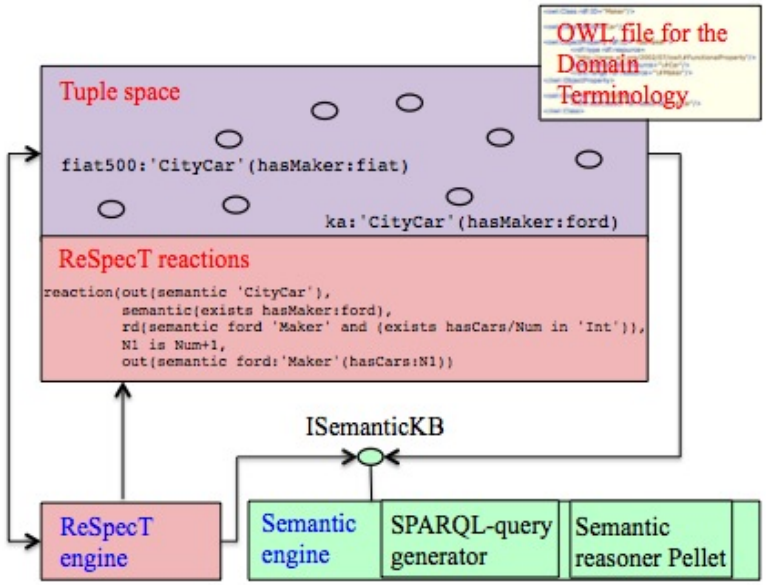
    (2) Individual deletion: deletes an individual from the ABox (exploited for the primitive *in*)

    (3) Instance checking: checks if an individual a belongs to a concept C (exploited to select semantic reactions to be activated)

    (4) Instance retrieval: retrieves all the individuals belonging to a concept C (exploited for the primitive *in*/*rd*)

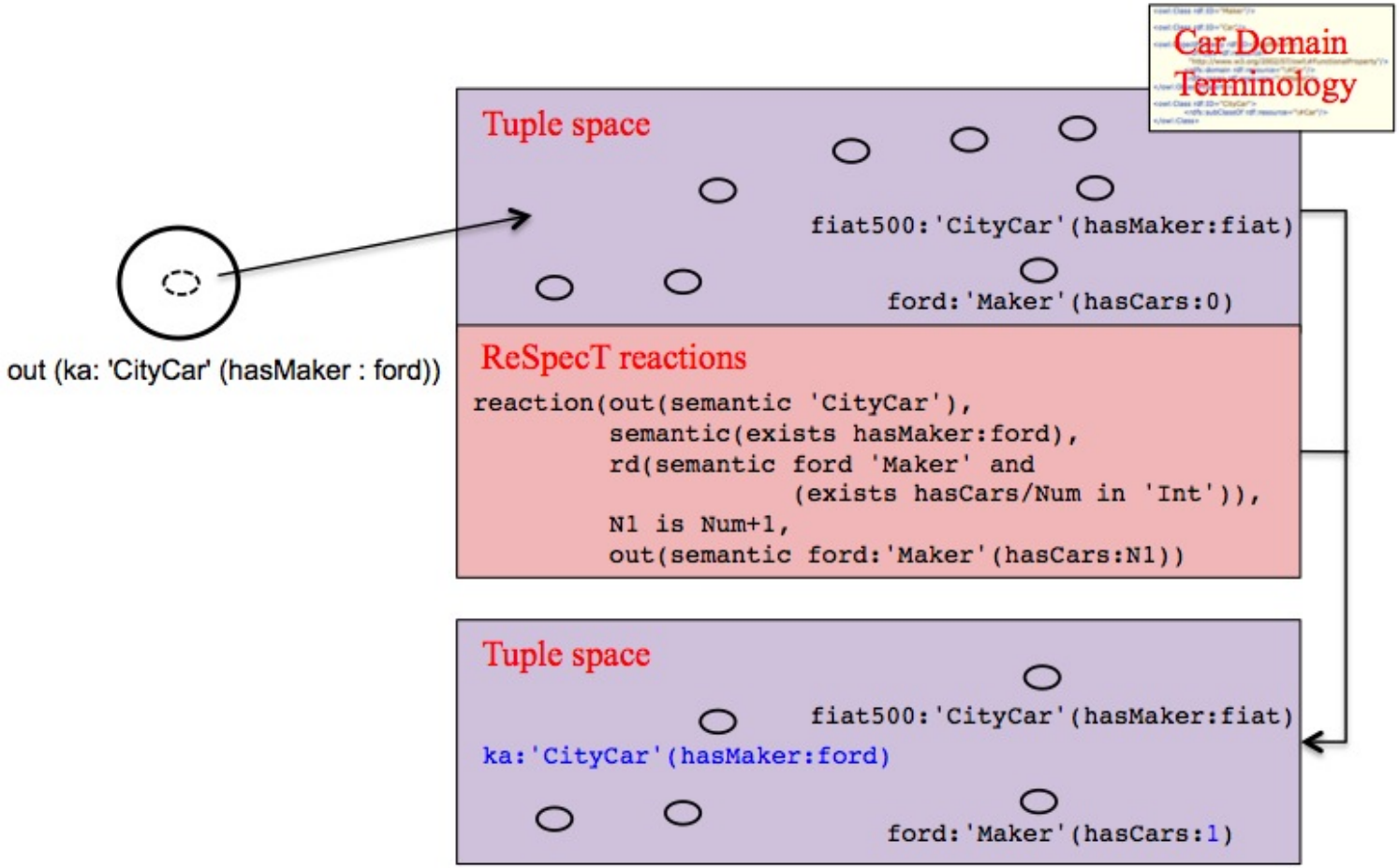
    (5) Subsumption checking: checks if a concept C subsumes a concept D (exploited to select semantic reactions to be activated)

The TuCSoN Semantic Tuple-Centre Architecture



- The semantic engine provides the semantic support through the ISemanticKB interface
  - Loads the OWL domain terminology related to a tuple centre
  - Provides the reasoning service by exploiting an external reasoner
    * The most popular DL reasoners are: RACER [Haarslev and Möller, 2001], Pellet [Sirin et al., 2007], FACT++ [Tsarkov and Horrocks, 2006], KAON2 [Motik and Sattler, 2006] and HermiT [Shearer et al., 2008]
    * TuCSoN tuple centres implement ISemanticKB via the Pellet:
      – Represents the most complete SHOIN(D) reasoner with competitive performance
      – Is Java-based like TuCSoN and ReSpecT
      – Is free and open-source
      – Supports conjunctive queries as an expressive formalism for querying DL knowledge bases through the SPARQL language [Pérez et al., 2009], that is a *W3C Candidate Recommendation* as query language for RDF, so, suitable for querying OWL ontologies

Example of Usage



Some Evaluation Results

- **Aim**: Evaluate how much the use of semantic techniques affects the tuple centre behaviour in terms of performance

→ Test the implementation of semantic tuple centres with Pellet 2.2.2 against one of the W3C reference ontologies for DL reasoner benchmarks [Bock et al., 2008]:

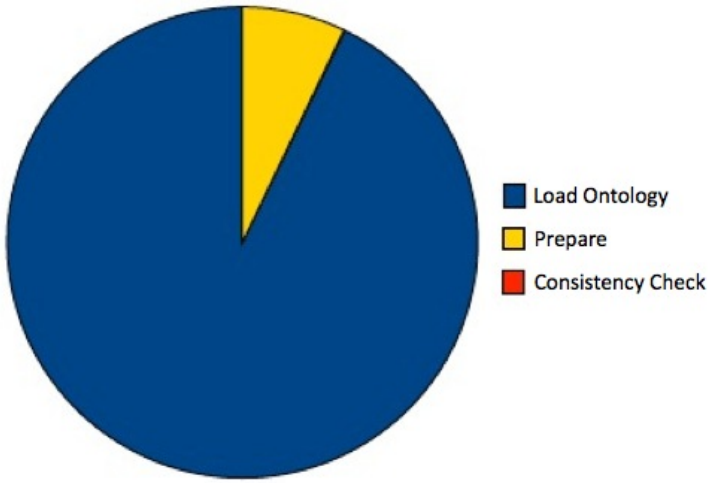
  - Lumb ontology: covers only part of the constructs supported by OWL DL (SHIQ(D)). It contains 44 classes with 36 subclass axioms and 6 equivalent class axioms and 32 roles

- The test was executed on *Intel (R) Core(TM)2 Quad CPU Q8200 2.33GHz*, equipped with *8 GB of RAM*, running on *Eclipse Helios* and *Microsoft Window Vista Home Premium*

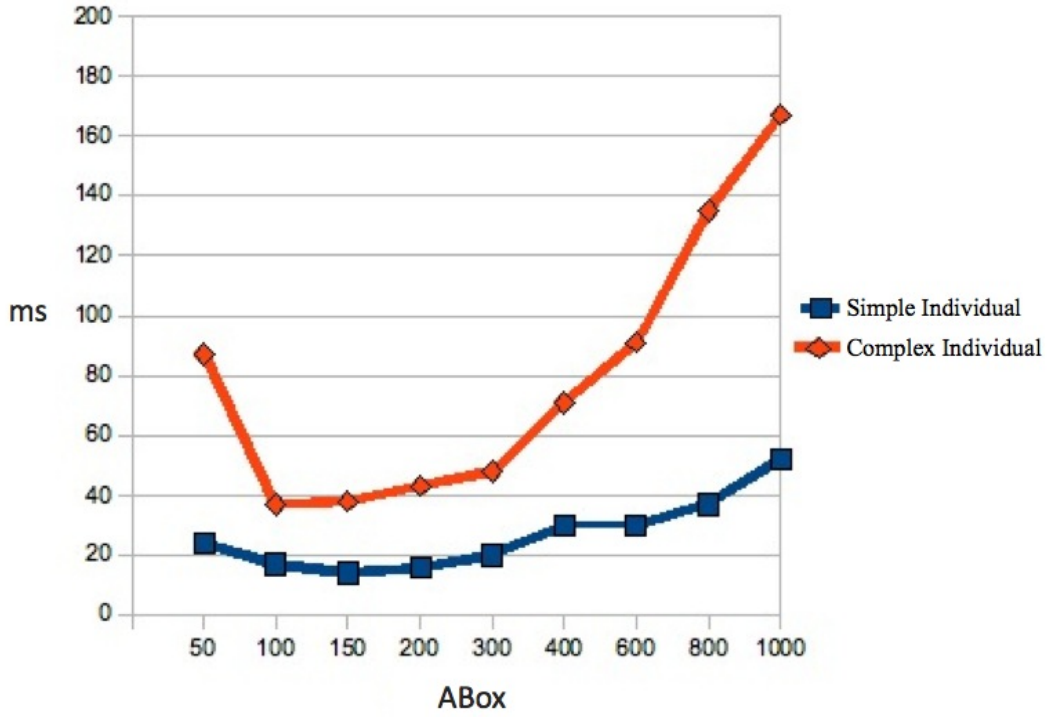
- **Test**:
  - Time to load a domain terminology
  - Time to insert a tuple
  - Time to read a tuple and to consume a tuple
  - Time to execute an instance check and to execute a subsumption check—both exploited in reaction activations

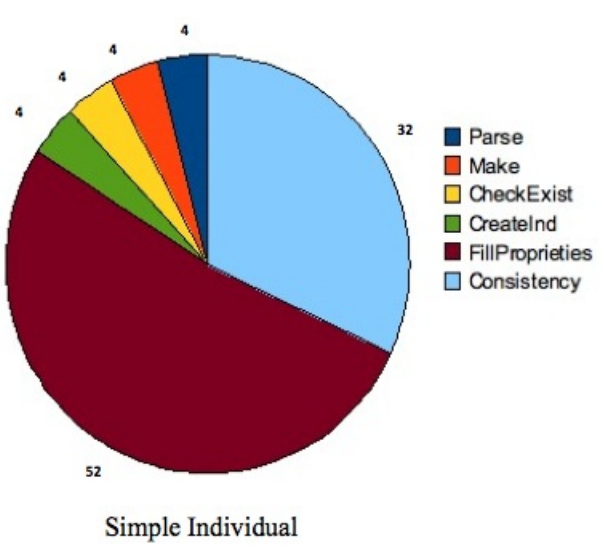
Domain Terminology Loading

- **Total time:** 1744 ms
  - Time to load the terminology in RAM
  - Terminology prepare
  - Terminology consistency check

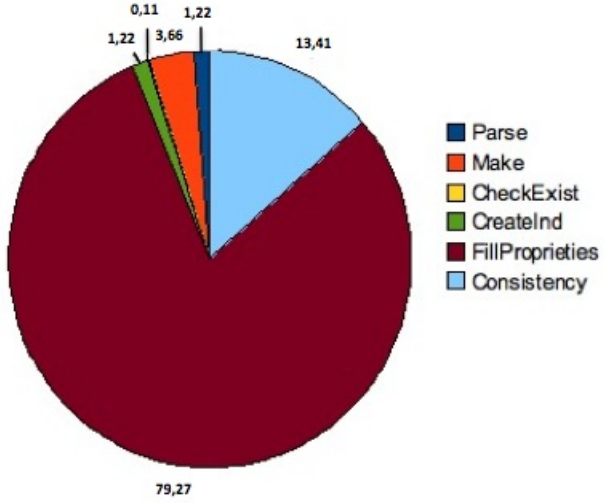


Tuple Insert



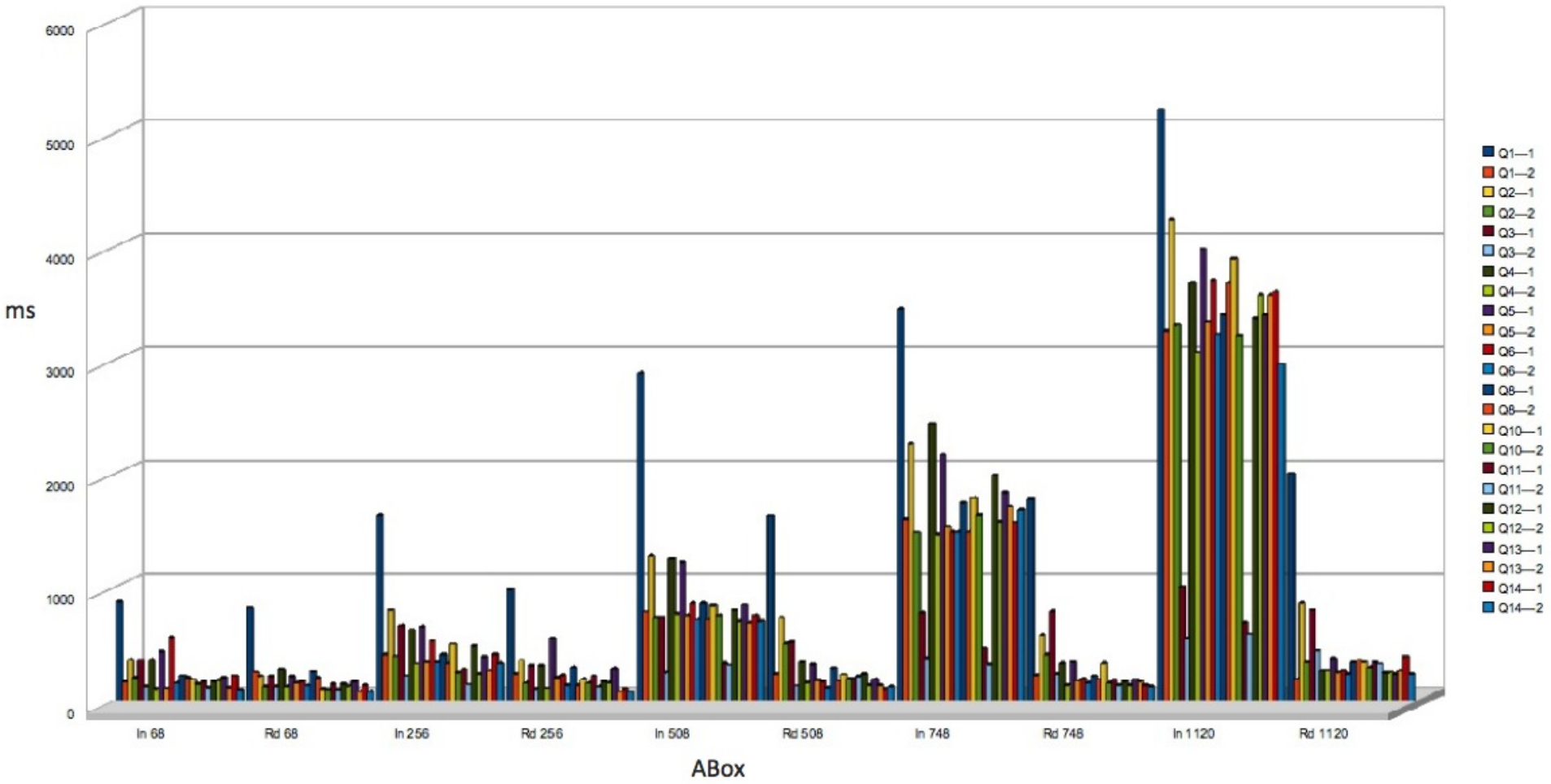


Simple Individual



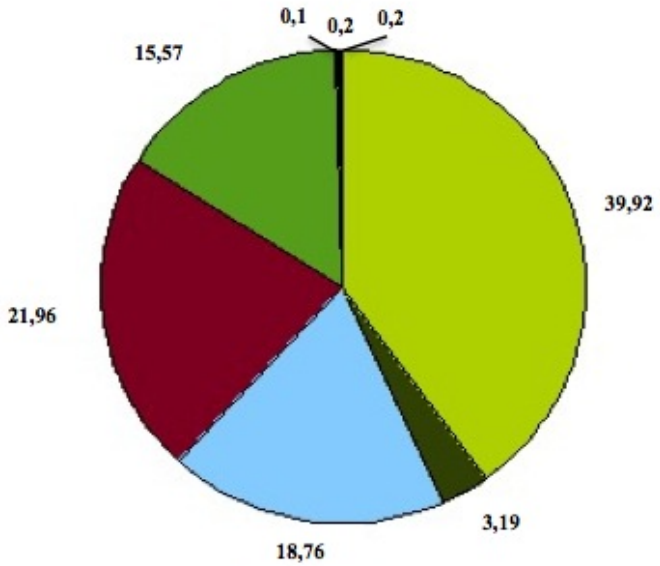
Complex Individual

Tuple Read and Consume

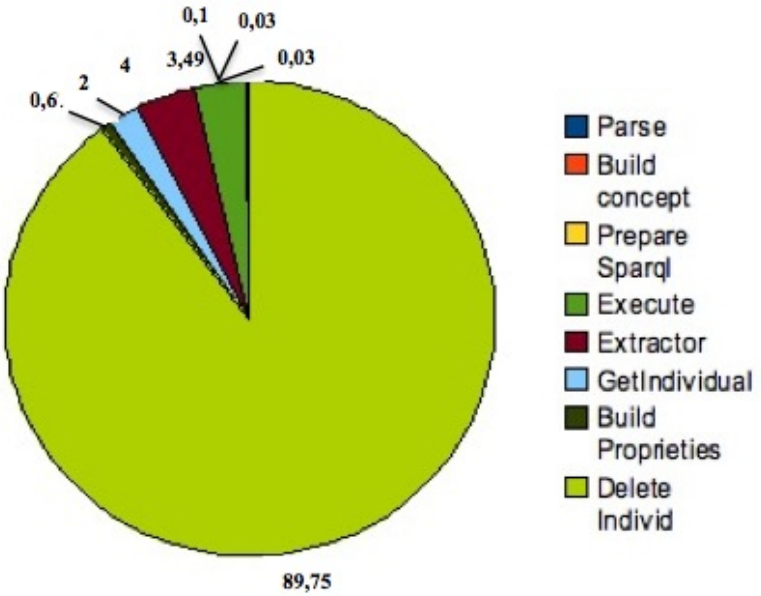


Tuple Read and Consume

- *In* with 1120 individuals in ABox

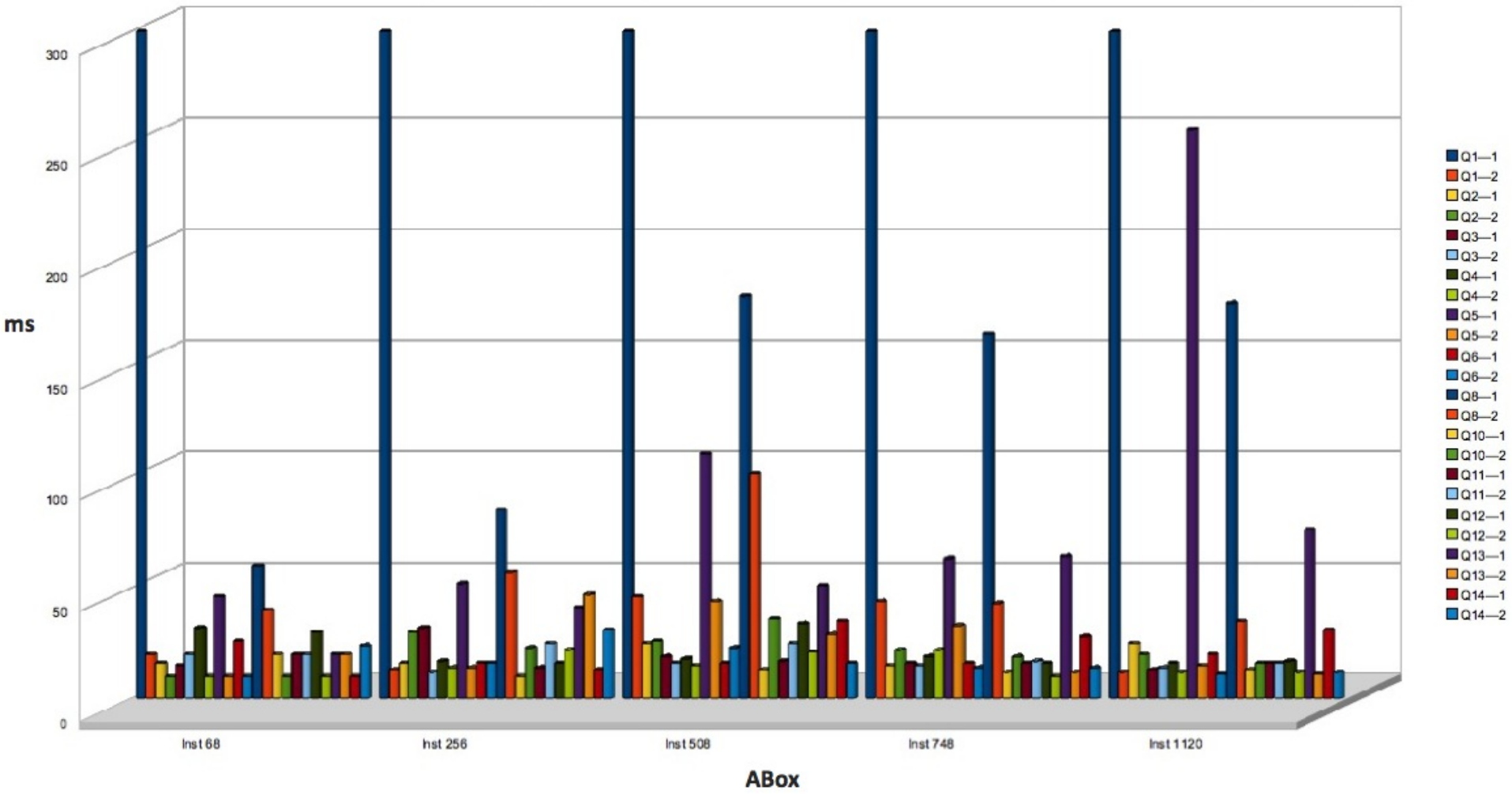


Simple Query

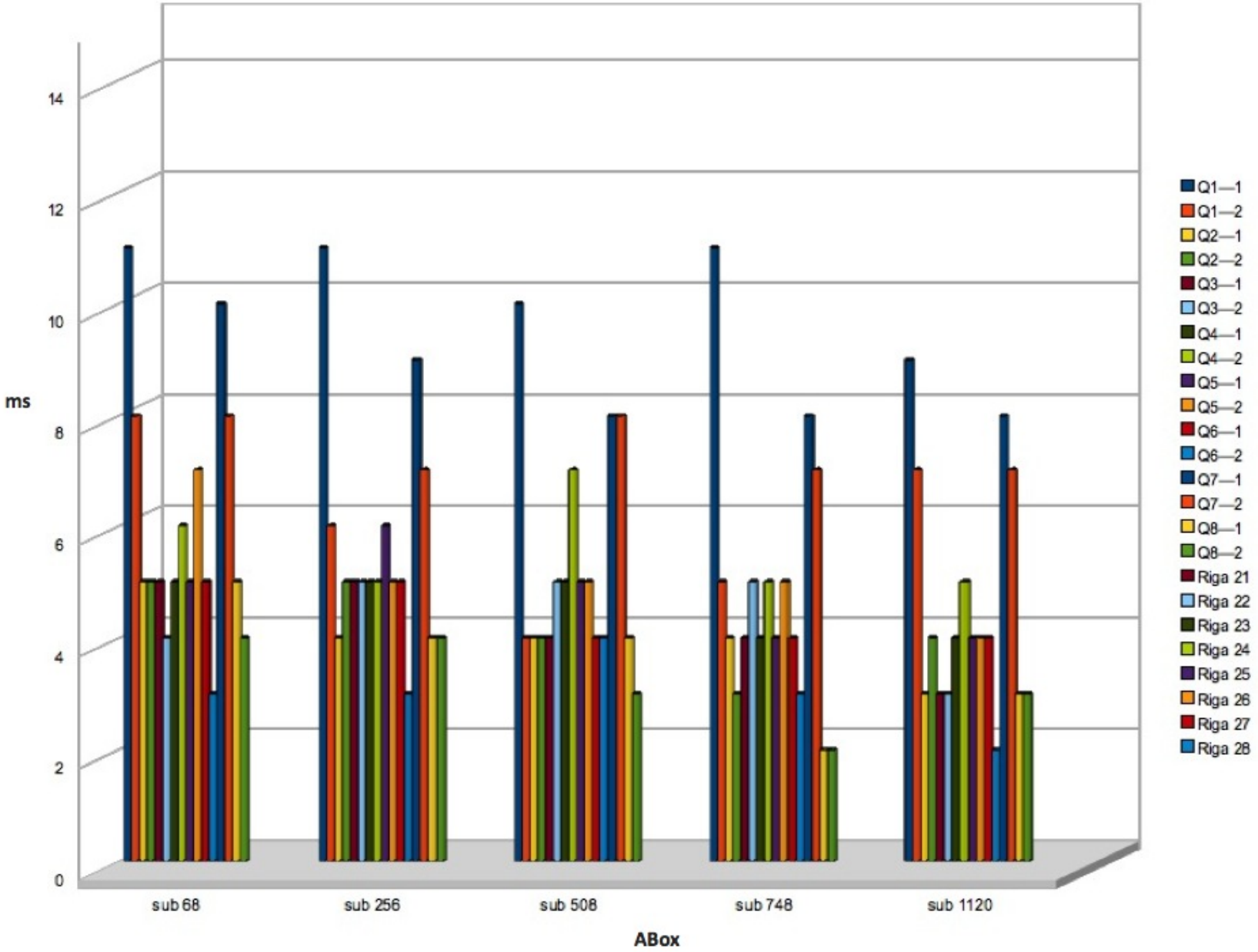


Complex Query

Instance Checking



Subsumption Checking



Other Ontologies

- `http://protege.cim3.net/file/pub/ontologies/`
- Travel ontology: 35 classes with 30 subclass axioms and 7 equivalent class axioms and 10 roles
  - *In* under 1500 ms
- Camera ontology: 35 classes with 10 subclass axioms and 3 equivalent class axioms and 15 roles
  - *In* under 500 ms

Basic Papers

- Elena Nardini, Mirko Viroli, Emanuele Panzavolta. Coordination in Open and Dynamic Environments with TuCSoN Semantic Tuple Centres. *The 25th Annual ACM Symposium on Applied Computing (SAC 2010)*, 22-26 March 2010. **The paper was selected as a best paper**.
- Elena Nardini, Andrea Omicini, Mirko Viroli. Semantic Tuple Centres. *Science of Computer Programming*. **Submitted**.

Outline

Medical-Data Sharing via Semantic Tuple Centres

- October 2009–February 2010, collaboration with Prof. Dr. Michael Ignaz Schumacher of *Institute of Business Information Systems* at the *University of Applied Sciences Western Switzerland* in Sierre, Switzerland
- A scenario in the healthcare domain:
  - Digital storage and computerised acquisition of medical data have been evolving quickly over the past decades [Van der Lei et al., 1991, Denny et al., 2005, Khon et al., 2000]
  - Communication of health data is an important factor in computerised data acquisition to overcome limits of paper-based information exchange, which is often slow [van der Kam et al., 2000] and error prone [Khon et al., 2000]
  - Among the several e-Health research activities concerning the health information exchange, research on Electronic Health Record (EHR) – medical record of a patient stored in a digital format – is particularly intensive [Khon et al., 2000, Varshney, 2009]
  - Medical data – like demographics, medical history, medication, allergy list, lab results or radiology – belonging to an EHR are called fragments, and can be distributed over diefferent EHR systems
  - The introduction of EHR offers several benefit [Malloch, 2007]:
    * Better patient safety
    * Lower cost of health services
    * Better audit and research
  - **Requirements:** In order to keep the EHR benefits, EHR systems should ensure interoperability among EHR fragments in a open and distributed context
  - **The idea:** TuCSoN semantic tuple centres in order to face with such requirements
- Collaboration financed from COST Action IC0801 `http://www.agreement-technologies.eu/`
- Project *SemHealthCoord – Semantic Health Coordination* (`http://aislab.hevs.ch/projects/semhealthcoord-semantic-health-coordination/`) accepted from the Swiss National Science Foundation (SNSF) `http://www.snf.ch/E/Pages/default.aspx`

Basic Papers

- Elena Nardini, Andrea Omicini, Mirko Viroli, Michael Ignaz Schumacher. Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres. *ACM Applied Computing Review*. **In press**.

Outline

Outline

Imprecise and Vague Knowledge

- Artificial Intelligence literature puts in evidence real-world information is often vague and imprecise [Klir and Yuan, 1994, Straccia, 2001, Zadeh, 1965]
  - In applications involving sensors, reading measurements usually come with degrees of evidence [Lukasiewicz and Straccia, 2007]
  - In applications like multimedia processing, object recognition might come with degrees of truth
  - Knowledge required to formulate precise queries is typically unavailable in open contexts [Balzarotti et al., 2007].
    - ⋆ For example, a user may request to find a cinema that is close to her, without bothering about a precise distance range

An Application Scenario

- A Virtual Bookshop is a kind of a *Virtual Enterprise* application aggregating several companies of different sorts to sell books through the Internet [Ricci et al., 2002]
- Four basic roles can be identified:
  - bookseller
  - carrier
  - interbank service
  - Internet service provider
- Two main problems
  1. Different syntactic structure of information

     Example: when a Web portal receives a request for a book of genre fantasy, whereas sellers only have books of genre classic fantasy and contemporary fantasy, a syntactic approach could not match the genres
  2. How to manage vague knowledge?

     Example: in the book domain there could be vague/imprecise information like book for kids or book for adults.
     Then, books could belong to a category with a given degree, for example, a book could be fantasy with degree **0.7**

Aim of the Paper

- Although several works discuss how to represent vague knowledge, we found only the work of Balzarotti et al. [Balzarotti et al., 2007] exploiting fuzziness to describe knowledge in tuple spaces
  - However semantic matching is not supported

→ We aim at enhancing coordination in open context where information is often not completed and precise

→ Starting from the model of the semantic tuple centres – as defined in [Nardini et al., 2010] – we devise out the elements required to equip them with fuzziness:

  - fuzzy ontologies
  - fuzzy tuples
  - fuzzy templates
  - fuzzy semantic matching
  - fuzzy primitives

→ Then, we propose a possible extension of the model toward fuzziness, in particular by referring the TuCSoN infrastructure

Outline

The Semantic Tuple Centre Model [Nardini et al., 2010]

Ingredients

- Domain ontology allowing the tuples stored to be semantically interpreted
- Semantic tuples representing individuals that can be semantically interpreted by means of the domain ontology
- Semantic tuple templates used to retrieve semantic tuple, consisting in specification of sets of domain individual described by the domain ontology
- Semantic matching mechanism providing the semantic tuples described through templates in the execution of semantic primitives
- Semantic primitives (out, in and rd) representing the language whereby system components can write, read and consume semantic tuples

Such components are formally defined through SHOIN(D)—a *Description Logic* formalism [Baader et al., 2003, Horrocks et al., 2003]

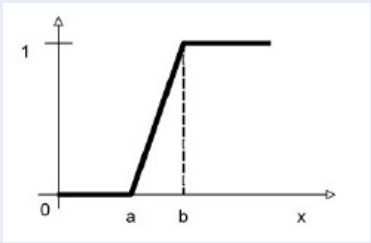
Fuzzy SHOIN(D) I

Starting from the pioneering work by Yen [Yen, 1991], Description Logics – SHOIN(D) included – were the subject of several fuzzy extensions
[Bobillo et al., 2008a, Lukasiewicz and Straccia, 2007, Straccia, 2005]

Syntactic components

- Fuzzy datatypes – fuzzy data-sets specified through functions providing membership degrees

  Example: Fuzzy data-set **High** defined as **High(x) = rs(x;80,250)**



Right-shoulder (rs) function

- Fuzzy modifiers – functions applied to fuzzy sets so as to change their membership function

  Example: Fuzzy modifiers **very** applied to **High** through the function $f_{very}(x) = x^2$ in order to obtain the data-set **very(High)**

- Fuzzy knowledge base and fuzzy axioms – composed by fuzzy TBox and fuzzy ABox. In fuzzy TBox we can define concepts like **([≥0.8] Car)** whereas in fuzzy ABox we can define individuals like **SportsCar(audi_tt) ≥ 0.92**

Fuzzy SHOIN(D) II

Semantic

- Fuzzy interpretation functions $.^I$ are introduced associating a degree in **[0,1]** to each fuzzy concept construct
- For instance, the concept constructs $\sqcap$ and $\sqcup$ have the following fuzzy interpretation functions in [Lukasiewicz and Straccia, 2007]:

$$(C_1 \sqcap C_2)^I(x) = C_1{}^I(x) \otimes C_2{}^I(x)$$
$$(C_1 \sqcup C_2)^I(x) = C_1{}^I(x) \oplus C_2{}^I(x)$$

where the operators $\otimes$ and $\oplus$ are interpreted according to one of the following logics:

  - Lukasiewicz Logic
  - Gödel Logic
  - Product Logic
  - Zadeh Logic

Example: Gödel Logic defines $\alpha \oplus_G \beta$ such as $\max\{\alpha, \beta\}$, whereas Lukasiewicz Logic defines $\alpha \oplus_L \beta$ such as $\min\{\alpha+\beta, 1\}$

Extending Tuple Centres for Supporting Fuzziness

- Following semantic tuple centres components, the ingredients required to extend them toward fuzziness are:
  - fuzzy ontologies – from crisp to fuzzy TBox
  - fuzzy tuples – from crisp to fuzzy ABox
  - fuzzy templates – from crisp to fuzzy concepts described in the fuzzy-TBox formalism
  - fuzzy semantic matching – from crisp to fuzzy reasoning upon fuzzy and semantic knowledge by exploiting fuzzy SHOIN(D) reasoners
  - fuzzy primitives – from individuals retrieved with degree **1** to individuals retrieved with degree in **[0,1]**
- In the following we describe how it is possible to extend with fuzziness the ReSpecT semantic tuple centres provided by the infrastructure TuCSoN

Fuzzy Ontology

- The domain ontology associated to a tuple centre is formally described as a SHOIN(D) TBox [Baader et al., 2003]

→ Fuzzy ontologies described through fuzzy SHOIN(D) TBoxes

Example: We can define the following concept assertions:

$$\texttt{High} \equiv \texttt{rs(80,250)}$$
$$\texttt{SportsCar} \equiv \texttt{Car} \sqcap \exists \texttt{hasSpeed.very(High)}$$

- For ReSpecT tuple centres there are the following possibilities:
  1. exploit fuzzy OWL DL
     → they are not standard
  2. exploit crisp OWL DL ontologies describing fuzzy ontologies [Bobillo et al., 2008b]
     → very large ontologies with performance problems
  3. exploit crisp OWL DL ontologies
     → it is only possible to describe crisp ontologies

Fuzzy Tuples

- Semantic tuples are formally described as SHOIN(D) individuals belonging to an ABox [Baader et al., 2003]

→ Fuzzy tuples described through fuzzy SHOIN(D) individuals belonging to a fuzzy ABox

Example: We can define the following fuzzy individuals:

$$\texttt{SportsCar(audi\_tt)} \geq 0.92$$
$$\texttt{hasMaker(audi\_tt, audi)} \geq 1$$

- The language for ReSpecT fuzzy tuples becomes as follows

```
Individual ::= iname ':' C

C ::= cname [→ α] | cname '(' R [→ α] ')'

R ::= rname ':' V | rname 'in' '(' Vset ')' | R ',' R

Vset ::= V [→ α] | V [→ α] ',' Vset

V ::= iname | number | string
```

- We could obtain the following fuzzy tuples

```
ca:'CityCar'(hasMaker:ford, hasMaxSpeed:130,
             hasColour in (red → 0.7 , black → 0.3 ))

audiTT:'SportCar' → 0.8 (hasMaker:audi, hasMaxSpeed : 260,
                         hasColour : black)
```

Fuzzy Templates I

- Semantic templates are formally described as SHOIN(D) concepts in TBox formalism [Baader et al., 2003]

→ Fuzzy templates described as fuzzy SHOIN(D) concepts in fuzzy TBox formalism

Example: We can define the following fuzzy concept:

$$[\geq 0.8]\ \texttt{SportCar}$$

Fuzzy Templates II

- The language for ReSpecT fuzzy templates becomes as follows

```
C ::= '$ALL' | '$NONE' | cname | C ',' C |
      C ';' C | '$not' C | D |
      '{' [ iname [→ α] { ',' , iname [→ α] } ] '}' |
      C '(' D ')' | '(' C ')' |
      CW | C'['≥ α']' | C'['≤ α']' |
      DT | M '(' C ')'

CW ::= C→ α | CW + CW

DT ::= crisp'('N','N')' | l'('N','N')' |
       r'('N','N')' | M

M ::= triangular'('N','N','N')' |
      trapezoidal'('N','N','N','N')'

D ::= F | '$exists' F | '$all' F | M

F ::= R 'in' C | R ':' I | R ':' Msymb N |
      R ':=' string | R

M ::= '#' R Msymb N

R ::= rname | rname '/' vname

Msymb ::= '>' | '<' | '≥' | '≤' | '='
```

- We could obtain the following fuzzy templates

```
'Car', ((hasMaxSpeed:280→ 0.6) +
        (hasMaker:ferrari→ 0.4))

[≥ 0.8] 'SportCar'

'Car', ($exists hasSpeed in linear(0.8)(r(80,250)) )
```

Fuzzy Matching Mechanism I

- The semantic matching mechanism is realised by exploiting SHOIN(D) reasoner services allowing ABox to be queried through TBox-based descriptions
- In order to support matching between fuzzy tuples and templates there are three different ways
  1. exploiting fuzzy reasoners with fuzzy ontologies
     - ★ fuzzy SHOIN(D) reasoners do not exist
  2. exploiting crisp reasoners with crisp ontologies representing fuzzy ontologies
     - ★ to best of our knowledge, the only reasoner supporting fuzzy SHOIN(D) is DeLorean [Bobillo et al., 2008b]
     - ★ it uses crisp domain ontologies to represent fuzzy ontologies reducing the reasoning within fuzzy SHOIN(D) to reasoning within crisp SHOIN(D)
     - → it is possible to exploit existent crisp reasoners like Pellet [Sirin et al., 2007]
     - ★ the complete version is not yet available
     - ★ there are performance problems caused by the huge ontologies (5 sec for a complex query with 44 classes and 25 properties)

Fuzzy Matching Mechanism II

3 exploiting crisp reasoners without fuzzy ontologies

It is possible to realise a *(de)fuzzificator* in charge of bridging between the tuple centre interface and a crisp SHOIN(D) reasoner

- ★ it transforms a fuzzy individual description in a crisp one when the corresponding tuple is inserted in the tuple centre
- ★ the crisp individual is stored in the crisp SHOIN(D) reasoner whereas the corresponding fuzzy individual is stored in the tuple centre
- ★ in face of a reading or consuming operation:

1. it interprets the fuzzy template as a crisp template to query the crisp reasoner
2. it retrieves an individual with the degree with which it matches the fuzzy template, by exploiting the fuzzy individual version stored in the tuple centre

→ It is needed to store some information about fuzzy tuples and an algorithm to process fuzzy templates, but ontologies are smaller

Fuzzy Matching Mechanism III

- Example of such a matching mechanism in ReSpecT

  1 When a new tuple is inserted in the tuple in the tuple centre, the (de)fuzzificator should separate the degrees associated to the tuple in order to obtain a non-fuzzy individual-assertion description in the ReSpecT language for semantic tuples.

  The degrees remain stored only in the tuple inserted in the tuple centre

  Example: We can have the two following fuzzy tuples to be inserted in the tuple centre:

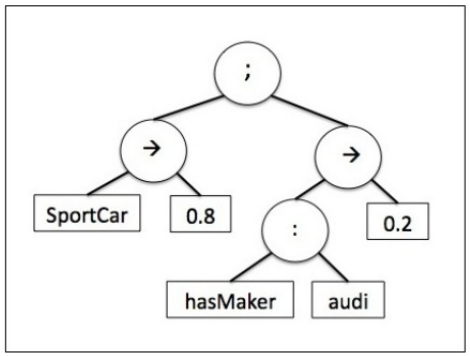
```
audiTT : 'SportCar'→ 0.8(hasMaker : audi, hasMaxSpeed : 260,
                         hasColour : black)

f380 : 'SportCar'→ 0.9(hasMaker : ferrari, hasMaxSpeed : 320,
                       hasColour : red)
```

Fuzzy Matching Mechanism IV

2 In face of fuzzy template like the following

`'SportCar'→0.8; (hasMaker:audi)→0.2`

the following abstract syntax tree can be exploited



in order to:

1. query the DL reasoner for individuals belonging to the concept `SportCar`
2. calculate for each of them its degree as the product of 0.8 by the degree as specified in the tuple centres
3. query the DL reasoner for individuals belonging to the concept `hasMaker:audi` by querying the DL reasoner
4. calculate for each of them its degree as the product of 0.2 by the degree as specified in the tuple space
5. execute the fuzzy operator ; (the counterpart of SHOIN(D) operator $\sqcup$) over the two above sets of individuals. By using e.g. the implementation based on *Lukasiewicz t-conorm*, we join the two sets and update degrees of each tuple by formula $\min\{\alpha+\beta,1\}$), where α and β are the two originatind degrees—0 if the tuple was not in the set

Fuzzy Matching Mechanism V

- Accordingly, supposing to have the following tuples:

```
audiTT : 'SportCar'→ 0.8(hasMaker : audi, hasMaxSpeed : 260,
                         hasColour : black)

f380 : 'SportCar'→ 0.9(hasMaker : ferrari, hasMaxSpeed : 320,
                       hasColour : red)
```

and the following template

```
'SportCar'→0.8; (hasMaker:audi)→0.2
```

matching the fuzzy tuple template 'SportCar'→0.8; (hasMaker:audi)→0.2 returns tuple audiTT with degree 0.84 and f380 with degree 0.72

Fuzzy Primitives

- The semantic tuple space primitives – in, rd and out – represent the coordination language whereby system components can read, consume and write knowledge described by means of a domain ontology [Nardini et al., 2010]
- Extending the tuple centre model with fuzziness requires some extensions in primitives

1. out is unchanged

   Example:

```
out(semantic audiTT : 'SportCar'→ 0.8(hasMaker : audi,
                                      hasMaxSpeed : 260,
                                      hasColour : black))
```

2. in and rd change

   In face of a fuzzy template, to return a fuzzy tuple along with the degree by which it satisfies the template, since this could be different from 1

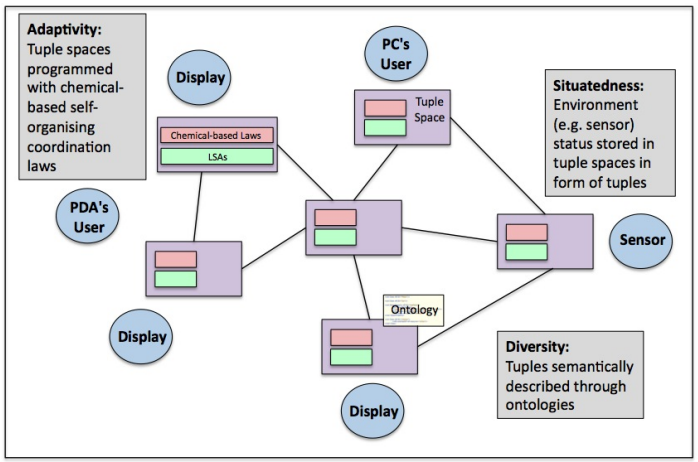
   Example:

```
in(semantic (X degree Y matching ([≥ 0.8] 'SportCar')))
```

Outline

SAPERE Project

- Self-Aware Pervasive Service Ecosystems
  EU STREP Project (October 2010-October 2013)
  FP7-ICT-2009.8.5: Self-awareness in Autonomic Systems
- **Home page:** `http://www.sapere-project.eu/`
- **Main aim:** Development of a highly-innovative theoretical and practical framework for the decentralized deployment and execution of self-aware and adaptive services for future and emerging pervasive network scenarios
- **Early research ideas:**
  - Tuple spaces + chemical reactions as coordination laws [Viroli and Casadei, 2009]
  - Tuples have a concentration (a.k.a. weight, or activity value)
  - Concentration is evolved exactly as in chemistry
  - Some reactions can even fire a tuple from one space to another



  - Match between chemical laws and reactants:
    – Semantic matching in order to deal with openness requirements
    – Fuzzy matching in order to deal with vagueness of information [Lukasiewicz and Straccia, 2007]

Basic Papers

- Mirko Viroli, Matteo Casadei, Elena Nardini, Andrea Omicini. Towards a Chemical-Inspired Infrastructure for Self-* Pervasive Applications *Self-Organizing Architectures, Lecture Notes in Computer Science 6090*, July 2010.
- Elena Nardini, Mirko Viroli, Matteo Casadei, Andrea Omicini. A Self-Organising Infrastructure for Chemical-Semantic Coordination: Experiments in TuCSoN. *WOA 2010—Dagli oggetti agli agenti. Modelli e tecnologie per sistemi complessi: context-dependent, knowledge-intensive, nature-inspired e self-*, CEUR Workshop Proceedings 621*, 5-7 September 2010.
- Elena Nardini, Andrea Omicini, Mirko Viroli. Description Spaces with Fuzziness. *The 26th Annual ACM Symposium on Applied Computing (SAC 2011)*, 21-25 March 2011. **In press**.
- Elena Nardini, Mirko Viroli, Gabriella Castelli, Marco Mamei, Franco Zambonelli. A Coordination Approach to Spatially-Situated Pervasive Service Ecosystem. **Sumbitted**.

Outline

Agent Oriented Software Engineering (AOSE) and Agent-Oriented Methodologies

- Elena Nardini, Ambra Molesini, Andrea Omicini, Enrico Denti. SPEM on Test: the SODA Case Study. *23th ACM Symposium on Applied Computing (SAC 2008)*,6-20 March 2008.
- Ambra Molesini, Elena Nardini, Enrico Denti, Andrea Omicini. Advancing Object-Oriented Standards Toward Agent-Oriented Methodologies: SPEM 2.0 on SODA. *9th Workshop "From Objects to Agents" (WOA 2008) volution of Agent Development: Methodologies, Tools, Platforms and Languages*, November 2008.
- Ambra Molesini, Elena Nardini, Enrico Denti, Andrea Omicini. Situated Process Engineering for Integrating Processes from Methodologies to Infrastructures. *24th Annual ACM Symposium on Applied Computing (SAC 2009)*, 8-12 March 2009.
- Ambra Molesini, Marco Prandini, Elena Nardini, Enrico Denti. Risk Analysis and Deployment Security Issues in a Multi-Agent System. *The 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010), 2010*, 22-24 January 2010.

Computer Supported Collaborative Learning

- Elena Nardini, Andrea Omicini, Maria Cristina Matteucci. Toward a Framework for Collaborative Learning based on Agent-based Technologies. *The 2008 International Education, Technology and Development Conference (INTED 2008)*,-5 March 2008.
- Elena Nardini, Matteo Casadei, Andrea Omicini, Pietro Gaffuri. A Conceptual Framework for Collaborative Learning Systems Based on Agent Technologies. *The 2008 International Conference on the Interactive Computer Aided Learning (ICL 2008)*, 24-26 September 2008.
- Elena Nardini, Andrea Omicini. Agent-Based Collaboration Systems: A Case Study. *Knowledge Construction in E-learning Context: CSCL, ODL, ICT and SNA in Education, CEUR Workshop Proceedings 398*, October 2008.
- Maria Cristina Matteucci, Andrea Omicini, Elena Nardini, Pietro Gaffuri. Knowledge Construction in E-learning Context: CSCL, ODL, ICT and SNA in Education. *CEUR Workshop Proceedings 398*, 1-2 September 2008.

Agent Coordination Context

- Elena Nardini, Andrea Omicini, Mirko Viroli. General-Purpose Coordination Abstractions for Managing Interaction in MAS. *The WI-IAT 2009 Workshops Proceedings*, 15-18 September 2009.

Conclusion

- Model and implementation of the semantic tuple centre model—a basic brick of coordination infrastructures for open, distributed and knowledge-intensive systems
- Description of the model, the architecture and the implementation in TuCSoN and evaluation of performance
- Application scenarios in which the semantic tuple centre model seems to be suitable to engineer the software system coordination space
- Ongoing work:
  - Exploit the experience with the semantic tuple centre model in the context of the e-Health and SAPERE project
  - Evaluate other semantic reasoners (external or internal) that can trade-off speed for expressiveness
  - From the model viewpoint, evaluating of a basic extension concerns the introduction of fuzzyness, relying on approaches like fuzzyDL [Bobillo and Straccia, 2008]

Outline

Bibliography I

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003).

The Description Logic Handbook: Theory, Implementation, and Applications.

Cambridge University Press.

Balzarotti, D., Costa, P., and Picco, G. P. (2007).

The LighTS tuple space framework and its customization for context-aware applications.

Web intelligence and Agent Systems, 5(2):215–231.

Bandara, A., Payne, T., Roure, D. D., Gibbins, N., and Lewis, T. (2008).

Semantic resource matching for pervasive environments: The approach and its evaluation.

Technical Report ECSTR-IAM08-001, Southampton, UK.

Bibliography II

Bobillo, F., Delgado, M., and Gómez-Romero, J. (2008a).

A crisp representation for fuzzy SHOIN with fuzzy nominals and general concept inclusions.

In da Costa, P. C. G., d'Amato, C., Fanizzi, N., Laskey, K. B., Laskey, K. J., Lukasiewicz, T., Nickles, M., and Pool, M., editors, *Uncertainty Reasoning for the Semantic Web I*, volume 5327 of *LNCS*, pages 174–178. Springer.

Bobillo, F., Delgado, M., and Gomez-Romero, J. (2008b).

Delorean: A reasoner for fuzzy OWL 1.1.

In Bobillo, F., da Costa, P. C. G., d'Amato, C., Fanizzi, N., Laskey, K. B., Laskey, K. J., Lukasiewicz, T., Martin, T. P., Nickles, M., Pool, M., and Smrz, P., editors, *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008)*, volume 423 of *CEUR Workshop Proceedings*, Karlsruhe, Germany. Sun SITE Central Europe, RWTH Aachen University.

Bobillo, F. and Straccia, U. (2008).

FuzzyDL: An expressive fuzzy description logic reasoner.

In *2008 International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 923–930. IEEE CS.

Bibliography III

Bock, J., Haase, P., Ji, Q., and Volz, R. (2008).

Benchmarking OWL reasoners.

In van Harmelen, F., Herzig, A., Hitzler, P., Lin, Z., Piskac, R., and Qi, G., editors, *Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea2008)*, volume 350 of *CEUR Workshop Proceedings*, ESWC 2008, Tenerife, Spain.

Busi, N., Ciancarini, P., Gorrieri, R., and Zavattaro, G. (2001).

Coordination models: A guided tour.

chapter 1, pages 6–24.

Cabri, G., Leonardi, L., and Zambonelli, F. (2000).

MARS: A programmable coordination architecture for mobile agents.

IEEE Internet Computing, 4(4):26–35.

Denny, J., Giuse, D., and Jirjis, J. (2005).

The Vanderbilt Experience with Electronic Health Records.

Seminars in Colon and Rectal Surgery, 12:59–68.

Bibliography IV

Denti, E., Natali, A., and Omicini, A. (1997).

Programmable coordination media.

In Garlan, D. and Le Métayer, D., editors, *Coordination Languages and Models*, volume 1282 of *LNCS*, pages 274–288. Springer-Verlag.

2nd International Conference (COORDINATION'97), Berlin, Germany, 1–3 September 1997. Proceedings.

Fensel, D. (2004).

Triple-space computing: Semantic web services based on persistent publication of information.

In Aagesen, F. A., Anutariya, C., and Wuwongse, V., editors, *Intelligence in Communication Systems*, volume 3283 of *LNCS*, pages 43–53.

IFIP International Conference (INTELLCOMM 2004), Bangkok, Thailand, 23–26 November 2004. Proceedings.

Freeman, E., Hupfer, S., and Arnold, K. (1999).

JavaSpaces Principles, Patterns, and Practice: Principles, Patterns and Practices.

The Jini Technology Series. Addison-Wesley Longman.

Bibliography V

Gelernter, D. (1985).

Generative communication in Linda.

ACM Transactions on Programming Languages and Systems, 7(1):80–112.

GigaSpaces (2007).

Home page.

Gruber, T. R. (1995).

Toward principles for the design of ontologies used for knowledge sharing.

Int. J. Hum.-Comput. Stud., 43:907–928.

Haarslev, V. and Möller, R. (2001).

RACER system description.

In Goré, R., Leitsch, A., and Nipkow, T., editors, *1st International Joint Conference on Automated Reasoning (IJCAR '01)*, volume 2083 of *LNCS*, pages 701–705. Springer, Siena, Italy.

Horrocks, I., Patel-Schneider, P. F., and Harmelen, F. V. (2003).

From shiq and rdf to owl: The making of a web ontology language.

Journal of Web Semantics, 1:2003.

Bibliography VI

Julien, C. and Roman, G.-C. (2006).

EgoSpaces: Facilitating rapid development of context-aware mobile applications.

IEEE Transactions on Software Engineering, 32(5):281–298.

Khon, L., Corrigan, J., and Donaldson, M. (2000).

To Err is Human: building a safer health system.

National Academy Press.

Khushraj, D., Lassila, O., and Finin, T. W. (2004).

sTuples: Semantic tuple spaces.

In Finin, T. W., Ghidini, C., La Porta, T., and Petrioli, C., editors, *1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, pages 268–277, Boston, MA, USA.

Klir, G. and Yuan, B. (1994).

Fuzzy sets and fuzzy logic: theory and applications.

Prentice-Hall, Inc.

Lukasiewicz, T. and Straccia, U. (2007).

Managing uncertainty and vagueness in description logics for the semantic web.

Bibliography VII

Malloch, K. (2007).

The electronic health record: An essential tool for advancing patient safety.

Nursing Outlook, 55:150–161.

Mamei, M. and Zambonelli, F. (2004).

Programming pervasive and mobile computing applications with the TOTA middleware.

In *Pervasive Computing and Communications*, pages 263–273.

2nd IEEE Annual Conference (PerCom 2004), Orlando, FL, USA, 14–17 March 2004. Proceedings.

Martín-Recuerda, F. (2005).

Towards Cspaces: A new perspective for the Semantic Web.

In Bramer, M. and Terziyan, V., editors, *Industrial Applications of Semantic Web*, volume 188, pages 113–139. Springer.

1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web, 25–27 August 2005, Jyväskylä, Finland. Proceedings.

Bibliography VIII

Minsky, N. H. and Leichter, J. (1995).

Law-Governed Linda as a coordination model.

In Ciancarini, P., Nierstrasz, O., and Yonezawa, A., editors, *Object-based Models and Languages for Concurrent Systems*, volume 924 of *LNCS*, pages 125–146. Springer.

Motik, B. and Sattler, U. (2006).

A comparison of reasoning techniques for querying large Description Logic ABoxes.

In Hermann, A. and Voronkov, A., editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 4246 of *LNCS*, pages 227–241. Springer.

13th International Conference (LPAR 2006), Phnom Penh, Cambodia, 13-17 November, 2006. Proceedings.

Murphy, A. L., Picco, G. P., and Roman, G.-C. (2006).

LIME: A coordination model and middleware supporting mobility of hosts and agents.

ACM Transactions on Software Engineering and Methodology, 15(3):279–328.

Bibliography IX

Nardini, E., Viroli, M., and Panzavolta, E. (2010).

Coordination in open and dynamic environments with tucson semantic tuple centres.

In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, Sierre, Switzerland. ACM.

Nixon, L. j. b., Simperl, E., Krummenacher, R., and Martin-recuerda, F. (2008).

Tuplespace-based computing for the semantic web: A survey of the state-of-the-art.

Knowl. Eng. Rev., 23(2):181–212.

Omicini, A. (2007).

Formal ReSpecT in the A&A perspective.

Electronic Notes in Theoretical Computer Sciences, 175(2):97–117.

5th International Workshop on Foundations of Coordination Languages and Software Architectures (FOCLASA'06), CONCUR'06, Bonn, Germany, 31 August 2006. Post-proceedings.

Bibliography X

Omicini, A. and Denti, E. (2001).

From tuple spaces to tuple centres.

Science of Computer Programming, 41(3):277–294.

Omicini, A. and Viroli, M. (2011).

Coordination models and languages: From parallel computing to self-organisation.

The Knowledge Engineering Review, 26(1):53–59.

Special Issue for the 25th Years of the Knowledge Engineering Review.

Omicini, A. and Zambonelli, F. (1999).

Coordination for Internet application development.

Autonomous Agents and Multi-Agent Systems, 2(3):251–269.

Special Issue: Coordination Mechanisms for Web Agents.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P. (2002).

Semantic matching of web services capabilities.

In *ISWC '02: Proc. of the First Int'l Semantic Web Conference on The Semantic Web*, pages 333–347, London, UK. Springer.

Bibliography XI

Papadopoulos, G. A. and Arbab, F. (1998).

Coordination models and languages.

In Zelkowitz, M. V., editor, *The Engineering of Large Systems*, volume 46 of *Advances in Computers*, pages 329–400. Academic Press.

Pérez, J., Arenas, M., and Gutierrez, C. (2009).

Semantics and complexity of SPARQL.

ACM Transactions on Database Systems, 34(3):16:1–16:45.

Ricci, A. and Omicini, A. (2003).

Supporting coordination in open computational systems with TuCSoN.

In *IEEE 12th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)*, pages 365–370, 1st International Workshop "Theory and Practice of Open Computational Systems" (TAPOCS 2003), Linz, Austria. IEEE CS.

Proceedings.

Bibliography XII

Ricci, A., Omicini, A., and Denti, E. (2002).

Virtual enterprises and workflow management as agent coordination issues.

International Journal of Cooperative Information Systems, 11(3/4):355–379.

Rossi, D., Cabri, G., and Denti, E. (2001).

Tuple-based technologies for coordination.

chapter 4, pages 83–109.

Shearer, R., Motik, B., and Horrocks, I. (2008).

HermiT: A highly-efficient OWL reasoner.

In Dolbear, C., Ruttenberg, A., and Sattler, U., editors, *5th International Workshop "OWL: Experiences and Directions" (OWLED 2008)*, volume 432 of *CEUR Workshop Proceedings*, ISWC 2008, Karlsruhe, Germany.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007).

Pellet: A practical owl-dl reasoner.

J. Web Sem., 5(2):51–53.

Bibliography XIII

Straccia, U. (2001).

Reasoning within fuzzy description logics.

Journal of Artificial Intelligence Research, 14(1):137–166.

Straccia, U. (2005).

Description logics with fuzzy concrete domains.

In Bachus, F. and Jaakkola, T., editors, *21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 559–567, Edinburgh, Scotland. AUAI Press.

Tolksdorf, R., Nixon, L. J. B., and Simperl, E. (2008).

Towards a tuplespace-based middleware for the Semantic Web.

Web Intelligence and Agent Systems, 6(3):235–251.

Tsarkov, D. and Horrocks, I. (2006).

FaCT++ description logic reasoner: System description.

In Furbach, U. and Shankar, N., editors, *3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, volume 4130 of *LNCS*, pages 292–297. Springer, Seattle, WA, USA, 17–20 august edition.

Bibliography XIV

van der Kam, W., Moormanb, P. W., and Koppejan-Mulder, M. (2000).

Effects of electronic communication in general practice.

International Journal of Medical Informatics, 60:59–70.

Van der Lei, J., Musen, M., van der Does, E., Main in't Veld, A., and van Bemmel, J. (1991).

Review of physician decision making using data from computer-stored medical records.

The Lancet, 338:1504–1508.

Varshney, U. (2009).

Pervasive Healthcare Computing.

Springer.

Viroli, M. and Casadei, M. (2009).

Biochemical tuple spaces for self-organising coordination.

In Field, J. and Vasconcelos, V. T., editors, *Coordination Languages and Models*, volume 5521 of *LNCS*, pages 143–162. Springer, Lisbon, Portugal.

11th International Conference (COORDINATION 2009), Lisbon, Portugal, June 2009. Proceedings.

Bibliography XV


Viroli, M., Casadei, M., Montagna, S., and Zambonelli, F. (2011).
Spatial coordination of pervasive services through chemical-inspired tuple spaces.
ACM Transactions on Autonomous and Adaptive Systems.


Wegner, P. (1997).
Why interaction is more powerful than algorithms.
Communications of the ACM, 40(5):80–91.


Wooldridge, M. and Jennings, N. R. (1995).
Intelligent agents: Theory and practice.
Knowledge Engineering Review, 10(2):115–152.


Wyckoff, P., McLaughry, S. W., Lehman, T. J., and Ford, D. A. (1998).
T Spaces.
IBM Systems Journal, 37(3):454–474.

Bibliography XVI

Yen, J. (1991).

Generalizing term subsumption languages to fuzzy logic.

In *12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 472–477, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Zadeh, L. A. (1965).

Fuzzy sets.

Information and Control, 8(3):338–353.

Semantic and Fuzzy Coordination Through Programmable Tuple Spaces

Elena Nardini

Alma Mater Studiorum – Università di Bologna
elena.nardini@unibo.it

Cesena - May 31, 2011