Stefano Bragaglia, Federico Chesani, Anna Ciampolini,
Paola Mello, Marco Montali, and Davide Sottara

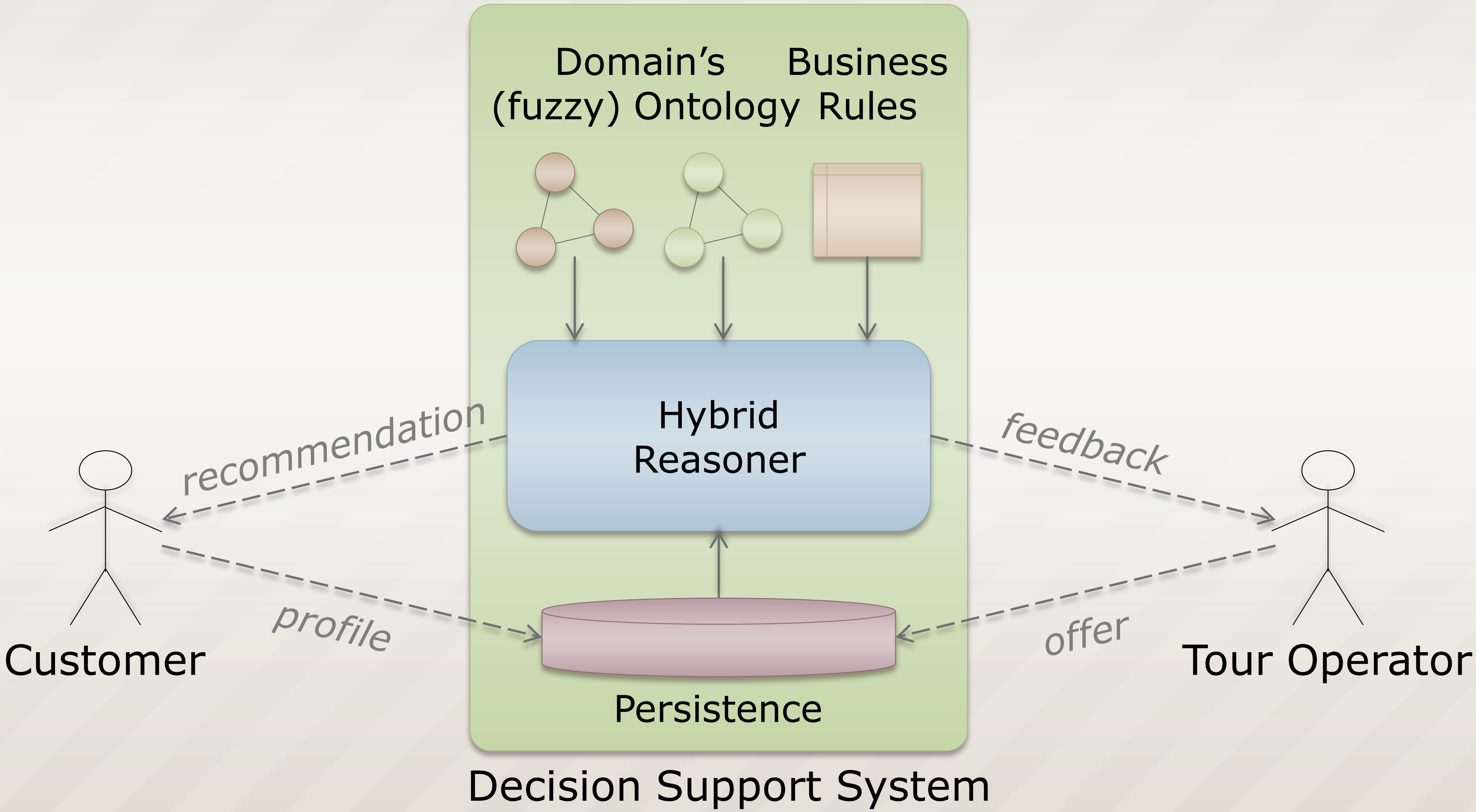
DEIS — University of Bologna
{name.surname}@unibo.it

An Hybrid Architecture Integrating Forward Rules with Fuzzy Ontological Reasoning

Introduction

- Nowadays, many domains rely on Rule-Based Systems (**RBSs**) to effectively manage their business
- Such systems allow to
  - model a domain
  - express the logic of the business processes
  - react to external stimuli
- When the conditions of a rule match the current status of the model, that rule triggers and its associated action takes place, possibly updating the model and triggering other rules

A Case Study


Domain's
(fuzzy) Ontology
Business
Rules
Hybrid
Reasoner
recommendation
feedback
profile
offer
Customer
Tour Operator
Persistence
Decision Support System

Limitations of RBSs

- E.g.: *Tour operators use a RBS to validate offers according to their quality standard*

- The process of developing RBSs rules is typically non-monotonic
  - refactoring of rules may be required when updating the knowledge base

- If a tour operator decides to introduce an *offer with at least one overnight stay* (calling it a *package*), the validation rule has to be changed accordingly

A Practical Example

- Validate offers according to quality standard, rejecting the bad ones

```
rule "validating offers"
when
▪ current item is an offer and
▪ it does not match the quality
  standard
then
▪ notify its author
▪ reject it
end
```

A Practical Example

- Validate offers according to quality standard, rejecting the bad ones

```
rule "validating offers"
when
  ▪ current item is an offer and
  ▪ it does not match the quality
    standard
then
  ▪ notify its author
  ▪ reject it
end
```

- The rule does not trigger on packages unless
  - a similar one is added for packages
  - the previous one is modified

A Practical Example

- Validate offers according to quality standard, rejecting the bad ones

```
rule "validating offers"
when
▪ current item is an offer and
▪ it does not match the quality
  standard
then
▪ notify its author
▪ reject it
end
```

- The rule does not trigger on packages unless
  - a similar one is added for packages
  - the previous one is modified

```
rule "validating items"
when
▪ (current item is an offer or
▪ is a package) and
▪ it does not match the quality
  standard
then
▪ notify its author
▪ reject it
end
```

Limitations of RBSs

- A Description Logics (**DL**) model, instead, would have inferred the relation between packages and offers
- Thus the RBS could exploit it and continue to operate without needing to change the rules

Limitations of RBSs

- A Description Logics (**DL**) model, instead, would have inferred the relation between packages and offers
- Thus the RBS could exploit it and continue to operate without needing to change the rules
- Similarly, many real-life domains are not «crisp», so Fuzzy Logics (**FL**) could help RBSs to handle «imperfect» knowledge (i.e.: by computing «how much the offer matches the quality standard»)

Motivations

- Growing interest into the combination of DL's descriptive capacity with RBS' operational semantics and FL expressiveness
  - **DL:** formal languages to represent knowledge, algorithms to reason upon it (consistency, classification, recognition)
  - **RBS:** express application logic with rules, triggered rules produce the outcomes expected by business logic
  - **FL:** express imperfect real-life domains naturally going beyond crisp knowledge
- Each single technology is mature by itself but some domains would benefit from all of them together (i.e.: Semantic Web)

Related Works

- The integration of couples of those reasoning styles has been already attempted or studied in literature:
  - **DL & RBS:** Jena, Algernon, Sweet-Rules (+FOL)
  - **FL & RBS:** FuzzyClips, FuzzyJess, Drools:Chance
  - **DL & FL:** DeLorean, FuzzyDL

- *No tool supporting ontological, rule-based and fuzzy reasoning at the same time is currently available*

Integration Approaches

- In general, the integration of different reasoning styles is rather difficult
- A few possible approaches has been identified:
  - **Loose integration:** uses available mature tools, requires an interface to dispatch each kind of knowledge to its pertaining module
  - **Tight integration:** defines a complex theory to cope with the desired reasoning styles and implements a system to support it

Implementation

- Our Java-based solution follows a loosely-coupled approach, exploiting
  - **Drools Expert** as RBS
  - **Pellet** as DL reasoner
  - **FuzzyDL** as FL reasoner
- The knowledge handled by each tools has to be kept aligned and consistent with the others
  - Drools as main component
  - Pellet an FuzzyDL called on demand

Implementation Issues

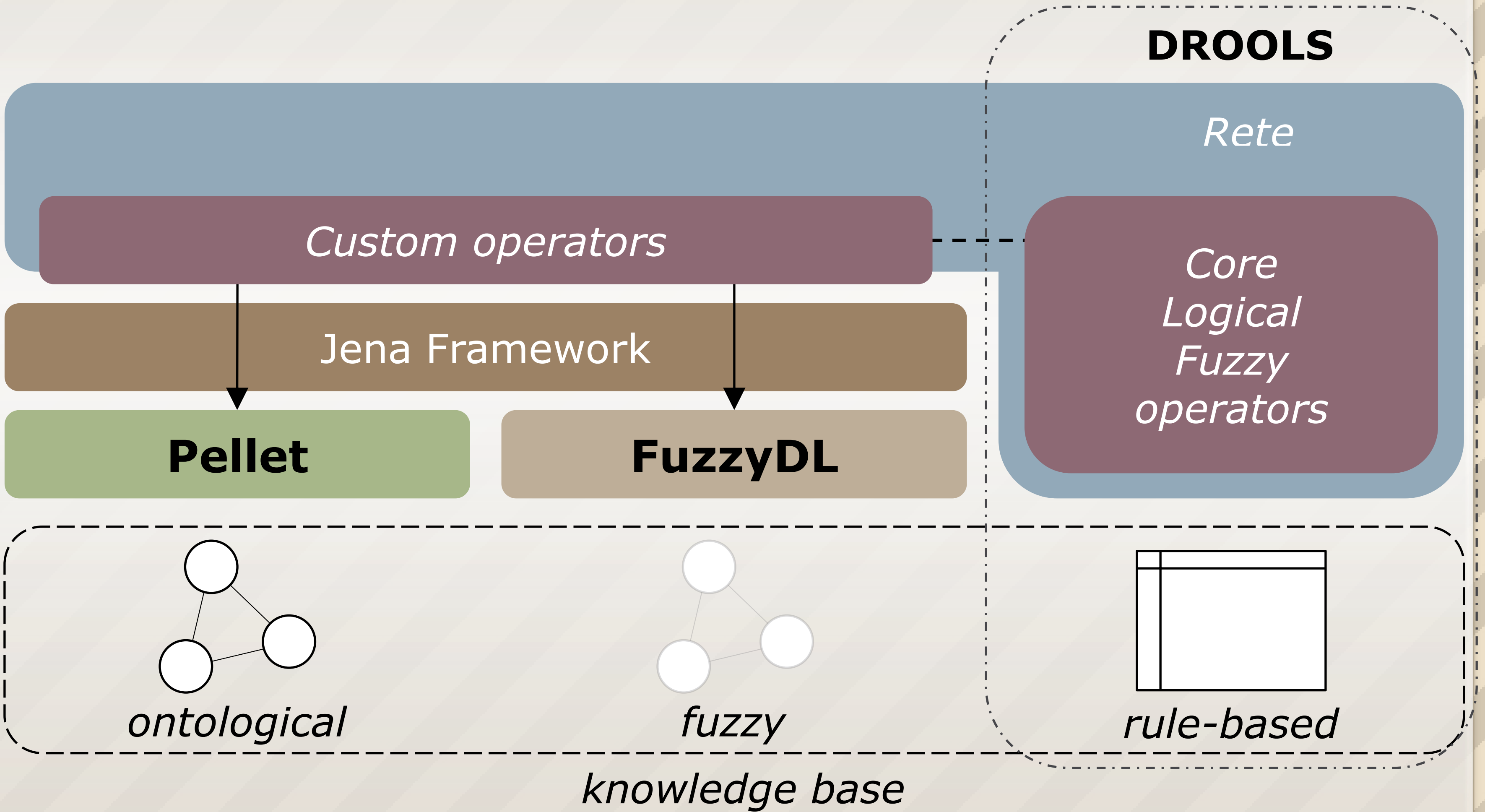
- With respect to DL & RBS, the main issue is due to their different contextual hypothesis:
  - RBS typically embrace *Close World Assumption*
  - DL usually adhere to *Open World Assumption*
- Hence, integrated systems has to deal with both deterministic and non-deterministic results
- Non-determinism could make the system undecidable

Implementation Issues

- When dealing with FL, its scope should be specified first:
  - Narrow sense (*truth functional many-valued logic*)
  - Broad sense

- In the context of Semantic Web, the assumed meaning is usually the former (easier to handle)

System Architecture



An Example of a Rule

- Suppose you want to model the fact that *sport offers should be recommended to young single male customers*
  - A sport offer is an offer associated with at least a sport event
  - A young single male customer is a male customer with no spouse and children…
  - …whose age is «roughly» between 15 and 35
  - Each time a new sport offer or young single male customer is added to the system, the rule should trigger

An example of a Rule

```
rule "Sport, young male singles"
filter 0.66 // drops matches below 0.66
when
  $c: Customer ( this isA Single.class,
                 gender == "m", age seems young )
  $o: Offer ( this isA SportOffer.class )
then
  send($o.toString(), $c.email,
     drools.getDegree());
end
```

Conclusions

- We have implemented a loosely-coupled hybrid reasoning tool capable of rule-based, ontological and fuzzy reasoning
- The rich environment provides a much increased expressiveness in rules that was only partially available before
- Thanks to its architecture centred on a single component, the tool remains stable and decidable during rules propagation
- The system may be easily extended to provide more functionalities by means of custom operators

Future works

- Unfortunately, current solution requires three distinct knowledge models which makes the memory usage quite inefficient and may lead to performance issues
- We are currently working on an improved version of the system headed toward a tighter integration (embedment) of the sub-modules, using only a single shared knowledge model