Programming with Fuzzy Logic Rules by using the FLOPER Tool

Pedro J. Morcillo and Ginés Moreno

{pmorcillo,gmoreno}@dsi.uclm.es

http://www.dsi.uclm.es/investigacion/dect/FLOPERpage.htm

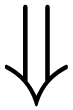
Department of Computing Systems

University of Castilla – La Mancha, SPAIN

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Multi-Adjoint Logic Programs

FUZZY LOGIC PROGRAMMING



FUZZY LOGIC

+

LOGIC PROGRAMMING

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Multi-Adjoint Logic Programs

$p(X)$	<prod	$q(X,Y)$ &godel $r(Y)$	with 0.8
$q(a,Y)$	<prod	$s(Y)$	with 0.7
$q(b,Y)$	<luka	$r(Y)$	with 0.8
$r(\_)$			with 0.6
$s(b)$			with 0.9

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Multi-Adjoint Logic Programs

$p(X)$	<prod	$q(X,Y)$ $\&\mathbf{godel}$ $r(Y)$	with 0.8
$q(a,Y)$	<prod	$s(Y)$	with 0.7
$q(b,Y)$	<luka	$r(Y)$	with 0.8
$r(\_)$			with 0.6
$s(b)$			with 0.9

$$p(X,TV0) \quad :- \quad q(X,Y,TV1), r(Y,TV2), and\_godel(TV1,TV2,TV3), and\_prod(0.8,TV3,TV0).$$

$$q(a,Y,TV0) \quad :- \quad s(Y,TV1), and\_prod(0.7,TV1,TV0).$$

$$q(b,Y,TV0) \quad :- \quad r(Y,TV1), and\_luka(0.8,TV1,TV0).$$

$$r(\_,0.6). \qquad s(b,0.9).$$

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- The system has been implemented with approximately 300 Sicstus Prolog clauses and it uses DCG's for defining parsing predicates

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- The system has been implemented with approximately 300 Sicstus Prolog clauses and it uses DCG's for defining parsing predicates
- The `prelude.pl` defines default aggregators

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- The system has been implemented with approximately 300 Sicstus Prolog clauses and it uses DCG's for defining parsing predicates
- The `prelude.pl` defines default aggregators
- Options for...
  - Loading a prolog file with extension ".pl"
  - Parsing a ".fpl" fuzzy program. The generated 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

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- Introducing a goal: `p(X) &godel r(a)`

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- Introducing a goal: $\texttt{p(X) \&godel r(a)}$
- Run: $[\texttt{Truth} = 0.3199, \texttt{X} = \texttt{b}]$ $[\texttt{Truth} = 0.48, \texttt{X} = \texttt{a}]$
  $\texttt{p(X, \_TV0), r(a, \_TV1), and\_godel(\_TV0, \_TV1, Truth)}$

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- Introducing a goal: $\texttt{p(X) \&godel r(a)}$
- Run: $[\texttt{Truth} = 0.3199, \texttt{X} = \texttt{b}]$ $[\texttt{Truth} = 0.48, \texttt{X} = \texttt{a}]$
  $\texttt{p(X, \_TV0), r(a, \_TV1), and\_godel(\_TV0, \_TV1, Truth)}$

  ..... by using a PROLOG-based, low level representation of the fuzzy code.....

Programming with Fuzzy Logic Rules by using the FLOPER Tool

The Prototype Tool

- Introducing a goal: `p(X) &godel r(a)`
- Run: $[\texttt{Truth} = 0.3199, \texttt{X} = \texttt{b}]$ $[\texttt{Truth} = 0.48, \texttt{X} = \texttt{a}]$
  `p(X, _TV0), r(a, _TV1), and_godel(_TV0, _TV1, Truth)`

  ..... by using a PROLOG-based, low level representation of the fuzzy code.....
- New DEBUGGING options.....
  - Depth. Fix the maximum evaluation level
  - Tree. Generate the associated evaluation tree (more precise information on computations)
  - Transformations. Open menu for F/U, PE,...

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

```
p(X) <prod q(X,Y) &godel r(Y) with 0.8
```

⇓

```
p(X,TV0) :- q(X,Y,TV1),r(Y,TV2),
  and_g(TV1,TV2,TV3),and_p(0.8,TV3,TV0).
```

⇓

```
rule(number(1),
     head(atom(pred(p,1),[var('X')])),
     impl(prod),body(and(godel, 2,
     [atom(pred(q,2),[var('X'),var('Y')]),
      atom(pred(r,1),[var('Y')])])),
     td(0.8)).
```

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 1: solutions in presence of infinite branches

R1: p(X) <prod q(X) with 0.9. R2: p(X) <godel r(X) with 0.8.
R3: q(X) <luka q(X) with 0.7. R4: r(a) with 0.6.

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 1: solutions in presence of infinite branches

R1: p(X) <prod q(X) with 0.9. R2: p(X) <godel r(X) with 0.8.
R3: q(X) <luka q(X) with 0.7. R4: r(a) with 0.6.

- TREE: Execution tree with depth 4 for goal `p(a)`

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 1: solutions in presence of infinite branches

R1: p(X) <prod q(X) with 0.9. R2: p(X) <godel r(X) with 0.8.
R3: q(X) <luka q(X) with 0.7. R4: r(a) with 0.6.

- TREE: Execution tree with depth 4 for goal `p(a)`

```
R0 < p(a), {} >
   R1 < &prod(0.9,q(a)), {X1/a} >
       R3 < &prod(0.9,&luka(0.7,q(a))), {X1/a,X7/a} >
           R3 < &prod(0.9,&luka(0.7,&luka(0.7,q(a)))), {X1/a,X7/a,..>
               R3 < &prod(0.9,&luka(0.7,&luka(0.7,&luka(0.7,q(a))))),
                                  ..............
   R2 < &godel(0.8,r(a)), {X2/a} >
       R4 < &godel(0.8,0.6), {X2/a} >
```

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 2: solutions even with unsuccessful atoms

  $\texttt{p(a)}$ <prod @aver$(1, \texttt{p(b)})$ with 0.9

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 2: solutions even with unsuccessful atoms

  $\texttt{p(a)}$ <prod @aver$(1, \texttt{p(b)})$ with 0.9

  ... remember that the associated Prolog clause is:

  $$\texttt{p(a, TV0)} :- \mathbf{p(b, \_TV1)}, \texttt{agr\_aver}(1, \texttt{\_TV1}, \texttt{\_TV2}), \texttt{and\_prod}(0.9, \texttt{\_TV2}, \texttt{TV0}).$$

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 2: solutions even with unsuccessful atoms

  `p(a)` <prod `@aver(1,p(b))` with 0.9

  ... remember that the associated Prolog clause is:

  ```
  p(a,TV0) :- p(b,_TV1),agr_aver(1,_TV1,_TV2),
                        and_prod(0.9,_TV2,TV0).
  ```

- TREE: Execution tree with depth 2 for goal `p(X)`

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Debugging capabilities of FLOPER

- Trace 2: solutions even with unsuccessful atoms

  $\texttt{p(a)}$ <prod @aver$(1, \texttt{p(b)})$ with 0.9

  ... remember that the associated Prolog clause is:

  $$\texttt{p(a, TV0)} :- \mathbf{p(b, \_TV1)}, \texttt{agr\_aver}(1, \texttt{\_TV1}, \texttt{\_TV2}), \texttt{and\_prod}(0.9, \texttt{\_TV2}, \texttt{TV0}).$$

- TREE: Execution tree with depth 2 for goal $\texttt{p(X)}$

```
R0 < p(X), {} >
        R1 < &prod(0.9,@aver(1,p(b))), {X/a} >
                R0 < &prod(0.9,@aver(1,0)), {X/a} >
```

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Conclusions and Further Research

- Fuzzy Logic Programming LANGUAGE:
  The Multi-Adjoint Logic Programming approach

Programming with Fuzzy Logic Rules by using the FLOPER Tool

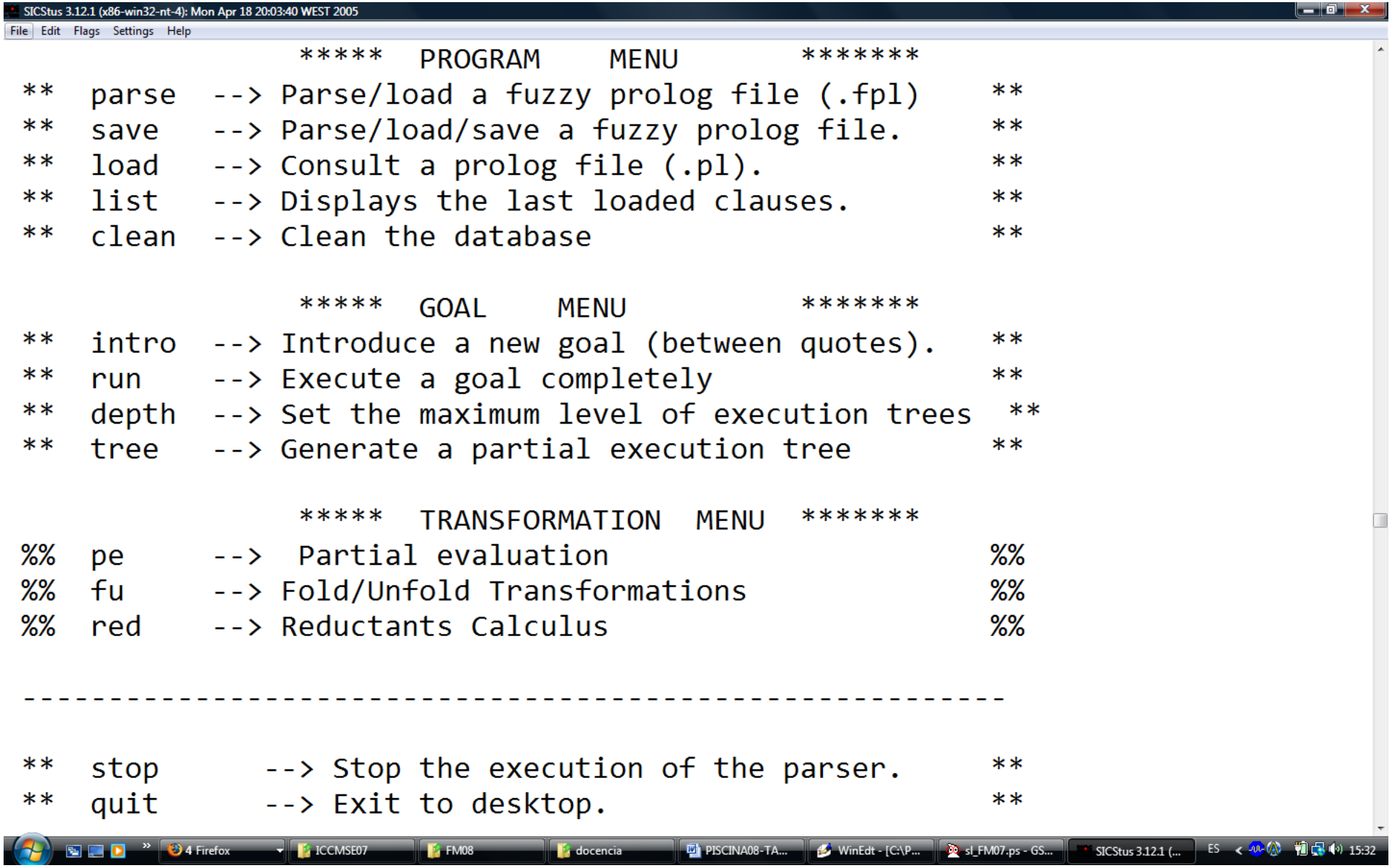
Conclusions and Further Research

- Fuzzy Logic Programming LANGUAGE:
  The Multi-Adjoint Logic Programming approach
- FLOPER provides two implementation techniques
  - Compilation to Prolog code [RUN]: simplicity, transparency, complete evaluation of goals
  - Low level representation [DEBUG]: detailed data, unfolding trees, powerful manipulations

Programming with Fuzzy Logic Rules by using the FLOPER Tool

Conclusions and Further Research

- Fuzzy Logic Programming LANGUAGE:
  The Multi-Adjoint Logic Programming approach
- FLOPER provides two implementation techniques
  - Compilation to Prolog code [RUN]: simplicity, transparency, complete evaluation of goals
  - Low level representation [DEBUG]: detailed data, unfolding trees, powerful manipulations
- More efforts are needed for increasing expressivity (new lattices, more computation rules, graphical interface) and including transformation techniques (optimization, specialization, reductants.....)


SICStus 3.12.1 (x86-win32-nt-4): Mon Apr 18 20:03:40 WEST 2005
File Edit Flags Settings Help
***** PROGRAM MENU *******
** parse --> Parse/load a fuzzy prolog file (.fpl) **
** save --> Parse/load/save a fuzzy prolog file. **
** load --> Consult a prolog file (.pl). **
** list --> Displays the last loaded clauses. **
** clean --> Clean the database **
***** GOAL MENU *******
** intro --> Introduce a new goal (between quotes). **
** run --> Execute a goal completely **
** depth --> Set the maximum level of execution trees **
** tree --> Generate a partial execution tree **
***** TRANSFORMATION MENU *******
%% pe --> Partial evaluation %%
%% fu --> Fold/Unfold Transformations %%
%% red --> Reductants Calculus %%
** stop --> Stop the execution of the parser. **
** quit --> Exit to desktop. **
4 Firefox
ICCMSE07
FM08
docencia
PISCINA08-TA...
WinEdt - [C:\P...
sl_FM07.ps - GS...
SICStus 3.12.1 (...
ES
15:32

SICStus 3.12.1 (x86-win32-nt-4): Mon Apr 18 20:03:40 WEST 2005

File Edit Flags Settings Help

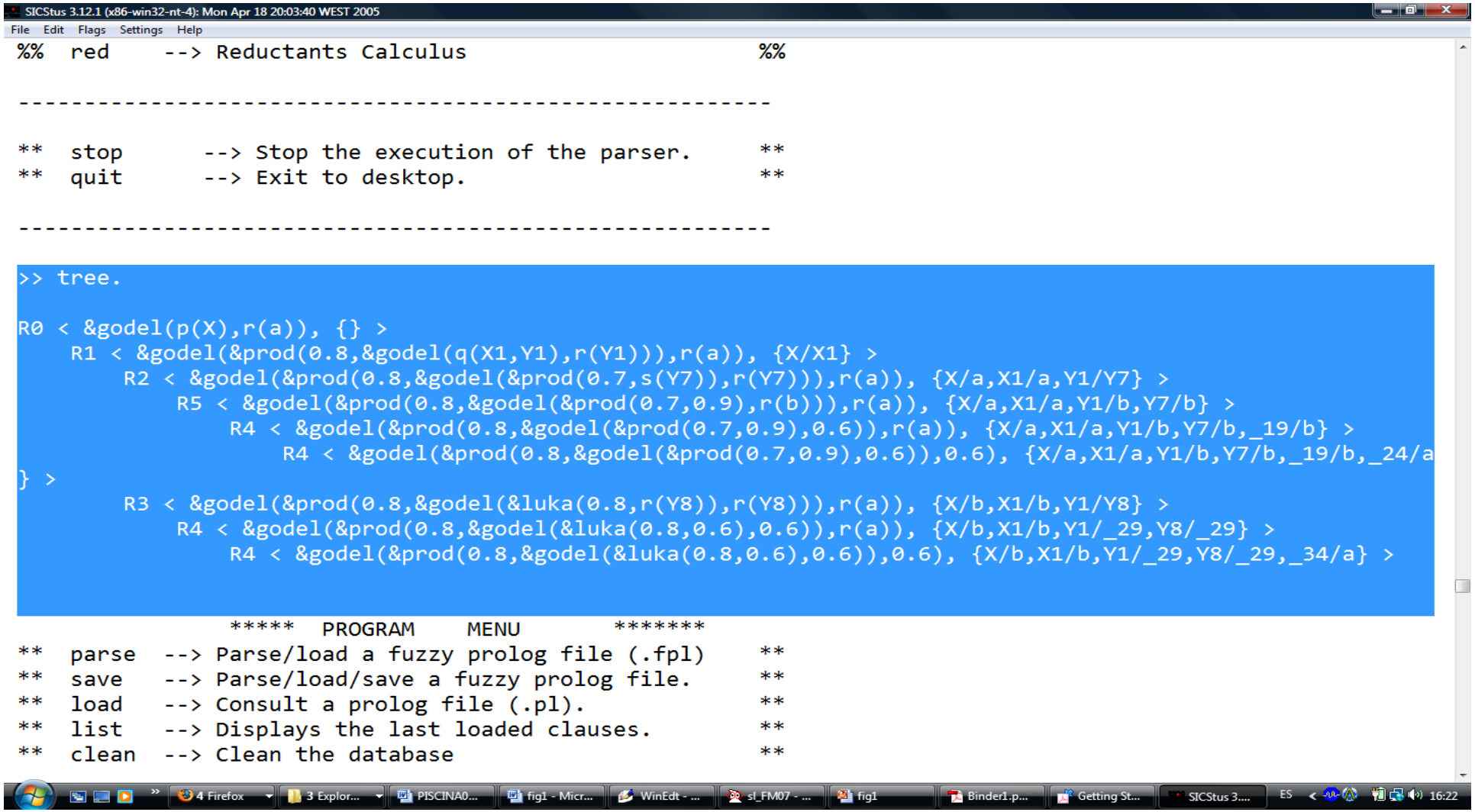
```
------------------------------------------------------------

>> list.

No loaded files.

ORIGINAL FUZZY-PROLOG CODE:
p(X) <prod q(X,Y) &godel r(Y) with 0.8.
q(a,Y) <prod s(Y) with 0.7.
q(b,Y) <luka r(Y) with 0.8.
r(_) with 0.6.
s(b) with 0.9.

GENERATED PROLOG CODE:
p(X,TV0):-q(X,Y,_TV2),r(Y,_TV3),and_godel(_TV2,_TV3,_TV5),and_prod(0.8,_TV5,T
V0).
q(a,Y,TV0):-s(Y,_TV1),and_prod(0.7,_TV1,TV0).
q(b,Y,TV0):-r(Y,_TV1),and_luka(0.8,_TV1,TV0).
r(_,0.6).
s(b,0.9).
```


SICStus 3.12.1 (x86-win32-nt-4): Mon Apr 18 20:03:40 WEST 2005
File Edit Flags Settings Help
%% red --> Reductants Calculus %%
** stop --> Stop the execution of the parser. **
** quit --> Exit to desktop. **
>> tree.
R0 < &godel(p(X),r(a)), {} >
R1 < &godel(&prod(0.8,&godel(q(X1,Y1),r(Y1))),r(a)), {X/X1} >
R2 < &godel(&prod(0.8,&godel(&prod(0.7,s(Y7)),r(Y7))),r(a)), {X/a,X1/a,Y1/Y7} >
R5 < &godel(&prod(0.8,&godel(&prod(0.7,0.9),r(b))),r(a)), {X/a,X1/a,Y1/b,Y7/b} >
R4 < &godel(&prod(0.8,&godel(&prod(0.7,0.9),0.6)),r(a)), {X/a,X1/a,Y1/b,Y7/b,_19/b} >
R4 < &godel(&prod(0.8,&godel(&prod(0.7,0.9),0.6)),0.6), {X/a,X1/a,Y1/b,Y7/b,_19/b,_24/a
} >
R3 < &godel(&prod(0.8,&godel(&luka(0.8,r(Y8)),r(Y8))),r(a)), {X/b,X1/b,Y1/Y8} >
R4 < &godel(&prod(0.8,&godel(&luka(0.8,0.6),0.6)),r(a)), {X/b,X1/b,Y1/_29,Y8/_29} >
R4 < &godel(&prod(0.8,&godel(&luka(0.8,0.6),0.6)),0.6), {X/b,X1/b,Y1/_29,Y8/_29,_34/a} >
***** PROGRAM MENU *******
** parse --> Parse/load a fuzzy prolog file (.fpl) **
** save --> Parse/load/save a fuzzy prolog file. **
** load --> Consult a prolog file (.pl). **
** list --> Displays the last loaded clauses. **
** clean --> Clean the database **
4 Firefox
3 Explor...
PISCINA0...
fig1 - Micr...
WinEdt - ...
sl_FM07 - ...
fig1
Binder1.p...
Getting St...
SICStus 3....
ES
16:22