Fuzzy Logic Programming and the FLOPER Environment

Ginés Moreno

DEC$^{\tau}$ Research Group, University of Castilla-La Mancha, Spain

Outline

1. **The FASILL language**
   - Fuzzy Logic Programming, FASILL and FLOPER
   - Complete Lattices and Similarity Relations
2. **Operational semantics of FASILL**
   - The Weak Unification Algorithm
   - Computational Steps and Derivations
3. **Declarative Semantics of FASILL**
   - Fuzzy Herbrand Interpretations
   - Fuzzy Herbrand Models
4. **Conclusions**

Outline

1. **The FASILL language**
   - Fuzzy Logic Programming, FASILL and FLOPER
   - Complete Lattices and Similarity Relations
2. **Operational semantics of FASILL**
   - The Weak Unification Algorithm
   - Computational Steps and Derivations
3. **Declarative Semantics of FASILL**
   - Fuzzy Herbrand Interpretations
   - Fuzzy Herbrand Models
4. **Conclusions**

Fuzzy Logic Programming and FASILL

- **Fuzzy Logic Programming** = Logic Prog. + Fuzzy Logic
- There is no a standard language. We have found two main approaches:
  1. SLD-resolution + **weak unification**
     Likelog *[Arcelli & Formato-99]* Bousi~Prolog *[Julián & Rubio-07]*
  2. **FUZZY inference** + syntactic unification
     f-Prolog *[Vojtas & Paulík-96]* MALP *[Medina et. al-01]*

The FASILL language is conceived for fusing both worlds

FASILL = **FUZZY inference + weak unification**

Fuzzy Aggregators and Similarity Into a Logic Language

- FASILL is a Fully Integrated Fuzzy Logic Language:
  - it copes with implicit/explicit truth degree annotations, a great variety of connectives
  - and unification by similarity.

- A FASILL program consists of a triple $\mathcal{P} = \langle \Pi, L, \mathcal{R} \rangle$, where:
  - Π is a set of program rules (richer than clauses).
  - L represents a lattice of truth degrees (beyond $\{true, false\}$).
  - $\mathcal{R}$ refers to a similarity relation (connecting pairs of symbols in the signature of $\mathcal{P}$ by means of truth degrees declared in L).

FASILL syntax: an example program

Program rules Π:

$$
\begin{array}{lcl}
vanguardist(hydropolis) & \leftarrow & 0.9. \\
elegant(ritz) & \leftarrow & 0.8. \\
close(hydropolis, taxi) & \leftarrow & 0.7. \\
good\_hotel(X) & \leftarrow & @aver(elegant(X), @very(close(X, metro))).
\end{array}
$$

Connective definitions in lattice $L = ([0,1], \leq)$:

$$[\![@_{\mathtt{aver}}]\!](x_1, x_2) \triangleq (x_1 + x_2)/2 \qquad [\![@_{\mathtt{very}}]\!](x) \triangleq x * x$$

Similarity relation $\mathcal{R}$:

$$
\begin{array}{lcl}
elegant/1 \sim vanguardist/1 & = & 0.6. \\
taxi \sim bus & = & 0.4. \\
bus \sim metro & = & 0.5.
\end{array}
$$

The Fuzzy Logic System FLOPER

Fuzzy LOgic Programming Environment for Research
(about 1000 Prolog clauses and DCG's for parsing predicates)

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

- Compiles FASILL code to Prolog code
- Accepts different complete lattices and similarity relations
- Available both on-line and installable versions
- Includes advanced options for
  - Running goals
  - Drawing derivation trees for debugging

http://dectau.uclm.es/floper/?q=sim

The on-line version of FLOPER: input boxes

Home

FLOPER Online

Testing:

FASILL program:

```
vanguardist(hydropolis) <- 0.9.
elegant(ritz)           <- 0.8.
close(hydropolis, taxi) <- 0.7.
good_hotel(X) <- @aver(elegant(X), @very(close(X, metro))).
```

Lattice:

```
:- dynamic agr_very/2,  and_godel/3, or_prod/3, or_godel/3, or_luka/3, agr_aver/3, pri_prod/3, pri_div/3,
pri_sub/3, pri_add/3, pri_min/3, pri_max/3.

member(X):-number(X), 0=<X,X=<1.

leq(X,Y):-X =< Y.

bot(0).
top(1).
```

Similarity equations:

```
elegant/1 ~ vanguardist/1 = 0.6.
metro ~ bus = 0.5.
bus ~ taxi = 0.4.
~tnorm = godel.
```

The on-line version of FLOPER: output windows

Fuzzy computed answers and execution tree:

F.c.a's for goal good_hotel(X)

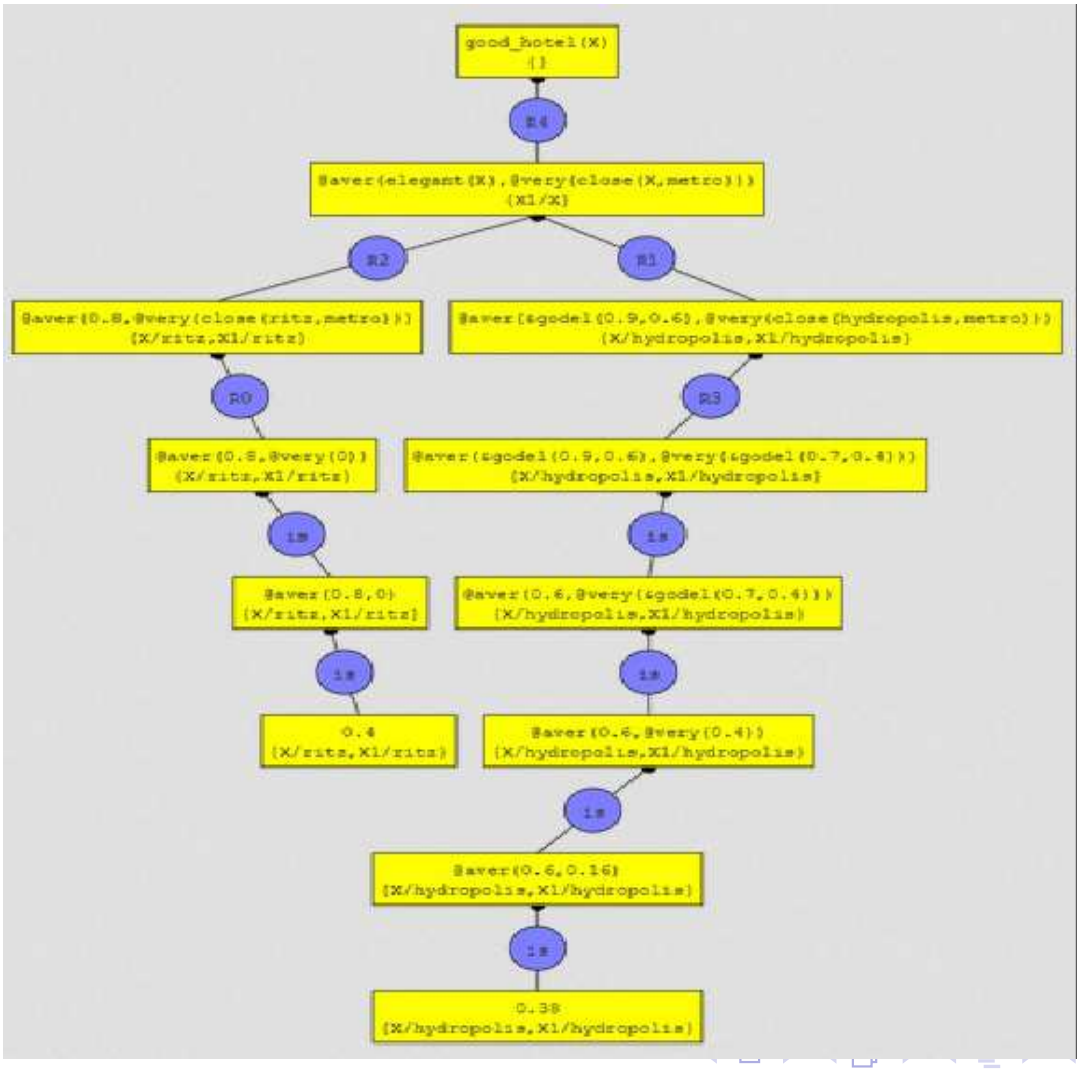
```
< 0.4, {X/ritz} >
< 0.38, {X/hydropolis} >
```

Derivation tree

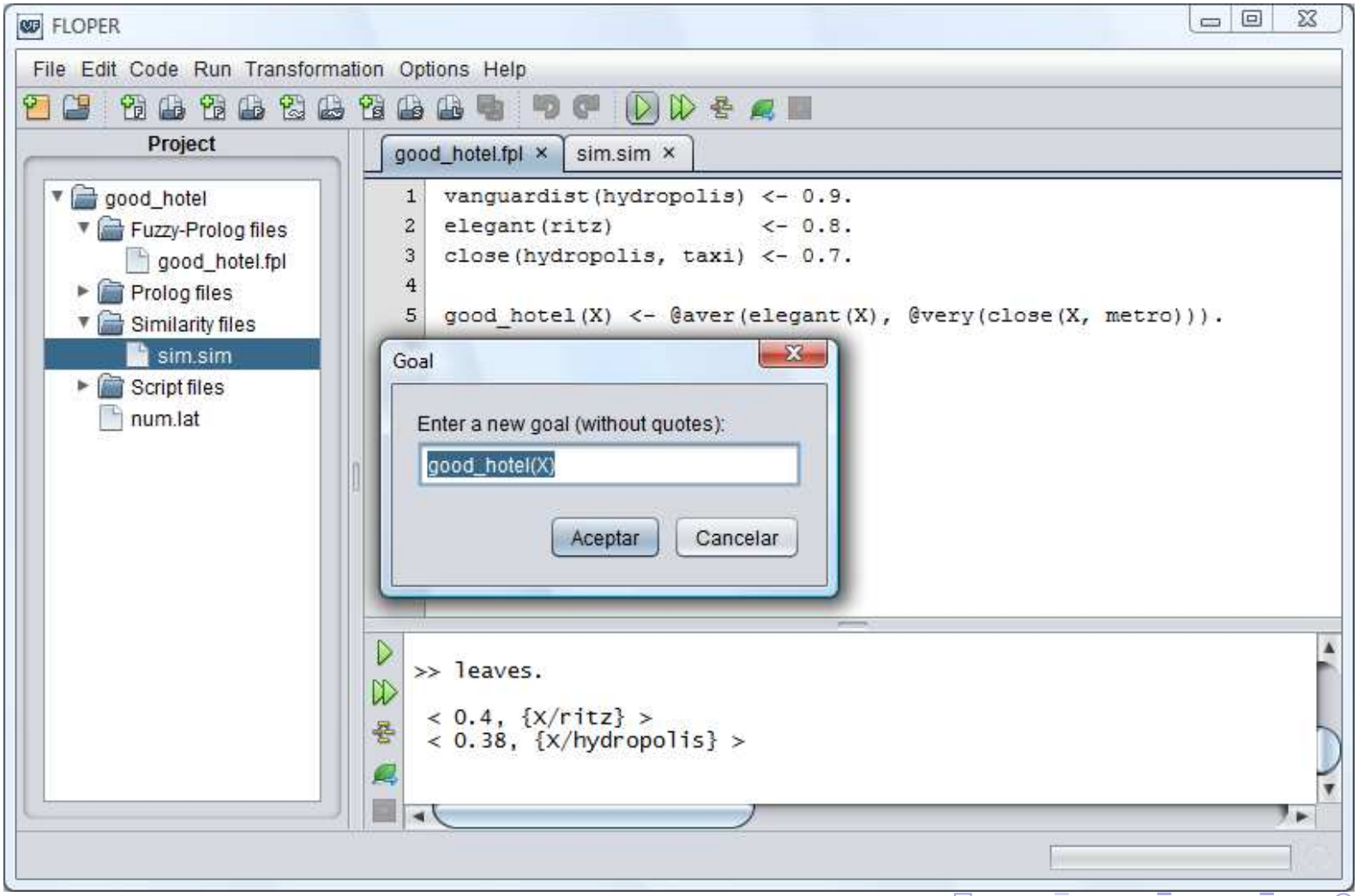
```
R0 < good_hotel(X), {} >
   R4 < @aver(elegant(X),@very(close(X,metro))), {X1/X} >
      R2 < @aver(0.8,@very(close(ritz,metro))), {X/ritz,X1/ritz} >
         R0 < @aver(0.8,@very(0)), {X/ritz,X1/ritz} >
            is < @aver(0.8,0), {X/ritz,X1/ritz} >
               is < 0.4, {X/ritz,X1/ritz} >
      R1 < @aver(&godel(0.9,0.6),@very(close(hydropolis,metro))), {X/hydropolis,X1/hydropolis} >
         R3 < @aver(&godel(0.9,0.6),@very(&godel(0.7,0.4))), {X/hydropolis,X1/hydropolis} >
            is < @aver(0.6,@very(&godel(0.7,0.4))), {X/hydropolis,X1/hydropolis} >
               is < @aver(0.6,@very(0.4)), {X/hydropolis,X1/hydropolis} >
                  is < @aver(0.6,0.16), {X/hydropolis,X1/hydropolis} >
                     is < 0.38, {X/hydropolis,X1/hydropolis} >
```

Top

Derivation tree depicted by FLOPER



Screenshot of FLOPER installed on a computer



Complete lattices for modeling truth-degrees

- FASILL is a PROLOG-like, first order language containing: variables, functions, predicates, and ... many connectives!!!

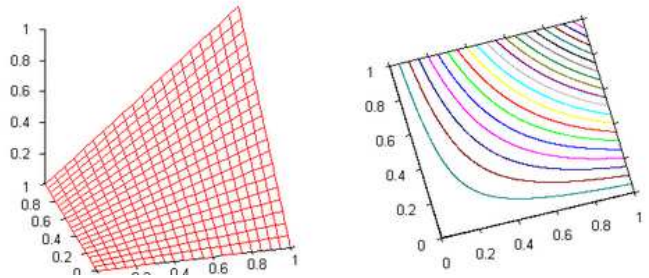
$$\begin{array}{llll} \&_1, & \&_2, & \ldots, & \&_k \quad \texttt{(conjunctions)} \\ |_1, & |_2, & \ldots, & |_l \quad \texttt{(disjunctions)} \\ @_1, & @_2, & \ldots, & @_n \quad \texttt{(aggregations)} \end{array}$$

- Instead of the crisp $\{true,\ false\}$, it uses any complete lattice $\langle L, \preceq \rangle$ to model truth degrees as well as fuzzy connectives

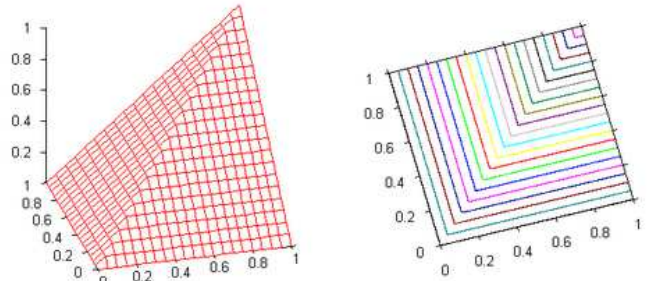
- Some well-known connectives acting on lattice $([0,1], \leq)$ are the ones defined on the fuzzy logics of *Product*, *Gödel* and *Łukasiewicz*:

Examples of t-norms (Product, Gödel and Łukasiewicz)

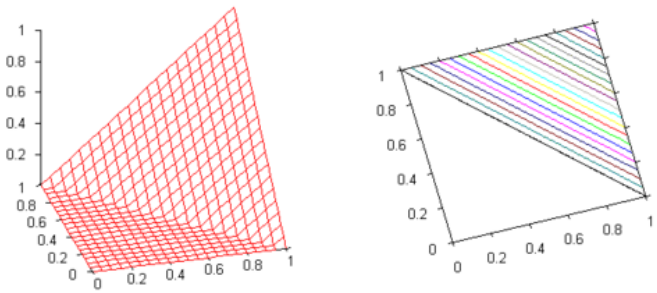
$$[\![\&_{\mathrm{P}}]\!](x,y) \triangleq x * y$$



$$[\![\&_{\mathrm{G}}]\!](x,y) \triangleq \min(x,y)$$

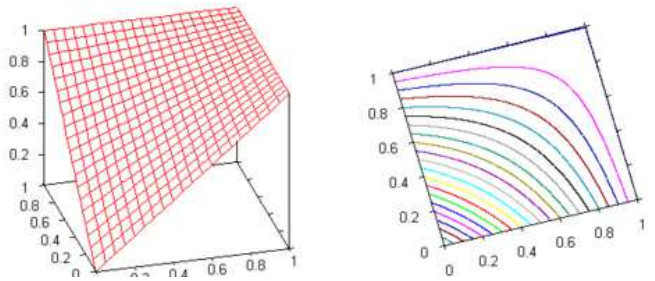


$$[\![\&_{\mathrm{L}}]\!](x,y) \triangleq \max(0, x+y-1)$$

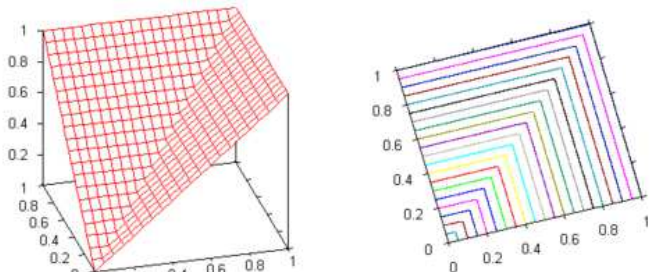


Examples of t-conorms (Product, Gödel and Łukasiewicz)

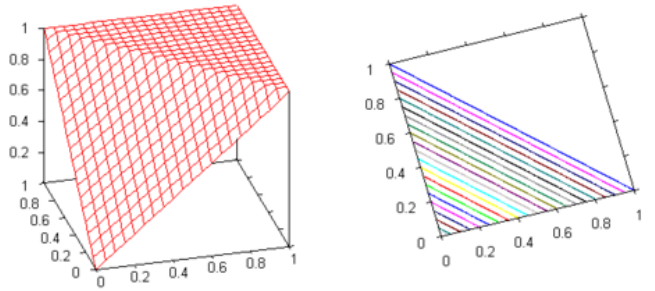
$$[\![|_{\mathtt{P}}]\!](x,y) \triangleq x + y - xy$$



$$[\![|_{\mathtt{G}}]\!](x,y) \triangleq \max(x,y)$$



$$[\![|_{\mathtt{L}}]\!](x,y) \triangleq \min(x+y,1)$$



Similarity Relations

- Similarity relations can be seen as an extension of the classical notion of equivalence relation
- A similarity relation $\mathcal{R}$ is a fuzzy binary relation on a given domain $\mathcal{U}$ fulfilling these properties:

  Reflexive $\quad \mathcal{R}(x,x) = \top, \quad \forall x \in \mathcal{U}$

  Symmetric $\quad \mathcal{R}(x,y) = \mathcal{R}(y,x), \quad \forall x,y \in \mathcal{U}$

  Transitive $\quad \mathcal{R}(x,z) \geq \mathcal{R}(x,y) \wedge \mathcal{R}(y,z), \quad \forall x,y,z \in \mathcal{U}$
- In classical Logic Programming different syntactic symbols represent distinct information.
- This restriction can be relaxed by introducing a similarity relation $\mathcal{R}$ on the alphabet of a first order language.

Similarity Relations

- A similarity relation $\mathcal{R}$ is coded with syntax:

$$\begin{array}{lll} \langle Relation \rangle & ::= & \langle Sim \rangle \ \langle Relation \rangle \mid \langle Sim \rangle \\ \langle Sim \rangle & ::= & \langle Id_f \rangle [\text{`/'} \ \langle Int_n \rangle] \ \text{`}\sim\text{'} \ \langle Id_g \rangle [\text{`/'} \ \langle Int_n \rangle] \ \text{`}=\text{'} \ \langle r \rangle \ \text{`.'} \\ & & \mid \ \text{`}\sim\text{'} \ \text{`tnorm'} \ \text{`}=\text{'} \ \langle tnorm \rangle \ \text{`.'} \end{array}$$

Example

```
~tnorm = godel.
elegant/1 ~ vanguardist/1 = 0.6.
metro ~ bus = 0.5.
bus ~ taxi = 0.4.
```

Similarity Relations

Example (reflexive, symmetric, transitive closure)

	elegant	vanguardist	metro	bus	taxi
elegant	1	0.6	0	0	0
vanguardist	0.6	1	0	0	0
metro	0	0	1	0.5	0.4
bus	0	0	0.5	1	0.4
taxi	0	0	0.4	0.4	1

- $\mathcal{R}$ can be extended to terms by induction:
  - $\hat{\mathcal{R}}(x,x) = \mathcal{R}(x,x) = \top$, if x is a variable
  - $\hat{\mathcal{R}}(f(t_1,\ldots,t_n), g(s_1,\ldots,s_n)) = \mathcal{R}(f,g) \wedge (\bigwedge_{i=1}^{n} \hat{\mathcal{R}}(t_i,s_i))$
- So, $\hat{\mathcal{R}}(elegant(taxi), vanguardist(metro)) = 0.6 \wedge 0.4$, because $\mathcal{R}(elegant, vanguardist) = 0.6$ and $\mathcal{R}(taxi, metro) = 0.4$.

Outline

1 The FASILL language
- Fuzzy Logic Programming, FASILL and FLOPER
- Complete Lattices and Similarity Relations

2 Operational semantics of FASILL
- The Weak Unification Algorithm
- Computational Steps and Derivations

3 Declarative Semantics of FASILL
- Fuzzy Herbrand Interpretations
- Fuzzy Herbrand Models

4 Conclusions

The Weak Unification Algorithm

- FASILL inherits from Bousi~Prolog a weak unification algorithm based on similarity relations.

- The weak unification algorithm:
  - $\{f(t_1, \ldots, t_n) \approx g(s_1, \ldots, s_n)\}$ weakly unifies (at a level λ) iff $\mathcal{R}(f, g) > \lambda$ and $\{t_1 \approx s_1, \ldots, t_n \approx s_n)\}$ weakly unifies (at a level λ).

  Output: a weak mgu of level λ, which is a substitution plus an approximation degree.

- Note that it computes a representative of a class of wmgus.

The Weak Unification Algorithm

Example (Find a wmgu (of level 0.3) for $f(h(X)), k(Y))$ and $g(Z, j(Y)))$

Assume $\mathcal{R}(f, g) = 0.8, \mathcal{R}(j, k) = 0.3, \ldots$

Unification problem	Weak Unifier	Degree
$\{f(h(X)), k(Y)) \approx g(Z, j(Y))\}$	$\{\}$	1
$\{h(X) \approx Z, k(Y) \approx j(X)\}$	$\{\}$	$1 \wedge 0.8$
$\{k(Y) \approx j(X)\}$	$\{Z/h(X)\}$	0.8
$\{Y \approx X\}$	$\{Z/h(X)\}$	$0.8 \wedge 0.3$
$\{\}$	$\{Z/h(X), Y/X\}$	0.3

Operational semantics

FASILL Operational semantics

Fuzzy inference (MALP) + weak unification (Bousi~Prolog).

- It is a backward-reasoning method which is modeled as a state transition system where:
  - State: pair with form $\langle goal;\ substitution \rangle$
  - Input (goal): for instance, $\langle good\_hotel(X);\ id \rangle$
  - Output (fuzzy computed answer): final state of the form $\langle truth\_degree;\ substitution \rangle$
  - Transition relation: based on the following three rules
    Successful steps ($\rightarrow_{SS}$) Interpretive steps ($\rightarrow_{IS}$)
    Failure steps ($\rightarrow_{FS}$)

Transition relation $\rightarrow_{SS}$

SUCCESSFUL STEP $\rightarrow_{SS}$

$\langle \mathcal{Q}[A]; \sigma \rangle \rightarrow_{SS} \langle (\mathcal{Q}[A/\&(\mathcal{B}, r)])\theta; \sigma\theta \rangle$ if

(1) A is the selected atom in $\mathcal{Q}$,
(2) $A' \leftarrow \mathcal{B}$ is a rule in the program $\mathcal{P}$
(3) $\langle \theta, r \rangle = wmgu(\{A' \approx A\})$

Example

Let $\mathtt{p(a)} \leftarrow \mathtt{p(f(a))}$ be a rule, $\mathcal{R}(p, q) = 0.9$, and $\&_{\mathtt{prod}}$ is the t-norm used to propagate similarities

$\langle \&_{\mathtt{godel}}(\mathtt{q(X)}, \mathtt{s(X)}); \mathtt{id} \rangle \rightarrow_{SS}$
$\langle \&_{\mathtt{godel}}(\&_{\mathtt{prod}}(0.9, \mathtt{p(f(a))}), \mathtt{s(a)}); \{\mathtt{X/a}\} \rangle$

Transition relation $\rightarrow_{FS}$

FAILURE STEP $\rightarrow_{FS}$

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

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

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

Example

This case is introduced to cope with (possible) unsuccessful admissible derivations. For instance, with a program

$\texttt{p(a)} \leftarrow 0.6$

we have : $\langle \texttt{@aver}(1, \texttt{p(b)}); \texttt{id} \rangle \rightarrow_{FS} \langle \texttt{@aver}(1, 0); \texttt{id} \rangle$

Transition relation $\rightarrow_{IS}$

INTERPRETIVE STEP $\rightarrow_{IS}$

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

where $\dot{@}$ is the truth function of connective $@$ in the complete lattice associated to the program $\mathcal{P}$

Example

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

$\langle \&_{\texttt{godel}}(\&_{\texttt{prod}}(0.9, 0.5), 0.7); \{X/a\} \rangle \rightarrow_{IS}$
$\langle \&_{\texttt{godel}}(0.45, 0.7); \{X/a\} \rangle$

Example of a complete derivation

$p(X) \leftarrow \&_{\texttt{godel}}(q(X), @_{\texttt{aver}}(r(X), s(X)))$. $\quad p/1 \sim w/1 = 0.9.$

$q(a) \leftarrow 0.8.$ $\quad \sim tnorm = \&_{prod}$

$r(X) \leftarrow 0.7.$

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

$\langle \&_{\texttt{prod}}(0.9, \&_{\texttt{godel}}(\underline{q(X_1)}, @_{\texttt{aver}}(r(X1), s(X1)))); \{X/X_1\} \rangle \rightarrow_{SS}^{\mathcal{R}_2}$

$\langle \&_{\texttt{prod}}(0.9, \&_{\texttt{godel}}(0.8, @_{\texttt{aver}}(\underline{r(a)}, s(a)))); \{X/a\} \rangle \rightarrow_{SS}^{\mathcal{R}_3}$

$\langle \&_{\texttt{prod}}(0.9, \&_{\texttt{godel}}(0.8, @_{\texttt{aver}}(0.7, \underline{s(a)}))); \{X/a\} \rangle \rightarrow_{FS}$

$\langle \&_{\texttt{prod}}(0.9, \&_{\texttt{godel}}(0.8, \underline{@_{\texttt{aver}}(0.7, 0)})); \{X/a\} \rangle \rightarrow_{IS}$

$\langle \&_{\texttt{prod}}(0.9, \underline{\&_{\texttt{godel}}(0.8, 0.35)}); \{X/a\} \rangle \rightarrow_{IS}$

$\langle \&_{\texttt{prod}}(0.9, 0.35); \{X/a\} \rangle \rightarrow_{IS}$

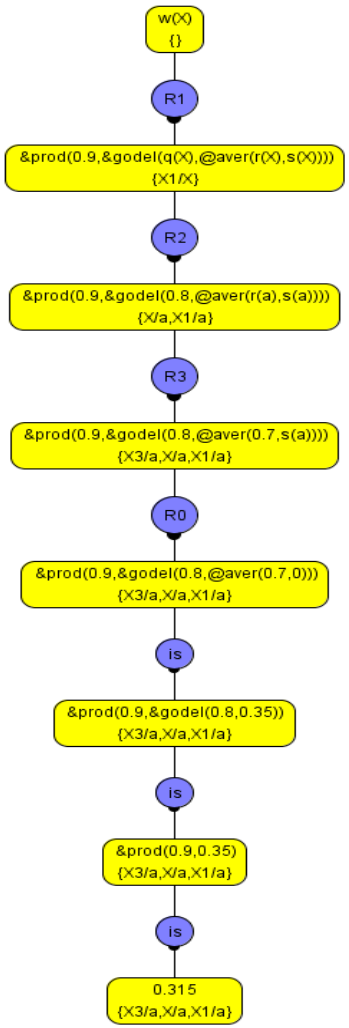
$\langle 0.315; \{X/a\} \rangle$

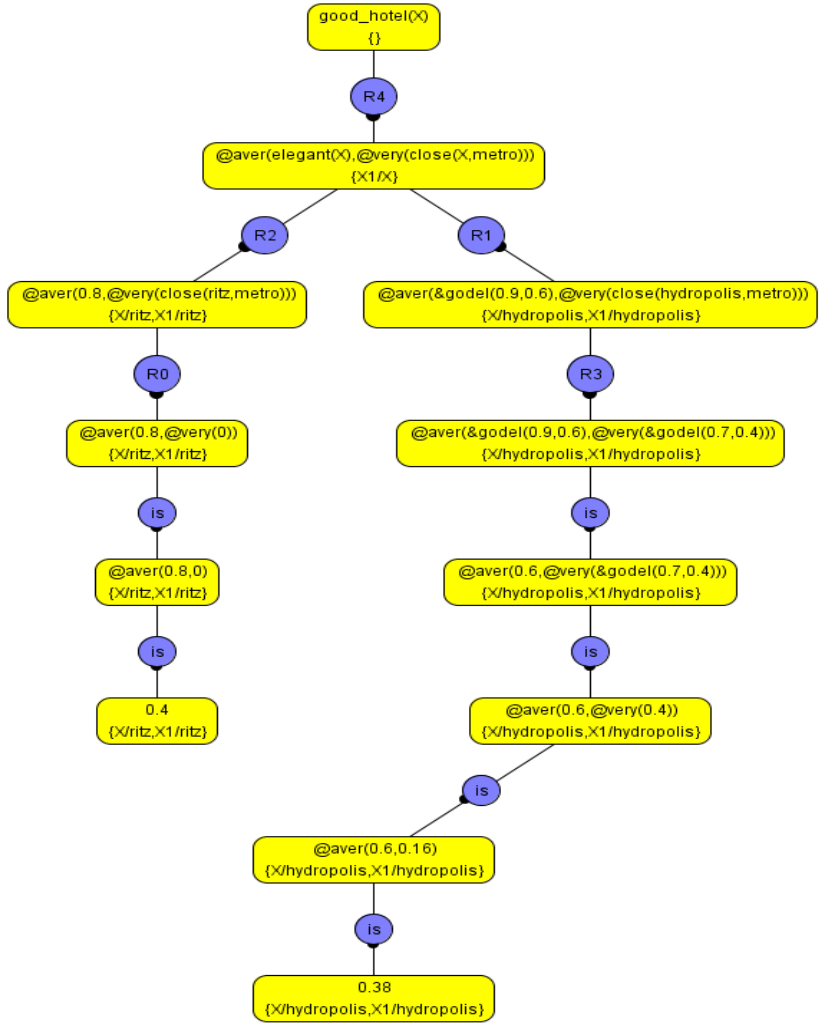
So, the f.c.a for this complete derivation is $\langle 0.315; \{X/a\} \rangle$

Formats for exporting derivation trees (txt, xml and png)

```
R0 <w(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}>
    R0 <&prod(0.9,&godel(0.8,@aver(0.7,0))),{X/a,X1/a}>
     is <&prod(0.9,&godel(0.8,0.35)),{X/a,X1/a}>
      is <&prod(0.9,0.35),{X/a,X1/a}>
       is <0.315,{X/a,X1/a}>
```


−<node>
<rule>R0</rule>
<goal>w(X)</goal>
<substitution>{}</substitution>
−<children>
−<node>
<rule>R1</rule>
<goal>and_prod(0.9,and_godel(q(X),agr_aver(r(X),s(X))))</goal>
<substitution>{X1/X}</substitution>
−<children>
−<node>
<rule>R2</rule>
<goal>and_prod(0.9,and_godel(0.8,agr_aver(r(a),s(a))))</goal>
<substitution>{X/a,X1/a}</substitution>
−<children>
−<node>
<rule>R3</rule>
<goal>and_prod(0.9,and_godel(0.8,agr_aver(0.7,s(a))))</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
−<node>
<rule>R0</rule>
<goal>and_prod(0.9,and_godel(0.8,agr_aver(0.7,0)))</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
−<node>
<rule>is</rule>
<goal>and_prod(0.9,and_godel(0.8,0.35))</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
−<node>
<rule>is</rule>
<goal>and_prod(0.9,and_godel(0.8,0.35))</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
−<node>
<rule>is</rule>
<goal>and_prod(0.9,0.35)</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
−<node>
<rule>is</rule>
<goal>0.315</goal>
<substitution>{X3/a,X/a,X1/a}</substitution>
−<children>
</children>
</node>
</children>
</node>
</children>
</node>
</children>
</node>
</children>
</node>
</children>
</node>


w(X)
{}
R1
&prod(0.9,&godel(q(X),@aver(r(X),s(X))))
{X1/X}
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))))
{X3/a,X/a,X1/a}
R0
&prod(0.9,&godel(0.8,@aver(0.7,0)))
{X3/a,X/a,X1/a}
is
&prod(0.9,&godel(0.8,0.35))
{X3/a,X/a,X1/a}
is
&prod(0.9,0.35)
{X3/a,X/a,X1/a}
is
0.315
{X3/a,X/a,X1/a}


good_hotel(X)
{}
R4
@aver(elegant(X),@very(close(X,metro)))
{X1/X}
R2
R1
@aver(0.8,@very(close(ritz,metro)))
{X/ritz,X1/ritz}
@aver(&godel(0.9,0.6),@very(close(hydropolis,metro)))
{X/hydropolis,X1/hydropolis}
R0
R3
@aver(0.8,@very(0))
{X/ritz,X1/ritz}
@aver(&godel(0.9,0.6),@very(&godel(0.7,0.4)))
{X/hydropolis,X1/hydropolis}
is
is
@aver(0.8,0)
{X/ritz,X1/ritz}
@aver(0.6,@very(&godel(0.7,0.4)))
{X/hydropolis,X1/hydropolis}
is
is
0.4
{X/ritz,X1/ritz}
@aver(0.6,@very(0.4))
{X/hydropolis,X1/hydropolis}
is
@aver(0.6,0.16)
{X/hydropolis,X1/hydropolis}
is
0.38
{X/hydropolis,X1/hydropolis}

Outline

1 The FASILL language
- Fuzzy Logic Programming, FASILL and FLOPER
- Complete Lattices and Similarity Relations

2 Operational semantics of FASILL
- The Weak Unification Algorithm
- Computational Steps and Derivations

3 Declarative Semantics of FASILL
- Fuzzy Herbrand Interpretations
- Fuzzy Herbrand Models

4 Conclusions

Fuzzy Herbrand Interpretations

Definition (Fuzzy Herbrand Interpretation)

A Fuzzy Herbrand interpretation of $\mathcal{P} = \langle \Pi, \mathcal{R}, L \rangle$ is a mapping $\mathcal{I} : \mathcal{B}_\mathcal{P} \to L$, where $\mathcal{B}_\mathcal{P}$ is the Herbrand base of $\mathcal{P}$.

- $\mathcal{I}$ is a L-fuzzy subset: $\{A|\alpha : A \in \mathcal{B}_\mathcal{P} \text{ and } \mathcal{I}(A) = \alpha\}$
- The set of fuzzy Herbrand interpretations $\mathcal{H}$, with the inclusion order $\subseteq$:

$$\mathcal{I}_1 \subseteq \mathcal{I}_2 \iff \mathcal{I}_1(A) \leq \mathcal{I}_2(A), \forall A \in \mathcal{B}_\mathcal{P},$$

  is a complete lattice.
- Top (and bottom) element of $\mathcal{H}$:

$$\mathcal{I}_\top = \{A|\top : A \in \mathcal{B}_\mathcal{P}\} \quad (\mathcal{I}_\bot = \{A|\bot : A \in \mathcal{B}_\mathcal{P}\})$$

.

Fuzzy Herbrand Interpretations

- Extension of $\mathcal{I}$ to the set of ground formulae:

$$\mathcal{I}(\varsigma(F_1, \ldots, F_n)) = \dot{\varsigma}(\mathcal{I}(F_1), \ldots, \mathcal{I}(F_n))$$

- Extension of $\mathcal{I}$ to the set of (closed and universally quantified) formulae $\forall A$:

$$\mathcal{I}(\forall A) = inf\{\mathcal{I}(A\theta) : A\theta \text{ is a ground instance of } A\}$$

Abuse of language: we use the same symbol for the Herbrand interpretation and its extensions.

Fuzzy Herbrand Model

Definition

A Herbrand interpretation $\mathcal{I}$ satisfies or is a *Herbrand model* of a conditional formula $H \leftarrow \mathcal{B}$ if, and only if, it verifies that $\mathcal{I}(H\vartheta) \geq \mathcal{I}(\mathcal{B}\vartheta)$, for all assignement ϑ.

- A direct naive translation of the classical concept of Herbrand model of a program $\mathcal{P}$ (as a Herbrand interpretation $\mathcal{I}$ which satisfies all the rules in $\mathcal{P}$) does not work.

Fuzzy Herbrand Model

Example

Let $\mathcal{P} = \langle \Pi, \mathcal{R}, L \rangle$ be a FASILL program, where $\Pi = \{p(a)\}$, $\mathcal{R} = \{\mathcal{R}(a, b) = 0.8\}$ and $L = [0, 1]$.

- With the naive definition the interpretation $\{p(a)|1, p(b)|0\}$ is wrongly considered a model $\mathcal{P}$.
- However, a true model is $\{p(a)|1, p(b)|0.8\}$.

Remark: $p(a)$ is completely true and, because a is close to b with 0.8, $p(b)$ should be true with 0.8 also.

Extended Program

Definition

Given $\mathcal{P} = \langle \Pi, \mathcal{R}, L \rangle$, the set of rules which are similar to the rules in the linearization of Π, $lin(\Pi)$, for a level λ,

$$\mathcal{K}_\lambda(\Pi) = \{H \leftarrow \alpha \wedge \mathcal{B} : H' \leftarrow \mathcal{B} \in lin(\Pi), \mathcal{R}(H, H') = \alpha \geq \lambda\}$$

Example

For the program $\mathcal{P}$ of the last example

$$\mathcal{K}_{0.8}(\Pi) = \{p(a), p(b) \leftarrow 0.8\}$$

- Note that the interpretation $\{p(a)|1, p(b)|0\}$ does not satisfies all the rules in $\mathcal{K}_{0.8}(\Pi)$.
- Contrary to the interpretation $\{p(a)|1, p(b)|0.8\}$.

Linearization Process

- Linearize rules which have nonlinear atoms on their heads by replacing:
  - n_i multiple occurrences of the same variable x_i by new fresh variables $y_k (1 \leq k \leq n_i)$; and
  - a set of similarity constrains $x_i \sim y_k$ (with $1 \leq k \leq n_i$).
- By definition $\mathcal{I}(s \sim t) = \mathcal{R}(s, t)$, for all $\mathcal{I}$.

Example

Assume a FASILL program $\mathcal{P}$ with only one rule

$$R = p(x, x) \leftarrow q(x).$$

Then,

$$lin(R) \;=\; p(y_1, y_2) \leftarrow x \sim y_1 \wedge x \sim y_2 \wedge q(x)$$

Need for the Linearization Process

- When the rules in a program are not linear we may loose information in the presence of a similarity relation.

Example (Last program $\mathcal{P}$ equipped with $\mathcal{R}(a,b) = \beta \in L$)

Then, all ground rules which are similar to (ground instances of) R can be obtained from $lin(R) = p(y_1, y_2) \leftarrow x \sim y_1 \wedge x \sim y_2 \wedge q(x)$:

- $p(a,b) \leftarrow a \sim a \wedge a \sim b \wedge q(a)$ $(\equiv p(a,b) \leftarrow \beta \wedge q(a))$,
- $p(b,a) \leftarrow b \sim a \wedge b \sim b \wedge q(b)$ $(\equiv p(b,a) \leftarrow \beta \wedge q(b))$.

These ground instances are lost if we instantiate the rule R directly. In this case, we only obtain:

- $p(a,a) \leftarrow q(a)$ and $p(b,b) \leftarrow q(b)$

corresponding to the unique assignments $\{x/a\}$ and $\{x/b\}$.

Herbrand Model of Level λ of a Program

Definition (Herbrand Model of level λ)

Let $\lambda \in L$ be a cut value. A Herbrand interpretation $\mathcal{I}^\lambda$ is a *Herbrand model of level λ* (or λ-model) of the program $\mathcal{P}$ iff it is a model for each rule $H \leftarrow \alpha \wedge \mathcal{B}$ in $\mathcal{K}_\lambda(\Pi)$.

($\alpha \wedge \mathcal{I}^\lambda(\mathcal{B}\vartheta) \leq \mathcal{I}^\lambda(H\vartheta)$ for each assignment ϑ and rule in $\mathcal{K}_\lambda(\Pi)$).

- We control the level of closeness we want to consider in the resolution of a problem by means of the fixed cut value λ.
- The cut value λ imposes a limit from which a truth degree or approximation degree is considered as "true".

Least Herbrand Model of Level λ of a Program

As usual we give the declarative meaning of a program by selecting the simplest model of that program.

Definition (Least Herbrand Model of level λ)

The interpretation $\mathcal{I}_{\mathcal{P}}^{\lambda} = \inf\{\mathcal{I}_j^{\lambda} : \mathcal{I}_j^{\lambda} \text{ is a } \lambda\text{-model of } \mathcal{P}\}$ is called the *least fuzzy Herbrand model* (or least λ-model) of $\mathcal{P}$.

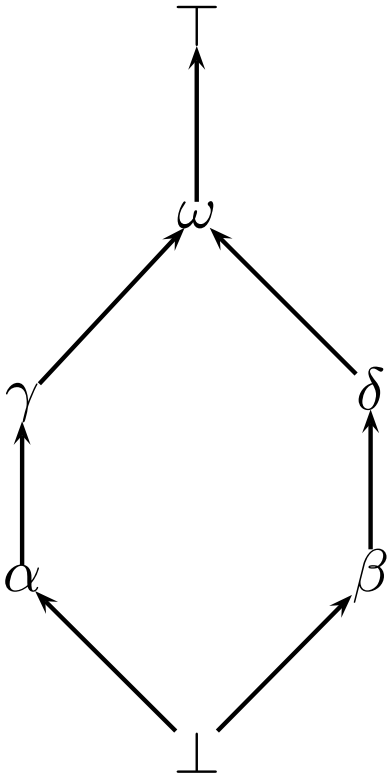
Theorem

The Herbrand interpretation $\mathcal{I}_{\mathcal{P}}^{\lambda} = \inf\{\mathcal{I}_j^{\lambda} : \mathcal{I}_j^{\lambda}$ is a λ-model of $\mathcal{P}\}$ is the least Herbrand model of level λ for $\mathcal{P}$.

Least Fuzzy Herbrand Model

	$\mathcal{I}_{\mathcal{P}}$
vanguardist(hydropolis)	0.9
vanguardist(ritz)	0.6
elegant(hydropolis)	0.6
elegant(ritz)	0.8
close(hydropolis, taxi)	0.7
close(hydropolis, metro)	0.4
close(hydropolis, bus)	0.5
good_hotel(hydropolis)	0.38
good_hotel(ritz)	0.4

Least Fuzzy Herbrand Model



$$\Pi = \begin{cases} R_1: & r(b) & \leftarrow & \delta \\ R_2: & q(a) & \leftarrow & \gamma \\ R_3: & p(x) & \leftarrow & q(x) \vee_{\mathbb{G}} r(x) \end{cases}$$

$\mathcal{R}(a, a') = \alpha$ and $\mathcal{R}(b, b') = \beta$

$\dot{\vee}_{\mathbb{G}}(x, y) = \sup\{x, y\}$

	$p(a)$	$p(a')$	$p(b)$	$p(b')$	$q(a)$	$q(a')$	$r(b)$	$r(b')$
$\mathcal{I}_{\mathcal{P}}$	γ	α	δ	β	γ	α	δ	β

Outline

1 The FASILL language
- Fuzzy Logic Programming, FASILL and FLOPER
- Complete Lattices and Similarity Relations

2 Operational semantics of FASILL
- The Weak Unification Algorithm
- Computational Steps and Derivations

3 Declarative Semantics of FASILL
- Fuzzy Herbrand Interpretations
- Fuzzy Herbrand Models

4 Conclusions

Conclusions

Our records in the development of the fully integrated fuzzy logic language FASILL:

- DESIGN: Syntax coping with truth degrees and similarities, as well as operational, declarative and fix-point semantics.
- IMPLEMENTATION: The FLOPER system is able to compile, execute and debug FASILL programs and it has been used for developing real-world applications.