Rfuzzy: An expressive simple fuzzy compiler

Víctor Pablos Ceruelo, Susana Muñoz-Hernandez and Hannes Straß

Babel Group, Facultad de Informática
Universidad Politécnica de Madrid (T.U. of Madrid), Spain
http://babel.ls.fi.upm.es

IWANN 2009, Salamanca, Spain
We are able to model *Credibility* and *Fuzziness*. Example of rule:

\[
\text{IF cloudy THEN it\_may\_rain}
\]

- **Credibility**: Consider that it is cloudy. How much do we trust the rule that says that it may rain?
- **Fuzziness**: How much cloudy is it? Can we see the sun? We have different degrees.
Motivation (II): *Rfuzzy vs Fuzzy Prolog*

- **How Rfuzzy reduces *Fuzzy Prolog* complexity**
  - Uses real numbers instead of intervals between real numbers to represent truth values.
  - Answers queries with direct values instead of constraints.
  - Programmer does not have to code variables to manage truth values.

- **How Rfuzzy increases *Fuzzy Prolog* expressiveness**
  - Defines a types extension to limit the set of valid individuals.
  - Allows to define default truth values for subsets of individuals.

We do not compare *Rfuzzy* with other approaches that substitute the SLD-resolution with a fuzzy variant that is able to handle partial truth. The main advantage of our proposal over them is that we can reuse thousands of existing lines of prolog code, and mix fuzzy with crisp resolution.
Rfuzzy syntax example 1

```prolog
% type def of 1st argument of predicate young
:- set_prop young/1 => person/1.

person(john).  person(mike).
person(marcus). person(hannes).
person(susana). person(victor).

% default and concrete truth values decl
:- default(young/1, 0).  % default: Nobody
                       % is young
young(john) value 0.9.  % He is 11 years old.
young(mike) value 0.8.  % He is 15 years old.
```

VPC, SMH, HS (Babel Group, UPM, Spain)
Rfuzzy syntax example II

```prolog
:- default(teenager/1, 0).
% function truth value declaration
teenager :# ([  (9, 0),
             (10, 1), (19, 1),
             (20, 0) ]) .
```

VPC, SMH, HS (Babel Group, UPM, Spain)

Rfuzzy Framework

IWANN'09, Salamanca
Rfuzzy syntax example III

:- default(teenager/1, 0).
teenager :# ([ (9, 0), (10, 1), (19, 1), (20, 0) ]).

:- default(child/1, 0).
child :# ([ (0, 1), (10, 1), (12, 0) ]).

:- default(young/1, 0).
young(P) cred (min, 0.3) :- max child(P),
                            teenager(P).
% Every one is young with a tv of 0.3

Available aggregation operators: minimum, maximum, (inverse) product, (inverse) Lukasiewicz operator.
Rfuzzy syntax example IV

% Susana says that she is still young ...
young(susana) value 0.7.

% But the fact is that all the authors are still young. (the previous declaration is removed)
% Conditional truth value declaration
:- default(young/1, 0.7) ⇒ paper_author/1.
paper_author(susana). paper_author(hannes).
paper_author(victor).

% Previous def and decl of young
:- default(young/1, 0).
young(P) cred (min, 0.3) :∼ max child(P),
              teenager(P).
Queries, Answers and Constructivity

On compilation we add a new argument (V is the truth value arg)

% non-constructive
?- young(hannes,V).
V = 0.7 ? ;
no

% constructive
?- young(X,V), V ≥ 0.8.
V = 0.9, X = john ? ;
V = 0.8, X = mike ? ;
no

We say that it is constructive because it does not return a constraint X ≠ susana, hannes, victor, marcus.
Instead of that it returns the valid values in a Prolog way (i.e. one by one).
When looking for the truth value of an individual Rfuzzy returns only one truth value, and it is always the most concrete truth value declared in the program.
The global compilation process has two preprocessor steps:

1. the Rfuzzy program is translated into CLP(\(\mathcal{R}\)) constraints by means of the Rfuzzy package and
2. those constraints are translated into ISO Prolog by using the CLP(\(\mathcal{R}\)) package.

We do not create a new SLD-resolution. Instead of that we use prolog SLD resolution. This allows us to mix prolog code with fuzzy prolog code, and reuse a lot of prolog existing code.
Example: MYCIN can be modelled by using \textit{Rfuzzy}

\begin{itemize}
\item \textbf{IF} The stain of the organism \textbf{is} Gram negative, AND
\item The morphology of the organism \textbf{is} rod, AND
\item The aerobicity of the organism \textbf{is} aerobic
\item \textbf{THEN} There \textbf{is} strongly suggestive evidence (0.8) that the class of the organism \textbf{is} Enterobacteriaceae
\end{itemize}

Rules have credibility and facts have a truth value.

\begin{itemize}
\item \text{CF(}\text{stain} = \text{gram–negative}) = 0.4.
\item \text{CF(}\text{morphology} = \text{rod}) = 0.6.
\item \text{CF(}\text{aerobicity} = \text{aerobic}) = -0.4.
\end{itemize}
Conclusions

*Rfuzzy* offers to the users a new framework to represent fuzzy problems over real numbers. Main *Rfuzzy* advantages over *Fuzzy Prolog* are a simpler syntax and the elimination of answers with constraints.

- Its fuzzy values are simple real numbers instead of intervals between real numbers,
- it hides the management of truth value variables and
- as it does not answer queries with constraints, answers are always ground terms. This is a more human-readable way of answering questions and it can be used in web forms (constraints can not be used there).

Applications: Search Engines, Knowledge Extraction (from databases, ontologies, etc.), Semantic Web, Business Rules, Coding Rules (where the violation of one rule can be given a truth value), etc.
Current work

?- ¬young(X, V).

Apply constructive negation to the engine, but first we have to find the correct semantics. What do we want to obtain when we make a query like this one?

Have fuzzy relations in the Semantic Web, allowing the user to represent not only who is young or not but the fact that the authors of this paper belong to the young set with a truth value of 0.7 ... .
Questions

Questions are guaranteed in life; Answers aren't.