Declarative Traces Into Fuzzy Computed Answers

Pedro J. Morcillo Ginés Moreno

Jaime Penabad Carlos Váquez

Faculty of Computer Science Engineering

University of Castilla – La Mancha

02005, Albacete, SPAIN

Outline of the talk

- Multi-Adjoint Logic Programming
- The Fuzzy Logic System FLOPER
- Fuzzy Computed Answers with Traces
- Conclusions and Further Research

Multi-Adjoint Logic Programming

FUZZY LOGIC PROGRAMMING

$\Downarrow$

FUZZY LOGIC

+

LOGIC PROGRAMMING

Multi-Adjoint Logic Programming

- Although there is no an standard language, we have found two major approaches:

1. Likelog *[Arcelli & Formato-99]*
   Bousi – Prolog *[Julián & Rubio-07]*

SLD-resolution + **FUZZY (similarity)** unification

2. f-Prolog *[Vojtas & Paulík-96]*

FUZZY SLD-resolution + (syntactic) unification

- MALP *[Medina & Ojeda–Aciego & Vojtas-01]*

Admissible/Interpretive Computation + (syntactic) unification

Multi-Adjoint Logic Programming

- Let $\mathcal{L}$ be a PROLOG-like first order language, with:

  constants variables functions

  predicates quantifiers: $\forall, \exists$ BUT MORE connectives!!!

$$
\begin{array}{lllll}
\&_1, & \&_2, & \ldots, & \&_k & \texttt{(conjunctions)} \\
\vee_1, & \vee_2, & \ldots, & \vee_l & \texttt{(disjunctions)} \\
\leftarrow_1, & \leftarrow_2, & \ldots, & \leftarrow_m & \texttt{(implications)} \\
@_1, & @_2, & \ldots, & @_n & \texttt{(aggregations)}
\end{array}
$$

- Instead of naive $\{true,\ false\}$, use a multi–adjoint lattice to model truth degrees $\langle L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n \rangle$.

$$\langle [\mathbf{0}, \mathbf{1}], \preceq, \leftarrow_{\mathbf{Luka}}, \&_{\mathbf{Luka}}, \leftarrow_{\mathbf{Prod}}, \&_{\mathbf{Prod}}, \leftarrow_{\mathbf{Godel}}, \&_{\mathbf{Godel}} \rangle$$

Multi-Adjoint Logic Programming

- SYNTAX: Assume some connective definitions like:

$$\begin{array}{llll} \&_{\texttt{Godel}}(x_1, x_2) & \triangleq & min(x_1, x_2) & \text{- - in brief} \quad \&_{\texttt{G}} \\ \&_{\texttt{Prod}}(x_1, x_2) & \triangleq & x_1 * x_2 & \text{- - in brief} \quad \&_{\texttt{P}} \\ \vee_{\texttt{Luka}}(x_1, x_2) & \triangleq & min(1, x_1 + x_2) & \text{- - in brief} \quad \vee_{\texttt{L}} \\ @_{\texttt{aver}}(x_1, x_2) & \triangleq & (x_1 + x_2)/2 & \end{array}$$

- A program is a set of "weighted" rules $\texttt{A} \leftarrow_{\texttt{i}} \mathcal{B}$ **with** α:

$$\begin{array}{llllll} \mathcal{R}_1 : & p(X) & \leftarrow_{\texttt{P}} & \&_{\texttt{G}}(q(X), @_{\texttt{aver}}(r(X), s(X))) & with & 0.9 \\ \mathcal{R}_2 : & q(a) & \leftarrow & & with & 0.8 \\ \mathcal{R}_3 : & r(X) & \leftarrow & & with & 0.7 \\ \mathcal{R}_4 : & s(X) & \leftarrow & & with & 0.5 \end{array}$$

Multi-Adjoint Logic Programming

- STATE : Is a pair with form $\boxed{\langle goal;\ substitution\rangle}$
- INPUT (goal): For instance, $\boxed{\langle p(X);\ id\rangle}$
- OUTPUT (fuzzy computed answer): Is a (final) state of the form $\boxed{\langle truth\_degree;\ substitution\rangle}$
- PROCEDURAL SEMANTICS:
  Operational phase: Admissible steps ($\rightarrow_{AS}$)
  Interpretive phase: Interpretive steps ($\rightarrow_{IS}$)
- Given a program $\mathcal{P}$, goal $\mathcal{Q}$ and substitution σ, we define an STATE TRANSITION SYSTEM whose transition relations are $\rightarrow_{AS}$ and $\rightarrow_{IS}$

Multi-Adjoint Logic Programming

ADMISSIBLE STEP OF KIND $\rightarrow_{AS1}$

$\langle \mathcal{Q}[A]; \sigma \rangle \ \rightarrow_{AS1} \ \langle (\mathcal{Q}[A/v\&_i\mathcal{B}])\theta; \sigma\theta \rangle$ if

(1) A is the selected atom in $\mathcal{Q}$,

(2) $\theta = mgu(\{A' = A\})$,

(3) $A' \leftarrow_i \mathcal{B}$ with v in $\mathcal{P}$ and $\mathcal{B}$ is not empty.

EXAMPLE. Let $\mathtt{p(a)} \leftarrow_{\mathtt{prod}} \mathtt{p(f(a))}$ $\mathtt{with}$ 0.7 be a rule

$\langle (\mathtt{p(b)} \&_{\mathtt{G}} \mathtt{p(X)}) \&_{\mathtt{G}} \mathtt{q(X)}; \mathtt{id} \rangle \rightarrow_{AS1}$

$\langle (\mathtt{p(b)} \&_{\mathtt{G}} 0.7 \&_{\mathtt{prod}} \mathtt{p(f(a))}) \&_{\mathtt{G}} \mathtt{q(a)}; \{\mathtt{X/a}\} \rangle$

Multi-Adjoint Logic Programming

ADMISSIBLE STEP OF KIND $\rightarrow_{AS2}$

$\langle \mathcal{Q}[A]; \sigma \rangle \;\; \rightarrow_{AS2} \;\; \langle (\mathcal{Q}[A/v])\theta; \sigma\theta \rangle$ if

(1) A is the selected atom in $\mathcal{Q}$,

(2) $\theta = mgu(\{A' = A\})$, and

(3) $A' \leftarrow$ with v in $\mathcal{P}$.

EXAMPLE. Let $\mathtt{p(a)} \leftarrow \mathtt{with}\ \mathtt{0.7}$ be a rule

$\langle (\mathtt{p(b)} \&_{\mathtt{G}} \mathtt{p(X)}) \&_{\mathtt{G}} \mathtt{q(X)}; \mathtt{id} \rangle \;\; \rightarrow_{AS2}$
$\langle (\mathtt{p(b)} \&_{\mathtt{G}} \mathtt{0.7}) \&_{\mathtt{G}} \mathtt{q(a)}; \{\mathtt{X/a}\} \rangle$

Multi-Adjoint Logic Programming

ADMISSIBLE STEP OF KIND $\rightarrow_{AS3}$

$\langle \mathcal{Q}[A]; \sigma \rangle \ \rightarrow_{AS3} \ \langle (\mathcal{Q}[A/\bot]); \sigma \rangle$ if

(1) A is the selected atom in $\mathcal{Q}$,

(2) There is no rule in $\mathcal{P}$ whose head unifies with A

This case is introduced to cope with (possible) unsuccessful admissible derivations. For instance, with program $\mathtt{p(a)}$ <prod @aver$(1, \mathtt{p(b)})$ with 0.9

we have : $\langle \mathtt{p(b)}; \mathtt{id} \rangle \ \rightarrow_{AS3} \ \langle 0; \mathtt{id} \rangle$

Multi-Adjoint Logic Programming

INTERPRETIVE STEP $\to_{IS}$

$$\langle Q[@(r_1, r_2)]; \sigma\rangle \;\; \to_{IS} \;\; \langle Q[@(r_1,r_2)/[\![@]\!](r_1,r_2)];\sigma\rangle$$

where $[\![@]\!]$ is the truth function of connective $@$ in the multi-adjoint lattice associated to $\mathcal{P}$

EXAMPLE. Since the truth function associated to $\&_{\texttt{prod}}$ is the product operator, then

$$\langle(0.8\&_{\texttt{luka}}((0.7\&_{\texttt{prod}}0.9)\&_{\texttt{G}}0.7)); \{X/a\}\rangle \;\; \to_{IS}$$
$$\langle(0.8\&_{\texttt{luka}}(0.63\&_{\texttt{G}}0.7)); \{X/a\}\rangle$$

Declarative Traces Into Fuzzy Computed Answers

Multi-Adjoint Logic Programming

$$\langle \underline{p(X)};\ id\rangle \rightarrow_{AS1}{}^{\mathcal{R}_1}$$

$$\langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(\underline{q(X_1)}, @_{\mathtt{aver}}(r(X1), s(X1)))); \{X/X_1\}\rangle \rightarrow_{AS2}{}^{\mathcal{R}_2}$$

$$\langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, @_{\mathtt{aver}}(\underline{r(a)}, s(a)))); \{X/a\}\rangle \rightarrow_{AS2}{}^{\mathcal{R}_3}$$

$$\langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, @_{\mathtt{aver}}(0.7, \underline{s(a)}))); \{X/a\}\rangle \rightarrow_{AS2}{}^{\mathcal{R}_4}$$

$$\langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, \underline{@_{\mathtt{aver}}(0.7, 0.5)})); \{X/a\}\rangle \rightarrow_{IS}$$

$$\langle \&_{\mathtt{P}}(0.9, \underline{\&_{\mathtt{G}}(0.8, 0.6)}); \{X/a\}\rangle \rightarrow_{IS}$$

$$\langle \underline{\&_{\mathtt{P}}(0.9, 0.6)}; \{X/a\}\rangle \rightarrow_{IS}$$

$$\langle 0.54; \{X/a\}\rangle$$

So, for derivation D_1 the f.c.a is $\langle 0.54; \{X/a\}\rangle$

Multi-Adjoint Logic Programming

- Better than $\rightarrow_{IS}$ steps, perform small interpretive steps to visualize in detail the complexity of fuzzy connectives

$\longrightarrow_{SIS1}$: expand a connective definition $\Omega(x_1, \ldots, x_n) \triangleq E$

$$\langle Q[\Omega(r_1, \ldots, r_n)]; \sigma\rangle \quad \rightarrow_{SIS1} \quad \langle Q[\Omega(r_1, \ldots, r_n)/E'];\sigma\rangle$$

where $E' = E[x_1/r_1, \ldots, x_n/r_n]$

$\longrightarrow_{SIS2}$: evaluate a primitive operator $\Omega(r_1, \ldots, r_n) = r$

$$\langle Q[\Omega(r_1, \ldots, r_n)]; \sigma\rangle \quad \rightarrow_{SIS2} \quad \langle Q[\Omega(r_1, \ldots, r_n)/r];\sigma\rangle$$

Multi-Adjoint Logic Programming

$$
\begin{array}{lll}
D_2: & \langle \underline{p(X)};\ id\rangle \rightarrow_{AS1}{}^{\mathcal{R}_1} \ \ldots\ldots & \rightarrow_{AS2}{}^{\mathcal{R}_4} \\
 & \langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, \underline{@_{\mathtt{aver}}(0.7, 0.5)})); \{X/X_1\}\rangle & \rightarrow_{SIS1} \\
 & \langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, \underline{(0.7+0.5)}/2)); \{X/a\}\rangle & \rightarrow_{SIS2} \\
 & \langle \&_{\mathtt{P}}(0.9, \&_{\mathtt{G}}(0.8, \underline{1.2/2})); \{X/a\}\rangle & \rightarrow_{SIS2} \\
 & \langle \&_{\mathtt{P}}(0.9, \underline{\&_{\mathtt{G}}(0.8, 0.6)}); \{X/a\}\rangle & \rightarrow_{SIS1} \\
 & \langle \&_{\mathtt{P}}(0.9, \underline{min(0.8, 0.6)}); \{X/a\}\rangle & \rightarrow_{SIS2} \\
 & \langle \underline{\&_{\mathtt{P}}(0.9, 0.6)}; \{X/a\}\rangle & \rightarrow_{SIS1} \\
 & \langle \underline{0.9 * 0.6}; \{X/a\}\rangle & \rightarrow_{SIS2} \\
 & \langle 0.54; \{X/a\}\rangle &
\end{array}
$$

- Now, instead of 3 $\rightarrow_{IS}$, we have 3 $\rightarrow_{SIS1}$ plus 4 $\rightarrow_{SIS2}$

The Fuzzy Logic System FLOPER

http://dectau.uclm.es/floper/

PROLOG has been largely used for four purposes.....

- Implementing the tool (about 1000 clauses, DCG's)
- Compiling fuzzy programs (compiled code in Prolog):
  - HIGH LEVEL: transparent execution of goals
  - LOW LEVEL: drawing derivations and trees
- Modeling multi-adjoint lattices (represent truth degrees)

The Fuzzy Logic System FLOPER

- Basic options for...
  - Loading a prolog file with extension ".pl"
  - Parsing a ".fpl" fuzzy program. The resulting Prolog code is also asserted in the system
  - Saving the generated Prolog code into a ".pl" file
  - Listing both the fuzzy and Prolog code
  - Clean, Stop and Quit
- Advanced options for...
  - Running a fuzzy program after introducing a goal
  - Drawing derivations and unfolding trees (varying its depth and the level of detail of interpretive steps)
  - Loading/showing multi-adjoint lattices (".pl" files)

Declarative Traces Into Fuzzy Computed Answers

The Fuzzy Logic System FLOPER. Option RUN

$PROGRAM:$ $\mathtt{p(X)} \leftarrow_{\mathtt{P}} \&_{\mathtt{G}}(\mathtt{q(X)}, @_{\mathtt{aver}}(\mathtt{r(X)}, \mathtt{s(X)}))$ **with 0.9**

$\mathtt{q(a)}$ **with 0.8** $\mathtt{r(X)}$ **with 0.7** $\mathtt{s(\_)}$ **with 0.5**

——————————————»»»»»»>

$\mathtt{p(X, TV0)}$:- $\mathtt{q(X, TV1)}$, $\mathtt{r(Y, TV2)}$, $\mathtt{s(X, TV3)}$,

$\mathtt{agr\_aver(TV2, TV3, TV4)}$,

$\mathtt{and\_godel(TV1, TV4, TV5)}$,

$\mathtt{and\_prod(0.9, TV5, TV0)}$.

$\mathtt{q(a, 0.8)}$. $\mathtt{r(X, 0.7)}$. $\mathtt{s(\_, 0.5)}$.

$GOAL:$ $\mathtt{p(X)}$ ——»»»»»»> ?- $\mathtt{p(X, Truth\_degree)}$

$OUTPUT:$ $[\mathtt{X} = \mathtt{a}, \mathtt{Truth\_degree} = 0.54]$

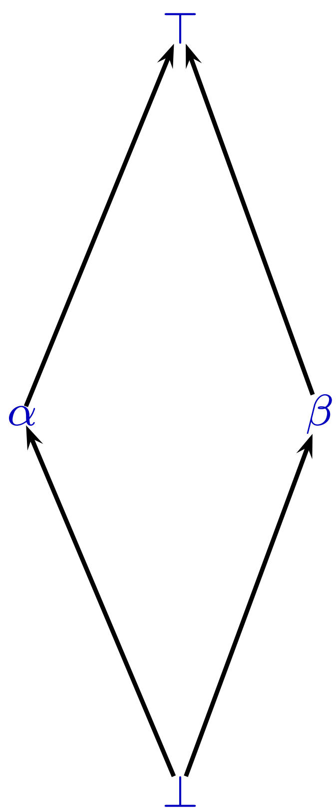
The Fuzzy Logic System FLOPER. Option LAT

- Prolog file modeling an infinite lattice of real numbers $[0, 1]$

```
member(X) :- number(X),0=<X,X=<1.                              leq(X,Y) :- X=<Y.
                  bot(0).                      top(1).

and_luka(X,Y,Z)  :- pri_add(X,Y,U1),pri_sub(U1,1,U2),pri_max(0,U2,Z).
and_godel(X,Y,Z) :- pri_min(X,Y,Z).
and_prod(X,Y,Z)  :- pri_prod(X,Y,Z).

or_luka(X,Y,Z)   :- pri_add(X,Y,U1),pri_min(U1,1,Z).
or_godel(X,Y,Z)  :- pri_max(X,Y,Z).
or_prod(X,Y,Z)   :- pri_prod(X,Y,U1),pri_add(X,Y,U2),pri_sub(U2,U1,Z).

agr_aver(X,Y,Z)  :- pri_add(X,Y,U),pri_div(U,2,Z).

pri_add(X,Y,Z)   :- Z is X+Y.    pri_min(X,Y,Z) :- (X=<Y,Z=X;X>Y,Z=Y).
pri_sub(X,Y,Z)   :- Z is X-Y.    pri_max(X,Y,Z) :- (X=<Y,Z=Y;X>Y,Z=X).
pri_prod(X,Y,Z)  :- Z is X * Y.  pri_div(X,Y,Z) :- Z is X/Y.
```

The Fuzzy Logic System FLOPER. Option LAT

- Prolog file modeling a finite lattice with partial ordering




```
members([bottom,alpha,beta,top]).
top(top).                                bot(bottom).

leq(bottom,X). leq(alpha,alpha).  leq(alpha,top).
leq(beta,beta).  leq(beta,top).  leq(X,top).

and_godel(X,Y,Z) :- pri_inf(X,Y,Z).

pri_inf(bottom,X,bottom):-!.
pri_inf(alpha,X,alpha):-leq(alpha,X),!.
pri_inf(beta,X,beta):-leq(beta,X),!.
pri_inf(top,X,X):-!.
pri_inf(X,Y,bottom).
```

The Fuzzy Logic System FLOPER. Option TREE

$$p(X) \quad \leftarrow_{\mathtt{P}} \quad \&_{\mathtt{G}}(q(X), @_{\mathtt{aver}}(r(X), s(X))) \quad \textbf{with} \quad \mathbf{0.9}$$

——————————————————————————»»»»»»»>

```
rule(number(1),
     head(atom(pred(p,1),[var('X')])),
     impl(prod),
     body(and(godel,2,
          [atom(pred(q,1),[var('X')]),
             agr(aver,2,
                [atom(pred(r,1),[var('X')]),
                 atom(pred(s,1),[var('X')])
          ])])),
     td(0.9)).
```

The Fuzzy Logic System FLOPER. Option TREE

- EXAMPLE: maintain the program and goal, but change the lattice of truth degrees, redefining “average”:
- Instead of: $@_{\mathtt{aver}}(x_1, x_2) \triangleq (x_1 + x_2)/2$, in PROLOG:
  $\mathtt{agr\_aver}(\mathtt{X}, \mathtt{Y}, \mathtt{Z}) :- \mathtt{pri\_add}(\mathtt{X}, \mathtt{Y}, \mathtt{U}), \mathtt{pri\_div}(\mathtt{U}, 2, \mathtt{Z}).$
- Use now: $@_{\mathtt{aver}}(x_1, x_2) \triangleq (\vee_{\mathtt{G}}(x_1, x_2) + \vee_{\mathtt{L}}(x_1, x_2))/2$
  $\mathtt{agr\_aver}(\mathtt{X}, \mathtt{Y}, \mathtt{Z}) :- \mathtt{or\_godel}(\mathtt{X}, \mathtt{Y}, \mathtt{Z1}), \mathtt{or\_luka}(\mathtt{X}, \mathtt{Y}, \mathtt{Z2}),$
  $\mathtt{pri\_add}(\mathtt{Z1}, \mathtt{Z2}, \mathtt{Z3}), \mathtt{pri\_div}(\mathtt{Z3}, 2, \mathtt{Z}).$
- Now, without changing the same program and goal, the solution for $\mathtt{p}(\mathtt{X})$ is not $[\mathtt{Truth} = 0.54, \mathtt{X} = \mathtt{a}]$ but $[\mathtt{Truth} = 0.72, \mathtt{X} = \mathtt{a}]$

The Fuzzy Logic System FLOPER. Option TREE

- Unfolding tree with option ismode = LARGE

```
R0 <p(X),{}>
 R1 <&prod(0.9,&godel(q(X1),@aver(r(X1),s(X1)))),{X/X1} >
  R2 <&prod(0.9,&godel(0.8,@aver(r(a),s(a)))),{X/a,X1/a}>
   R3 <&prod(0.9,&godel(0.8,@aver(0.7,s(a)))),{X/a,X1/a,X11/a}>
    R4 <&prod(0.9,&godel(0.8,@aver(0.7,0.5))),{X/a,X1/a,X11/a}>
      result < 0.7200000000000001,{X/a,X1/a,X11/a}>
```

- Unfolding tree with option ismode = MEDIUM

```
R0 <p(X),{}>
 R1 <&prod(0.9,&godel(q(X1),@aver(r(X1),s(X1)))),{X/X1} >
  R2 <&prod(0.9,&godel(0.8,@aver(r(a),s(a)))),{X/a,X1/a}>
   R3 <&prod(0.9,&godel(0.8,@aver(0.7,s(a)))),{X/a,X1/a,X11/a}>
    R4 <&prod(0.9,&godel(0.8,@aver(0.7,0.5))),{X/a,X1/a,X11/a}>
     is <&prod(0.9,&godel(0.8,0.85)),{X/a,X1/a,X11/a}>
      is <&prod(0.9,0.8),{X/a,X1/a,X11/a}>
       is <0.7200000000000001,{X/a,X1/a,X11/a}>
```

Declarative Traces Into Fuzzy Computed Answers

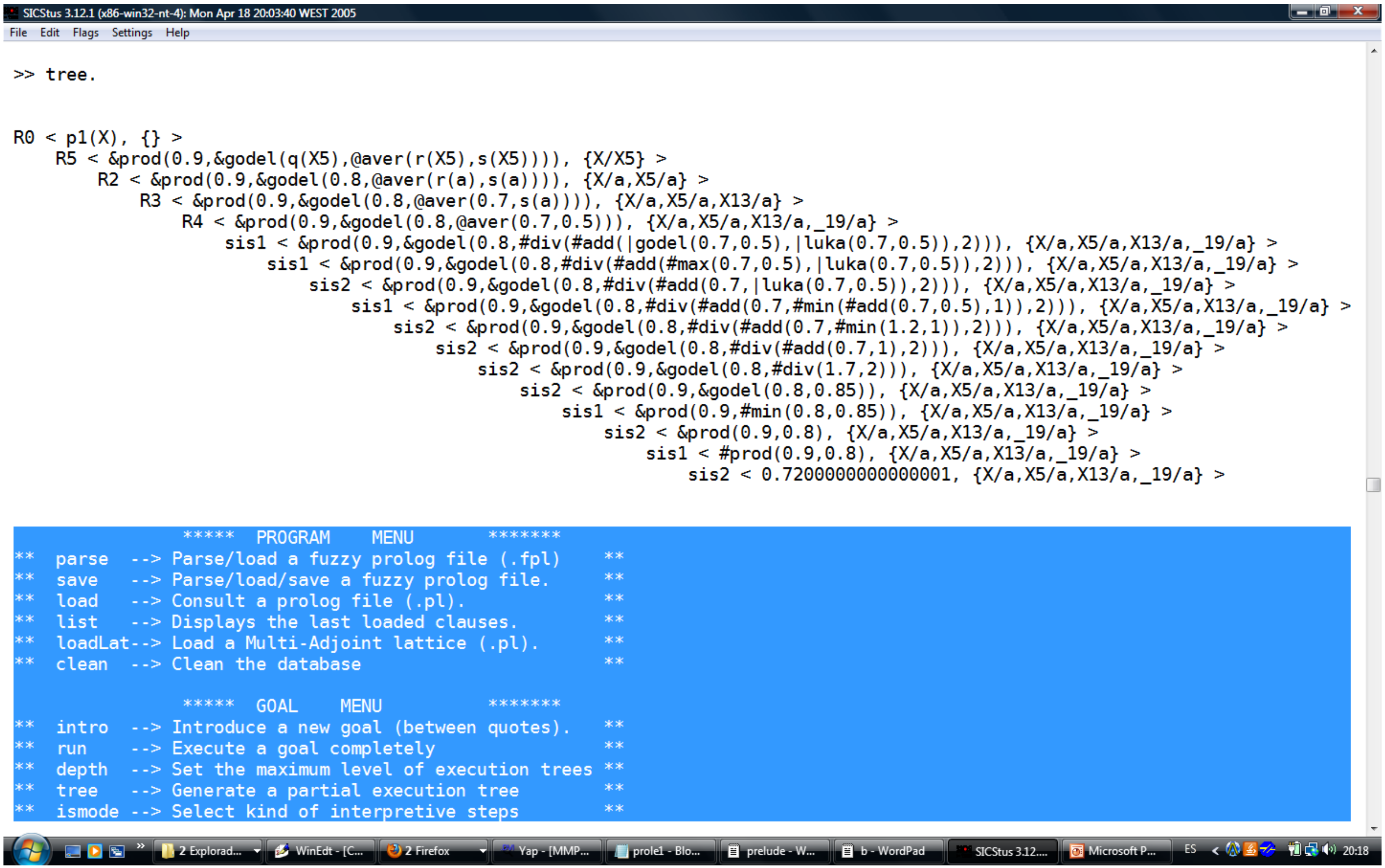
The Fuzzy Logic System FLOPER. Option TREE

- Unfolding tree with option ismode = SMALL

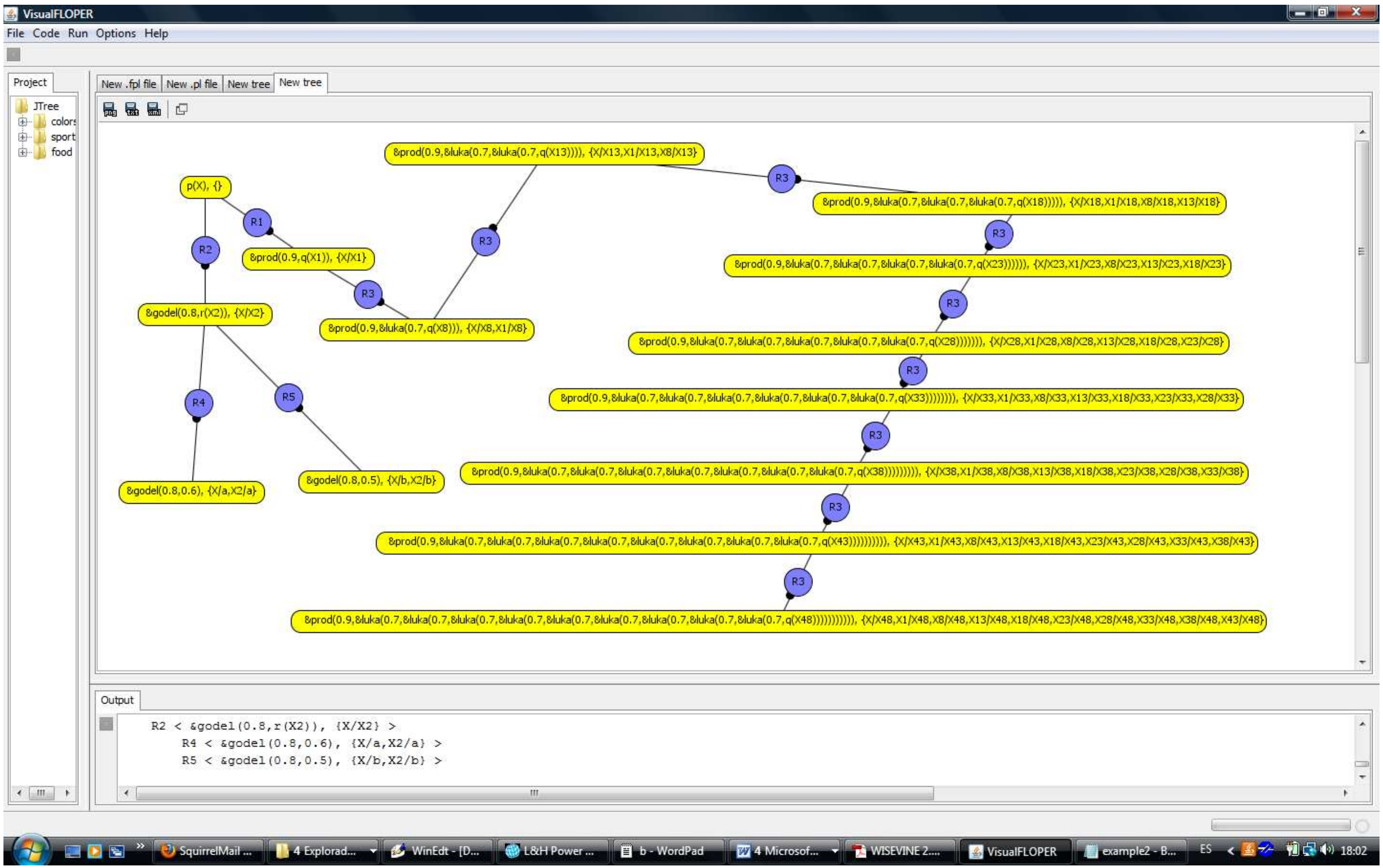
```
R0 <p(X),{}>
 R1 <&prod(0.9,&godel(q(X1),@aver(r(X1),s(X1)))),{X/X1} >
  R2 <&prod(0.9,&godel(0.8,@aver(r(a),s(a)))),{X/a,X1/a}>
   R3 <&prod(0.9,&godel(0.8,@aver(0.7,s(a)))),{X/a,X1/a,X11/a}>
    R4 <&prod(0.9,&godel(0.8,@aver(0.7,0.5))),{X/a,X1/a,X11/a}>
     sis1 <&prod(0.9,&godel(0.8,#prod(#add(|godel(0.7,0.5),|luka( ..
      sis1 <&prod(0.9,&godel(0.8,#prod(#add(#max(0.7,0.5),|luka(0.7.
       sis2 <&prod(0.9,&godel(0.8,#prod(#add(0.7,|luka(0.7,0.5)), ..
        sis1 <&prod(0.9,&godel(0.8,#prod(#add(0.7,#min(#add(0.7, ...
         sis2 <&prod(0.9,&godel(0.8,#prod(#add(0.7,#min(1.2,1)), ...
          sis2 <&prod(0.9,&godel(0.8,#prod(#add(0.7,1),0.5))), .....
           sis2 <&prod(0.9,&godel(0.8,#prod(1.7,0.5))), {X/a,X1/a ..
            sis2 <&prod(0.9,&godel(0.8,0.85)), {X/a,X1/a,X11/a}>
             sis1 <&prod(0.9,#min(0.8,0.85)), {X/a,X1/a,X11/a}>
              sis2 <&prod(0.9,0.8), {X/a,X1/a,X11/a}>
               sis1 <#prod(0.9,0.8), {X/a,X1/a,X11/a}>
                sis2 <0.7200000000000001, {X/a,X1/a,X11/a}>
```

Declarative Traces Into Fuzzy Computed Answers

The Fuzzy Logic System FLOPER



The Fuzzy Logic System FLOPER



Fuzzy Computed Answers with Traces

- Let the "proof weight" lattice $\mathcal{W}$ of *[Rodríguez & Romero-08]* that is, natural numbers with the inverted ordering and operator "$+$" assigned to all connectives ($\&_{\texttt{P}}$, $\&_{\texttt{G}}$, $@_{\texttt{aver}}$)

- $$\begin{array}{llllll} \mathcal{R}_1 : & p(X) & \leftarrow_{\texttt{P}} & \&_{\texttt{G}}(q(X), @_{\texttt{aver}}(r(X), s(X))) & with & 1 \\ \mathcal{R}_2 : & q(a) & \leftarrow & & with & 1 \\ \mathcal{R}_3 : & r(X) & \leftarrow & & with & 1 \\ \mathcal{R}_4 : & s(X) & \leftarrow & & with & 1 \end{array}$$

- Fuzzy Computed Answer counting admissible steps

```
» run.
            [Truth_degree=4, X=a]
```

Fuzzy Computed Answers with Traces

- Consider a lattice where truth degrees are lists and append is assigned to all connectives ($\&_{\mathtt{P}}$, $\&_{\mathtt{G}}$, $@_{\mathtt{aver}}$)

- $$\begin{array}{llll} p(X) & \leftarrow_{\mathtt{P}} & \&_{\mathtt{G}}(q(X), @_{\mathtt{aver}}(r(X), s(X))) & with \quad [\,rule1\,] \\ q(a) & \leftarrow & & with \quad [\,rule2\,] \\ r(X) & \leftarrow & & with \quad [\,rule3\,] \\ s(X) & \leftarrow & & with \quad [\,rule4\,] \end{array}$$

- Fuzzy Computed Answer detailing admissible steps
  » `run.`

```
[Truth_degree=[rule1,rule2,rule3,rule4], X=a]
```

Fuzzy Computed Answers with Traces

- Multi-adjoint lattices as CARTESIAN PRODUCTS:

```
member(info(X,_)):-number(X),0=<X,X=<1.                    bot(info(0,_)).
leq(info(X1,_),info(X2,_)) :- X1 =< X2.                   top(info(1,_)).

and_prod(info(X1,X2),info(Y1,Y2),info(Z1,Z2))  :-  pri_prod(X1,Y1,Z1),
                      pri_app(X2,Y2,Label),pri_app(Label,'&PROD.',Z2).

pri_app(X,Y,Z)  :-  name(X,L1),name(Y,L2),append(L1,L2,L3),name(Z,L3).
```

- $p(X) \leftarrow_{\mathtt{P}} \&_{\mathtt{G}}(q(X), @_{\mathtt{aver}}(r(X), s(X)))$ with $\mathtt{info}(0.9, \mathtt{rule1})$

...

- [X=a, Truth_degree=info(0.72, rule1.rule2.rule3.rule4. @AVER. &GODEL. &PROD.)]

Fuzzy Computed Answers with Traces

- More on multi-adjoint lattices as Cartesian products...
- To cope with small interpretive steps, it suffices by "attaching" appropriate labels to primitive operators

```
pri_prod(X,Y,Z,'#PROD.'):-Z is X * Y.

and_prod(info(X1,X2),info(Y1,Y2),info(Z1,Z2)):-
            pri_prod(X1,Y1,Z1,DatPROD),pri_app(X2,Y2,Dat1),
            pri_app(Dat1,'&PROD.',Dat2),pri_app(Dat2,DatPROD,Z2).
```

- [X=a, Truth_degree=info(0.72, rule1.rule2.rule3.rule4. @AVER. |GODEL. #MAX. |LUKA. #ADD.#MIN. @aver. #ADD.#DIV. &GODEL. #MIN. &PROD. #PROD.)]

Fuzzy Computed Answers with Traces

http://dectau.uclm.es/FuzzyXPath/

- Management of XML documents by using flexible variants of standard XPath/XQuery languages

- MALP implementation using FLOPER & SWI-Prolog:

```
member(tv(N,L)) :-  number(N),0=<N,N=<1,
                                (L=[];L=[_|_]).

and_prod(tv(X1,X2),tv(Y1,Y2),tv(Z1,Z2)) :-
          Z1 is X1*Y1,  append(X2,Y2,Z2).
```

Fuzzy Computed Answers with Traces

An appropriate call to predicate `fuzzyXPath` could return:

```
tv(1, [[],
    tv(0.9, [[],
        tv(0.9, [element(title, [], [Don Quijote de la Mancha]), [],
        tv(0.8, [[], [],
        tv(0.8, [[],
            tv(0.9, [[],
                tv(0.9, [element(title, [], [La Galatea]), [],
                tv(0.8, [[], [],
                tv(0.8, [[],
                    tv(0.9, [[],
                        tv(0.9, [element(title, [], [ Los trabajos ...
                    []]),
                []])])]),
           .....
        .....
```

Conclusions and Further Research

- Fuzzy Logic Programming LANGUAGE:
  The Multi-Adjoint Logic Programming approach
- FLOPER provides advanced options with “fuzzy taste”:
  - Compilation to Prolog code [RUN]: simplicity, transparency, complete evaluation of goals...
  - Low level representation [TREE]: traces, derivations, detailed computation steps (interpretive)...
  - Multi-adjoint lattices [LAT]: modeled in Prolog, rich/flexible behaviour of computations...
  - f.c.a’s with traces: [RUN] with appropriate [LAT] mimics [TREE] without extra computational effort

Conclusions and Further Research

However, more efforts are needed for.....

- Increasing expressivity: new computation rules, testing formal properties of fuzzy connectives, connection with other fuzzy languages, graphical interface, etc
- Including transformation techniques: optimization by Fold/Unfold, specialization by Partial Evaluation, thresholded tabulation, etc
- Ongoing application: we are currently implementing with FLOPER fuzzy variants of XPath/XQuery for the flexible manipulation of XML documents