IARIA Congress 2024 June 30, 2024 to July 04, 2024 - Porto, Portugal

The 2024 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications

Symbolic Unfolding of Similarity-based Fuzzy Logic Programs

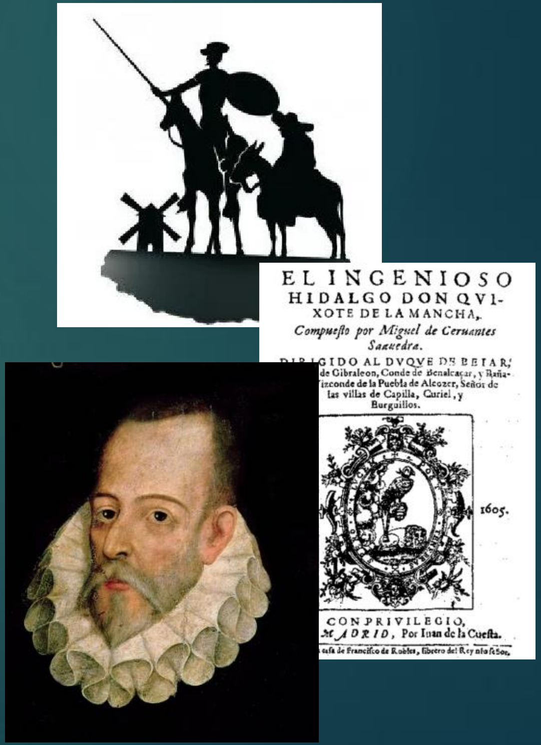
Ginés Moreno & José Antonio Riaza

Department of Computing Systems

University of Castilla-La Mancha

02071 Albacete (Spain)

Email: Gines.Moreno@uclm.es



EL INGENIOSO
HIDALGO DON QVI-
XOTE DE LA MANCHA,
Compueſto por Miguel de Ceruantes
Saauedra.
DIRIGIDO AL DVQVE DE BEIAR,
de Gibraleon, Conde de Benalcaçar, y Baña-
Vizconde de la Puebla de Alcozer, Señor de
las villas de Capilla, Curiel, y
Burguillos.

1605.

CON PRIVILEGIO,
MADRID, Por Iuan de la Cueſta.
caſa de Franciſco de Robles, librero del Rey nro ſeñor.

DEC-TAU research group

DECLARATIVE PROGRAMMING & AUTOMATIC PROGRAM TRANSFORMATION

- Founded in 2000 by Ginés Moreno and Pascual Julián
- History: 10 researchers (José Antonio Riaza) and 6 research projects
- Two decades with FASILL: *“Fuzzy Aggregators and Similarity Into a Logic Language”*
  - Paradigm Integration: Fuzzy Logic Programming (symbolic extensions)
  - Transformations: Partial Evaluation, Folding, Unfolding and Tuning
  - Applications: semantic web, neural networks, cloud computing…

Our work in 10 slides

- MOTIVATION I-II-III
- UNFOLDING I-II-III-IV-V-VI
- CONCLUSIONS

MOTIVATION I: unfolding pure logic programms: [Tamaki & Sato, 1984]

General idea: *replace a program rule by the set of new rules obtained after applying a computational step (in all posible ways) on its body and appropriately instantiating its head.*

- Original PROLOG program: P ={ p(X):-q(X). q(a).}
- Unfolded PROLOG program: P' ={ p(a). q(a).}
- Goal p(a) is evaluated to true in both programs:
  - by means of TWO resolution steps in P using the two clauses.
  - by means of just ONE resolution step in P' using the new fact p(a).

GAINS IN EFFICIENCY!!!: computational steps applied at unfolding time remain *compiled* on tranformed rules FOREVER.

MOTIVATION II: unfolding fuzzy logic programs: [Journals FSS & IJAR, 2023]

- Consider the lattice of truth degrees L = ([0,1], ≤) equipped with connectives &luka, &godel, &prod, |luka, |godel, |prod, @aver,… and the similarity relation R = { r ~ q = 0.4 }

- Original FASILL program P=<π,R,L> s.t. π ={ p(X):-@aver(r(X),0.8). q(a).}

- Unfolded FASILL program P’=<$\pi’$,R,L> s.t. $\pi’$={ p(a) :- @aver(0.4,0.8). q(a).}

- Second unfolding step P”=<$\pi’’$,R,L> s.t. $\pi’’$={ p(a) :- 0.6. q(a).}

- Goal p(a) returns 0.6 in P (tree computational steps), P’ (two steps) and P” (one step).

MOTIVATION III: Symbolic fuzzy logic programms: [Conf. RULEML, 2020]

- Consider the lattice of truth degrees L = ([0,1], ≤) equipped with connectives &prod, |luka, |godel, @aver,… and the SYMBOLIC similarity relation R# = { r ~ q = #s1}
- Original sFASILL program P#=<π#,R#,L> s.t. π# ={ p(X) :- #?s2(r(X),0.8). q(a).}
- Unfolded sFASILL program P’#=<π’#,R#,L> s.t. π’#={ p(a) :- #?s2(#s1 ,0.8). q(a).}
- Goal p(a) returns #?s2(#s1,0.8) in P (two computational steps) and P’ (one step).

WHY USING SYMBOLIC PROGRAMS? → tune fuzzy truth degrees/connectives and similarity relations accordingly to users preferences!!! [Conf. IWANN, 2021: *tuning engine*], [Journal IJAR, 2024, *semantic web*], [Conf. RULEML, 2019: *neural networks*]

USER1: if 0.48 → p(a) then #s1=0.6 and #?s2=&prod

USER2: if 0.9 → p(a) then #s1=0.9 and #?s2=|godel

UNFOLDING I: Formalization of symbolic unfolding

FORMAL DEFINITION: Let P# = ⟨Π#,R#,L⟩ be a sFASILL program and R: (H ← B) ∈ Π# be a rule (with non-empty body B). Then, the symbolic unfolding of rule R in program P# is the new sFASILL program P′# = ⟨Π′#,R#,L⟩, where Π′# = (Π# − {R}) ∪ {Hσ ← B′ | ⟨B; id⟩ ⇝ ⟨B′; σ⟩}.

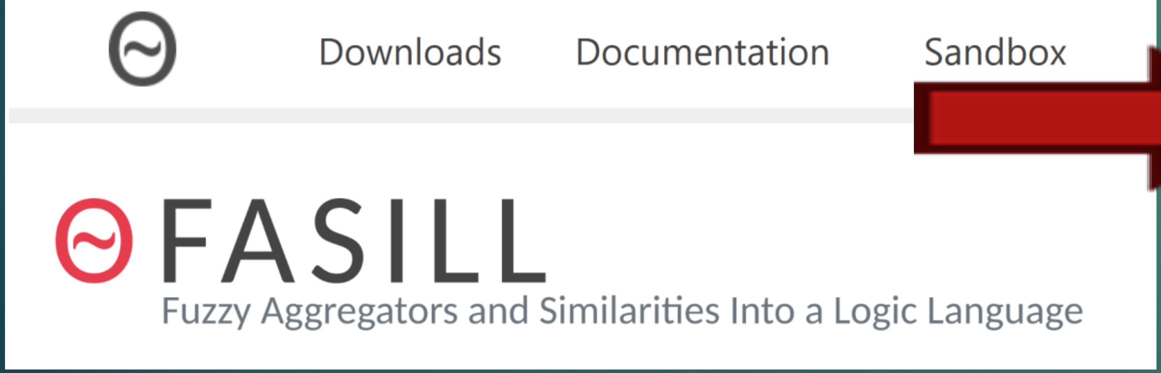
- BASIS of many techniques for OPTIMIZING, specializing, debugging,…
- Transformed programs RUN FASTER but TAKE CARE with:
  - SIZE OF UNFOLDED PROGRAMS: could grow more and more when repeteadly unfolding recursive rules…
  - APPLICABILITY CONDITIONS: some replacements of symbolic constants before/after unfolding a program could produce different answers...

Symbolic Unfolding of Similarity-based Fuzzy Logic Programs

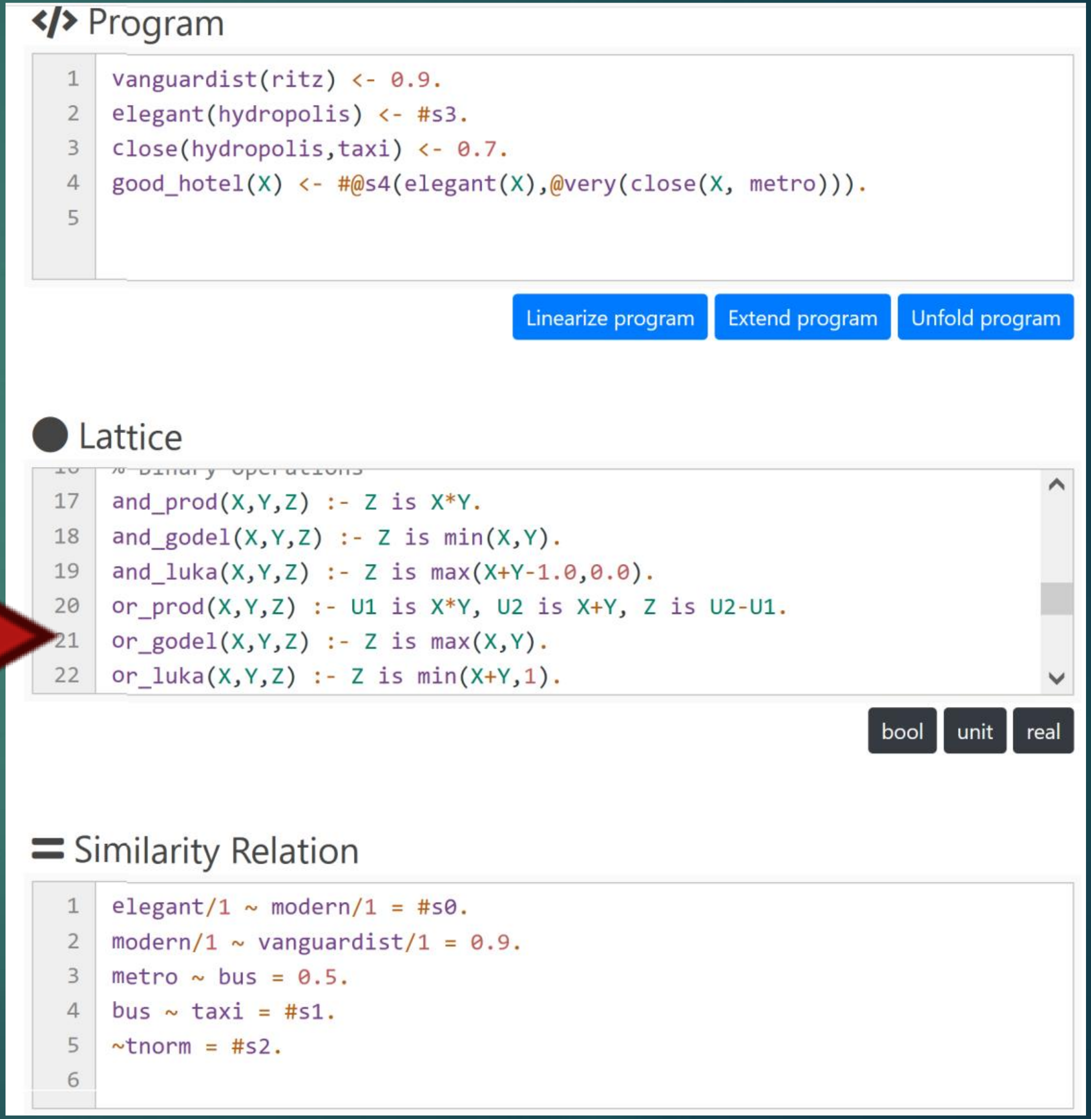
UNFOLDING II:

HTTPS://DECTAU.UCLM.ES/FASILL/SANDBOX

[JOURNAL IJAR, 2020]



Fuzzy Aggregators and Similarities Into a Logic Language



Program

```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(X) <- #@s4(elegant(X),@very(close(X, metro))).
```

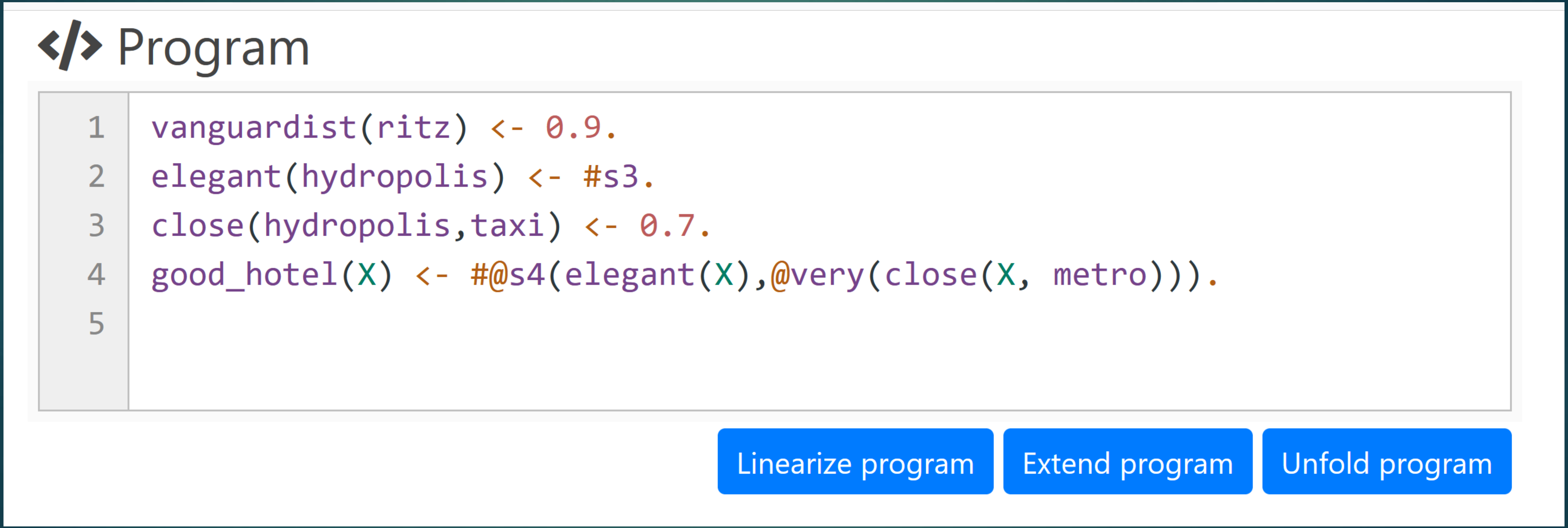
Lattice

```
% Binary operations
and_prod(X,Y,Z) :- Z is X*Y.
and_godel(X,Y,Z) :- Z is min(X,Y).
and_luka(X,Y,Z) :- Z is max(X+Y-1.0,0.0).
or_prod(X,Y,Z) :- U1 is X*Y, U2 is X+Y, Z is U2-U1.
or_godel(X,Y,Z) :- Z is max(X,Y).
or_luka(X,Y,Z) :- Z is min(X+Y,1).
```

Similarity Relation

```
elegant/1 ~ modern/1 = #s0.
modern/1 ~ vanguardist/1 = 0.9.
metro ~ bus = 0.5.
bus ~ taxi = #s1.
~tnorm = #s2.
```

UNFOLDING III: SYMBOLIC PROGRAM RULES

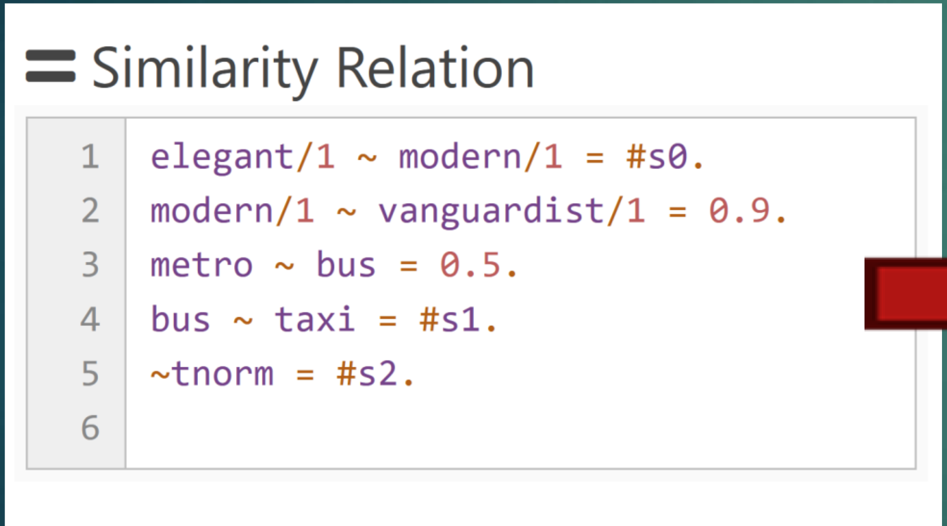


Program

```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(X) <- #@s4(elegant(X),@very(close(X, metro))).

```

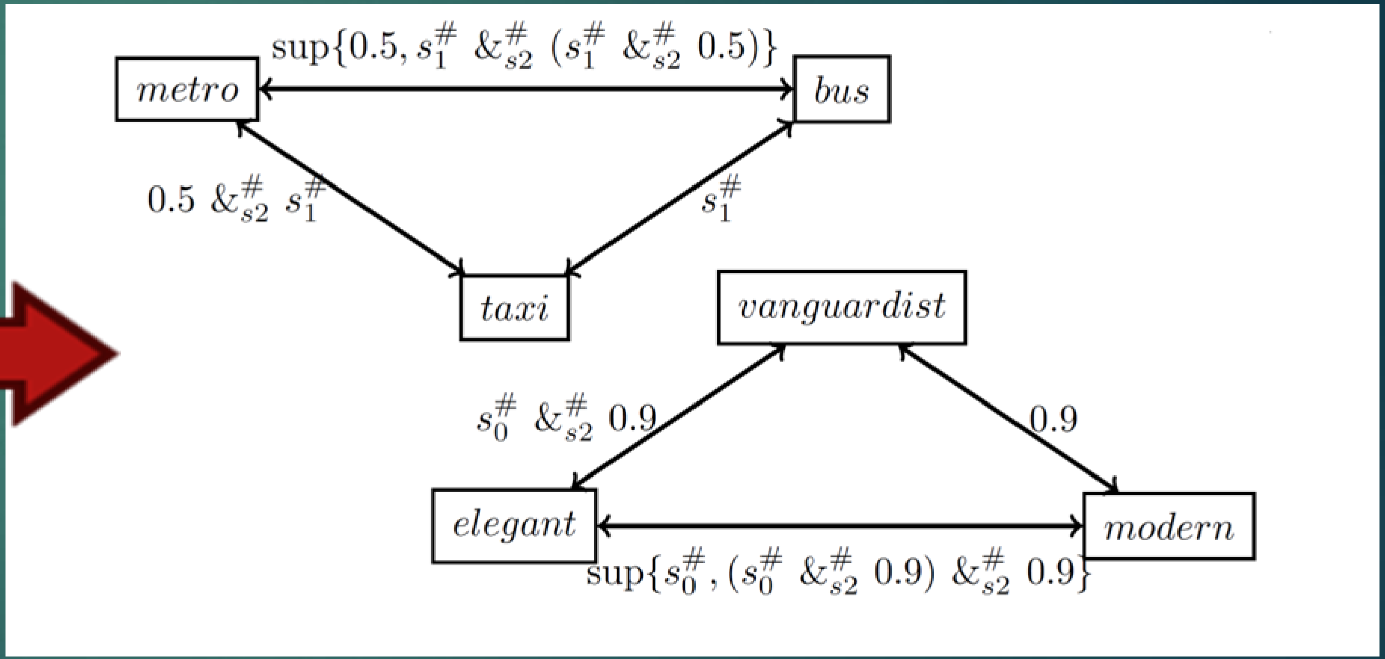
UNFOLDING IV: SYMBOLIC SIMILARITY RELATION [RULEML, 2020]



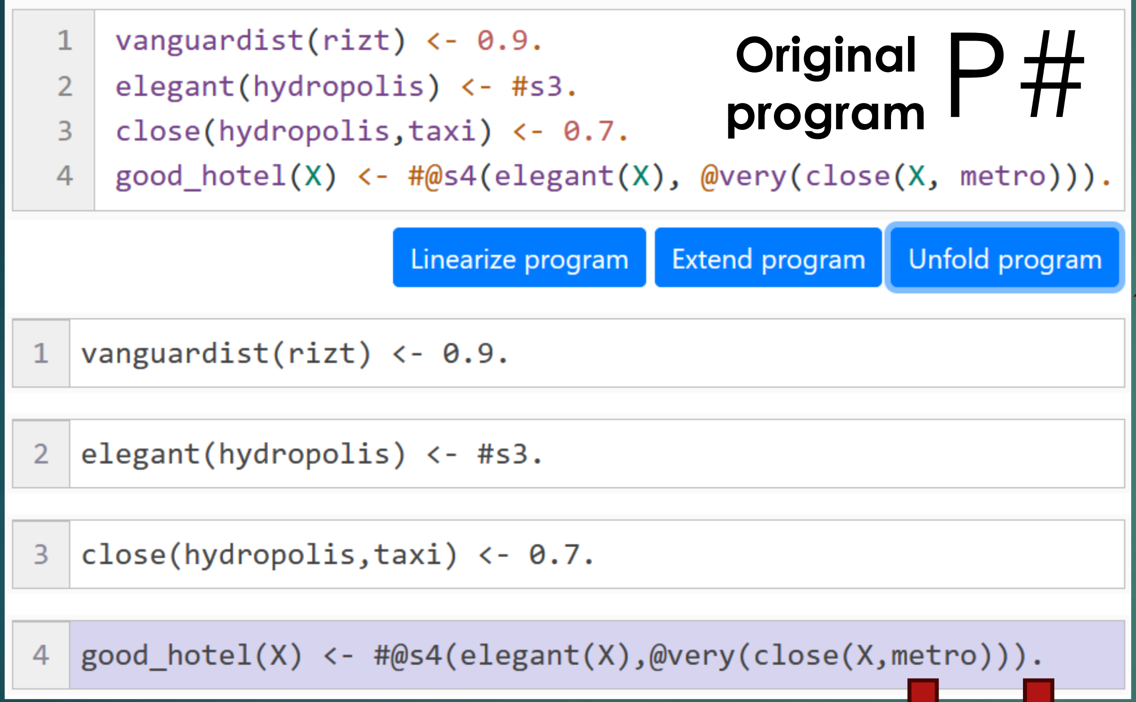
Similarity Relation

```
elegant/1 ~ modern/1 = #s0.
modern/1 ~ vanguardist/1 = 0.9.
metro ~ bus = 0.5.
bus ~ taxi = #s1.
~tnorm = #s2.

```


metro
bus
taxi
$\sup\{0.5, s_1^{\#} \ \&_{s2}^{\#} \ (s_1^{\#} \ \&_{s2}^{\#} \ 0.5)\}$
$0.5 \ \&_{s2}^{\#} \ s_1^{\#}$
$s_1^{\#}$
vanguardist
elegant
modern
$s_0^{\#} \ \&_{s2}^{\#} \ 0.9$
0.9
$\sup\{s_0^{\#}, (s_0^{\#} \ \&_{s2}^{\#} \ 0.9) \ \&_{s2}^{\#} \ 0.9\}$

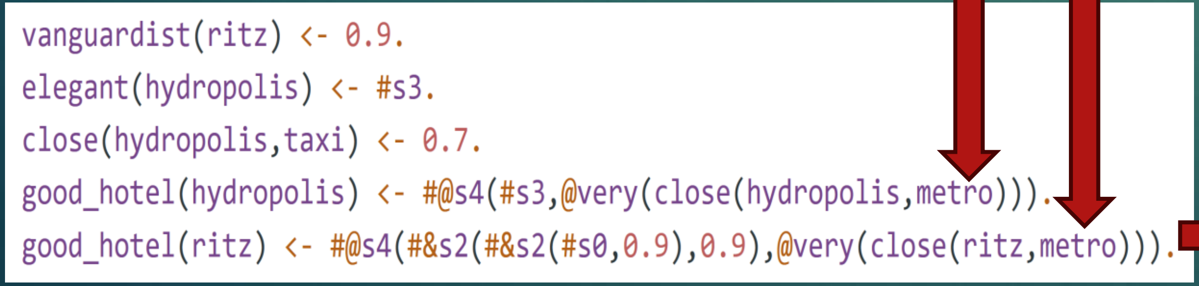
UNFOLDING V: TRANSFORMATION SEQUENCE BASED ON FOUR UNFOLDING STEPS



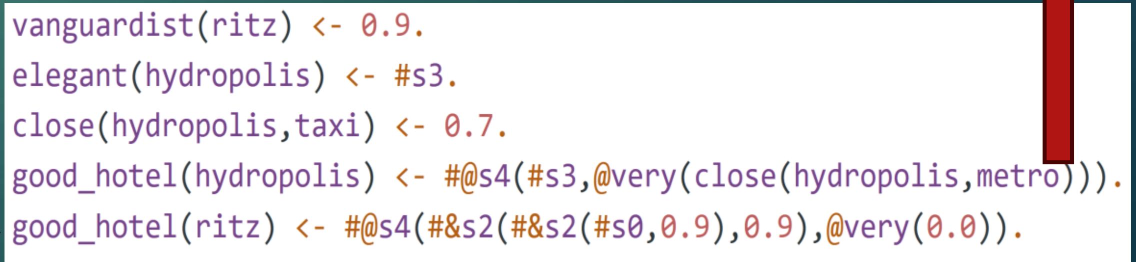
Original program P#

```
vanguardist(rizt) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(X) <- #@s4(elegant(X), @very(close(X, metro))).
```

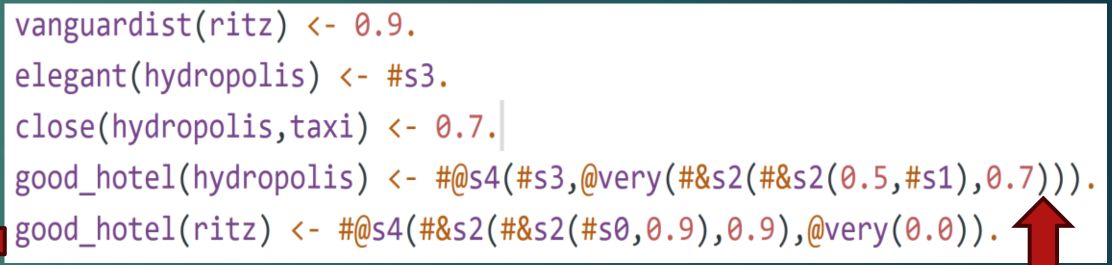
```
vanguardist(rizt) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(X) <- #@s4(elegant(X),@very(close(X,metro))).
```



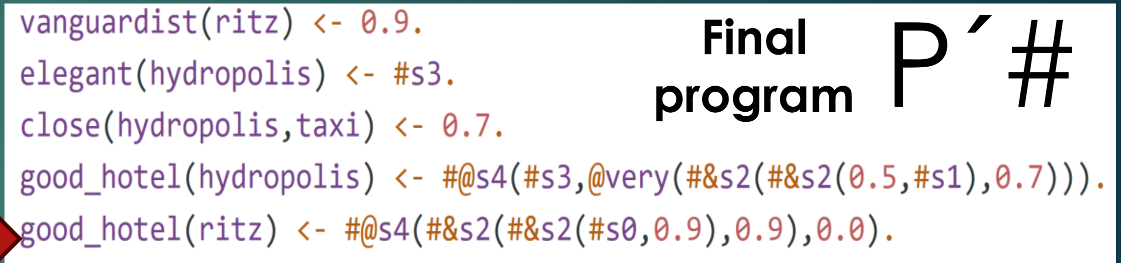
```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(hydropolis) <- #@s4(#s3,@very(close(hydropolis,metro))).
good_hotel(ritz) <- #@s4(#&s2(#&s2(#s0,0.9),0.9),@very(close(ritz,metro))).
```



```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(hydropolis) <- #@s4(#s3,@very(close(hydropolis,metro))).
good_hotel(ritz) <- #@s4(#&s2(#&s2(#s0,0.9),0.9),@very(0.0)).
```



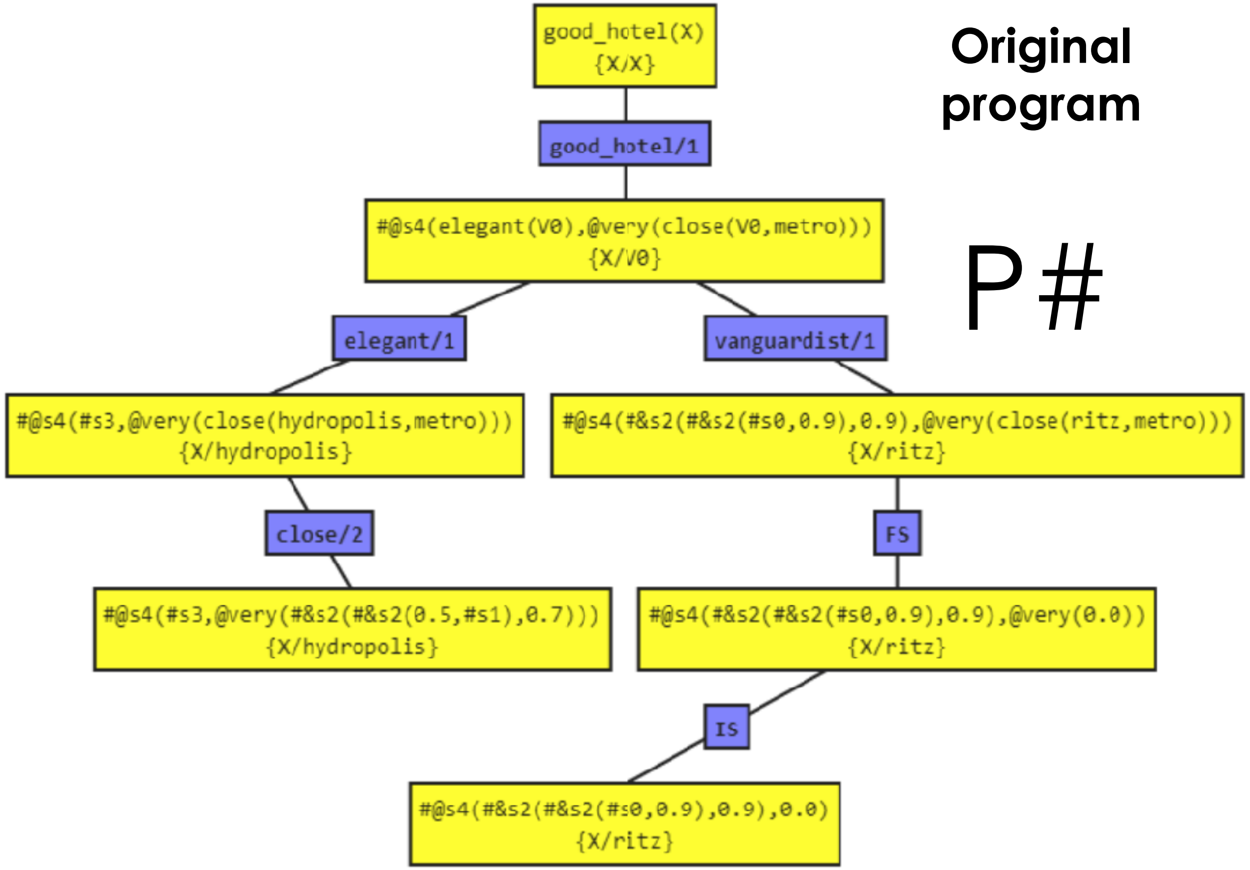
```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(hydropolis) <- #@s4(#s3,@very(#&s2(#&s2(0.5,#s1),0.7))).
good_hotel(ritz) <- #@s4(#&s2(#&s2(#s0,0.9),0.9),@very(0.0)).
```



Final program P´#

```
vanguardist(ritz) <- 0.9.
elegant(hydropolis) <- #s3.
close(hydropolis,taxi) <- 0.7.
good_hotel(hydropolis) <- #@s4(#s3,@very(#&s2(#&s2(0.5,#s1),0.7))).
good_hotel(ritz) <- #@s4(#&s2(#&s2(#s0,0.9),0.9),0.0).
```

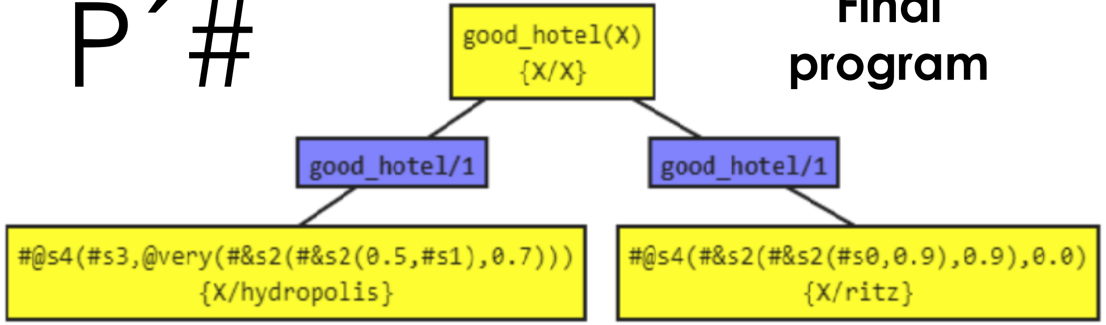
UNFOLDING VI: EXECUTION TREES FOR A GOAL W.R.T. P# AND P'#


good_hotel(X)
{X/X}
good_hotel/1
#@s4(elegant(V0),@very(close(V0,metro)))
{X/V0}
elegant/1
vanguardist/1
#@s4(#s3,@very(close(hydropolis,metro)))
{X/hydropolis}
#@s4(#&s2(#&s2(#s0,0.9),0.9),@very(close(ritz,metro)))
{X/ritz}
close/2
FS
#@s4(#s3,@very(#&s2(#&s2(0.5,#s1),0.7)))
{X/hydropolis}
#@s4(#&s2(#&s2(#s0,0.9),0.9),@very(0.0))
{X/ritz}
IS
#@s4(#&s2(#&s2(#s0,0.9),0.9),0.0)
{X/ritz}

Original program

P#

P´#


good_hotel(X)
{X/X}
good_hotel/1
good_hotel/1
#@s4(#s3,@very(#&s2(#&s2(0.5,#s1),0.7)))
{X/hydropolis}
#@s4(#&s2(#&s2(#s0,0.9),0.9),0.0)
{X/ritz}

Final program

CONCLUSION AND FURTHER RESEARCH

- Unfolding transformation for optimizing fuzzy logic programs:
- [Conf. IWANN, 2019] Symbolic but not similarity-based
- [Journal IJAR, 2023] Similarity-based but not symbolic
- [Conf. IARIA , 2024] BOTH SYMBOLIC AND SIMILARITY-BASED ☺

- Ongoing work: safe applicability conditions and correctness proofs (soundness, completeness, efficiency...)