Labelled Variables in Logic Programming: Foundations

Roberta Calegari[1], Enrico Denti[1], Agostino Dovier[2] Andrea Omicini[1]
{roberta.calegari, enrico.denti, andrea.omicini}@unibo.it
agostino.dovier@uniud.it

[1]Dipartimento di Informatica, Scienza e Ingegneria—Università di Bologna
[2]Dipartimento di Scienze Matematiche, Informatiche e Fisiche—Università di Udine

Talk @ *31th Annual Conference of GULP (CILC 2016)*
Università di Milano-Bicocca, Italy
20 June 2016

Outline

Context

Challenges of today's *pervasive system* [Mariani and Omicini, 2015]

- complex
- distributed
- situated
- intelligent

distributed situated intelligence

enabling models and technologies

Motivations I

Intelligence

Logic Programming

- programs as logic theories
- computation as deduction
- programming with relations and inference

Distribution

- multiple, distributed Prolog engines
  - minimal
  - (dynamically) configurable
  - easily deployable
  - inter-operable
  - multi-platform
  - ...

Motivations II

Situatedness

- reactiveness
  - ! *computational* reactiveness
  - ? *environment* reactiveness
    - domain-specific
      capture diverse computational models tailored to the local needs of situated components / devices
    - → labels
      [Gabbay, 1996] model
      [Holzbaur, 1992] mechanism

Labelled Variables in Logic Programming (LVLP)

WordNet I

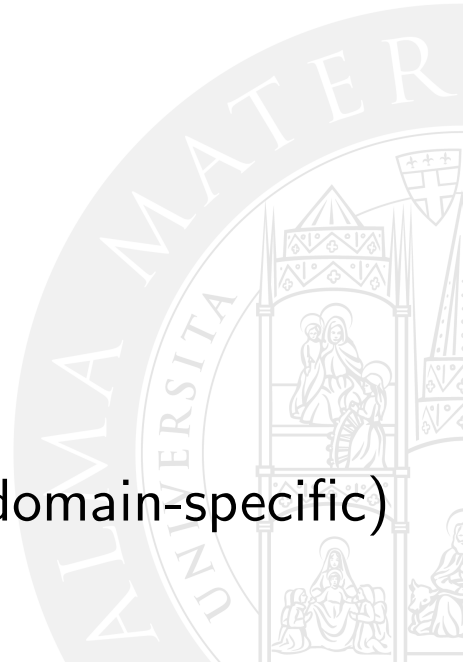
Knowledge-Intensive Environment

A WordNet scenario representing a network of semantically-related words

```
wordnet_fact([‘dog’,‘domestic dog’,‘canis’,‘pet’,‘mammal’,‘vertebrate’]).
wordnet_fact([‘cat’, ‘domestic cat’, ‘pet’, ‘mammal’, ‘vertebrate’]).
wordnet_fact([‘fish’, ‘aquatic vertebrates’, ‘vertebrate’]).
wordnet_fact([‘frog’, ‘toad’, ‘anauran’, ‘batrachian’]).

animal(X) :- X^[‘pet’], X = ‘minnie’.
animal(X) :- X^[‘fish’], X = ‘nemo’.
animal(X) :- X^[‘cat’], X = ‘vmolly’.
animal(X) :- X^[‘dog’], X = ‘frida’.
animal(X) :- X^[‘frog’], X = ‘cra’.
```

- Labels provide a customisable computational model (domain-specific)
- Bound to the standard LP computation



WordNet II

```
?- X^[‘pet’], animal(X).
yes. X / ‘minnie’
[X^[‘dog’, ‘domestic dog’, ‘canis’, ‘pet’, ‘mammal’, ‘vertebrate’]] ;
yes. X / ‘minnie’
[X^[‘cat’, ‘domestic cat’, ‘pet’, ‘mammal’, ‘vertebrate’]] ;
yes. X / ‘molly’
[X^[‘cat’, ‘domestic cat’, ‘pet’, ‘mammal’, ‘vertebrate’]] ;
yes. X / ‘frida’
[X^[‘dog’, ‘domestic dog’, ‘canis’, ‘pet’, ‘mammal’, ‘vertebrate’]]

?- X^[‘vertebrate’], animal(X).
yes. X = ‘minnie’
X^[‘dog’, ‘domestic dog’, ‘canis’, ‘pet’, ‘mammal’, ‘vertebrate’] ;
yes. X = ‘minnie’
X^[‘cat’, ‘domestic cat’, ‘pet’, ‘mammal’, ‘vertebrate’] ;
yes. X = ‘molly’
X^[‘cat’, ‘domestic cat’, ‘pet’, ‘mammal’, ‘vertebrate’] ;
yes. X = ‘frida’
X^[‘dog’, ‘domestic dog’, ‘canis’, ‘pet’, ‘mammal’, ‘vertebrate’] ;
yes. X = ‘nemo’
X^[‘fish’,‘aquatic vertebrates’, ‘vertebrate’]
```

Hybrid Computations I

Recipe Scenario

The knowledge base contains a collection of ingredients

```
meat(M) :- M^[100g, 145cal, [curedMeats]],       M=['Parma ham'].
meat(M) :- M^[100g, 210cal, [curedMeats]],       M=['ham'].
meat(M) :- M^[200g, 220cal, [whiteMeats]],       M=['chicken'].
vegetable(V) :- V^[100g,  35cal, [vegetables]], V=['carrots'].
vegetable(V) :- V^[ 50g,  10cal, [vegetables]], V=['salad'].
bread(B) :- B^[100g, 265cal, [whiteBread]],    B=['bread with poolish'].
bread(B) :- B^[100g, 240cal, [integralBread]], B=['whole grain bread'].

recipe(R) :- R^◇, meat(Meat), vegetable(Vegetable), R=[Meat,Vegetable].
recipe(R) :- R^◇, Meat^[any, any, [curedMeats]], bread(Bread), meat(Meat),
             vegetable(Vegetable), R=[Bread, Meat, Vegetable].
```

- Labels are not subject to the single-assignment assumption
- Diverse computational models next to LP: affecting set of solutions
- Diverse matching criteria: domain/user dependent

Hybrid Computations II

```
?- R^[any, 200cal, any], recipe(R).
yes. R / [‘tofu meatballs’, ‘carrots’]
R^[200g, 191cal, [alternativeMeats, vegetables]] ;
yes. R / [‘tofu meatballs’, ‘salad’]
R^[200g, 166cal, [alternativeMeats, vegetables]] ;
yes. R / [‘Parma ham’, ‘carrots’]
R^[200g, 180cal, [curedMeats, vegetables]] ;
yes. R / [‘ham’, ‘salad’]
R^[150g, 220cal, [curedMeats, vegetables]]
```

```
?- R^[any, 180cal,[alternativeMeats]], recipe(R).
yes. R / [‘tofu meatballs’,‘carrots’]
R^[200g, 191cal, [alternativeMeats, vegetables]] ;
yes. R / [‘tofu meatballs’,‘salad’]
R^[150g, 166cal, [alternativeMeats, vegetables]]
```

Standard LP extension Comparison I

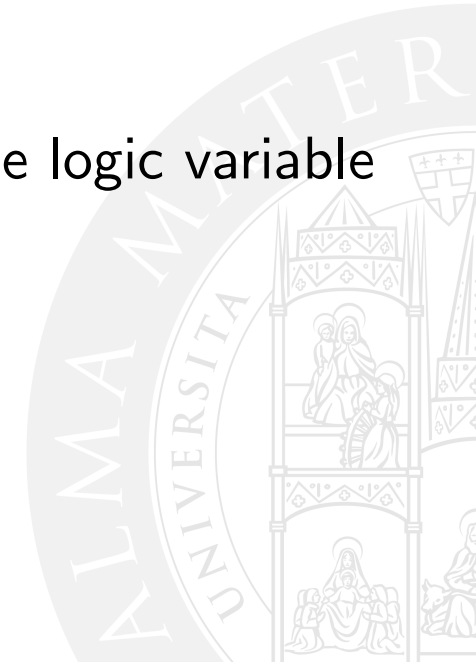
Constraint Logic Programming

- The computational model is supported
- Computational efficiency is not comparable

```
interval(X):- X^[-1,4].
interval(X):- X^[6,10].
```

Labels used to represent the integer interval over which the logic variable values span: take the form `X^[`min`,`max`]`

```
?- X^[2,7], interval(X).
yes.
X^[2,4] ;
yes.
X^[6,7]
```



Standard LP extension Comparison II

Integers with a Neighbourhood

The expressiveness of LVLP

- easily move to more complex domains
- integers with a neighbourhood

```
neighbourhood(X):- X^[-1,4], X=3.
neighbourhood(X):- X^[6,10], X=8.

?- X^[2,7], neighbourhood(X).
yes.  X / 3
X^[2,4]
```

Outline

Labelled Variables Model in Logic Programming: LVLP

- A set $\mathscr{B} = \{\beta_1, \ldots, \beta_n\}$ of *basic labels* that are the basic entities of the *domain of labels*
- A set $\mathscr{L} \subseteq \wp(\mathscr{B})$ of *labels*, where each $\ell \in \mathscr{L}$ is a subset of $\mathscr{B}$
- A set of *variables* $\mathscr{V}$, where $v \in \mathscr{V}$ is the generic variable
- A *labelling*: a set of pairs $\langle v_i, \ell_i \rangle$, associating the label ℓ_i to the variable v_i
- A (label-)combining function f_L, synthesising a new label from two

$$f_L : \mathscr{L} \times \mathscr{L} \longrightarrow \mathscr{L}$$

- A compatibility function f_C to check the compatibility between a term $t \in \mathscr{H}$ and a label $\ell \in \mathscr{L}$

$$f_C : \mathscr{H} \times \mathscr{L} \longrightarrow \{\text{true}, \text{false}\}$$

LVLP Unification & Rules

Program Rule

$$Head \leftarrow Labellings, Body.$$

- *Head* is an atomic formula
- *Labellings* is the list of Variable/Label associations in the clause
- *Body* is a list of atomic formulas

Unification of two labelled variables

$$(\text{true/false} \wedge f_C(\Theta, A), \Theta, A)$$

- *true/false* represents if there is an answer
- $f_C(\Theta, A)$ compatibility between variable values and their labels
- Θ represents the most general substitution for all variables
- A represents the corresponding label-variable associations

Unification Rules

	constant c_2	variable v_2	labelled variable $\langle v_2, \ell_2 \rangle$	compound term s_2
constant c_1	if $c_1 = c_2$ then true else false	true, $\{v_2/c_1\}$	if $f_C(c_1, \ell_2) =$ true then true, $\{v_1/c_2\}$, ℓ_1 else false	false
variable v_1	true, $\{v_1/c_2\}$	true, $\{v_1/v_2\}$	true, $\{v_1/v_2\}, \ell_2$	if v_1 does not occur in s_2 then true, $\{v_1/s_2\}$ else false
labelled variable $\langle v_1, \ell_1 \rangle$	if $f_C(c_2, \ell_1) =$ true then true, $\{v_1/c_2\}$, ℓ_1 else false	true, $\{v_1/v_2\}, \ell_1$	if $f_L(\ell_1, \ell_2) \neq \emptyset$ then true, $\{v_1/v_2\}$, $f_L(\ell_1, \ell_2)$ else false	if v_1 does not occur in s_2 and $f_L(\ell_1, f_L(\ell_2, \ldots, f_L(\ell_{n-1}, \ell_n) \ldots)) \neq \emptyset$ where $\ell_2, \ldots \ell_n$ are labels of variables in s_2 then true, $\{v_1/s_2\}$, $f_L(\ell_1, f_L(\ell_2, \ldots, f_L(\ell_{n-1}, \ell_n) \ldots))$ else false
compound term s_1	false	if v_2 does not occur in s_1 then true, $\{v_2/s_1\}$ else false	if v_2 does not occur in s_1 and $f_L(\ell_1, f_L(\ell_2, \ldots, f_L(\ell_{n-1}, \ell_n) \ldots)) \neq \emptyset$ where $\ell_2, \ldots \ell_n$ are labels of variables in s_1 then true, $\{v_2/s_1\}$, $f_L(\ell_1, f_L(\ell_2, \ldots, f_L(\ell_{n-1}, \ell_n) \ldots))$ else false	if s_1 and s_2 have same functor and arity and arguments (recursively) unify then true else false

Outline

Fixpoint Semantics

Denotational Semantics based on T_P

$$T_P(I) = \left\{ \begin{array}{l} \langle\, p(\widetilde{t}), \widetilde{\ell}\,\rangle : \\ \quad r = p(\widetilde{x}) \leftarrow \Lambda_{\widetilde{x}} \mid b_1, \ldots, b_n \qquad (1) \\ \quad \text{where r is a fresh renaming of a rule of } P, \\ \quad v \text{ is a valuation on } \mathscr{H} \text{ such that } v(\widetilde{x}) = \widetilde{t}, \qquad (2) \\ \quad \bigwedge_{i=1}^{n} \exists\, \widetilde{\ell}_i \text{ s.t. } \langle v(b_i), \widetilde{\ell}_i\rangle \in I, \qquad (3) \\ \quad \mathcal{X} \models \Lambda_{\widetilde{t}} \wedge \bigwedge_{i=1}^{n} f_C\big(v(b_i), \widetilde{\ell}_i\big), \qquad (4) \\ \quad \widetilde{\ell} = \widetilde{f_L}\big(r, \Lambda_{\widetilde{t}}, \widetilde{\ell}_1, \cdots, \widetilde{\ell}_n\big) \qquad (5) \end{array} \right\}$$

where $\Lambda_{\widetilde{t}} = v(\Lambda_{\widetilde{x}}) = [\langle t_1, \ell_1\rangle, \ldots, \langle t_n, \ell_m\rangle]$ if $\Lambda_{\widetilde{x}} = [\langle x_1, \ell_1\rangle, \ldots, \langle x_m, \ell_m\rangle]$

Soundness and Completeness of T_P

We demonstrate that

- T_P is a monotonic and continuous mapping on partial orders

$\rightarrow$ f_L is required to be ACI

Operational Semantics

Labelled-Machine State $\sigma = \langle t_0\ t_1 ... t_n, W, \Lambda \rangle$

- $t_0\ t_1 ... t_n$ is the list of terms (goals)
- W is the current list of variables bindings
- Λ is the set of the current labelling associations on W

Inference Step by Program Rule $s_0 \leftarrow \Lambda', s_1, s_2, \ldots, s_m$

- conditions
  - $\exists\, mgu\ \theta$ such that $\theta(t_0) = \theta(s_0)$
  - $\widetilde{f}_{\theta L}(\Lambda, \theta(\Lambda')) \neq \emptyset$
- final state
  - $\sigma' = \langle \theta(s_1, \ldots, s_m, t_1, \ldots, t_n), W' = W \circ \theta, \Lambda'' = \widetilde{f}_{\theta L}(\Lambda, \theta(\Lambda')) \rangle$

Labelled-Machine Final State $\sigma_f = (\epsilon, W_f, \Lambda_f)$

- ϵ is the empty sequence, W_f is the final list of variables and bindings, Λ_f is the corresponding labelling associations set
- A sequence of applications of inference steps is said a *derivation*
- A derivation is *successful* if it ends in a final state, *failing* if it ends in a non-final state where no inference step is possible

Semantics Equivalence

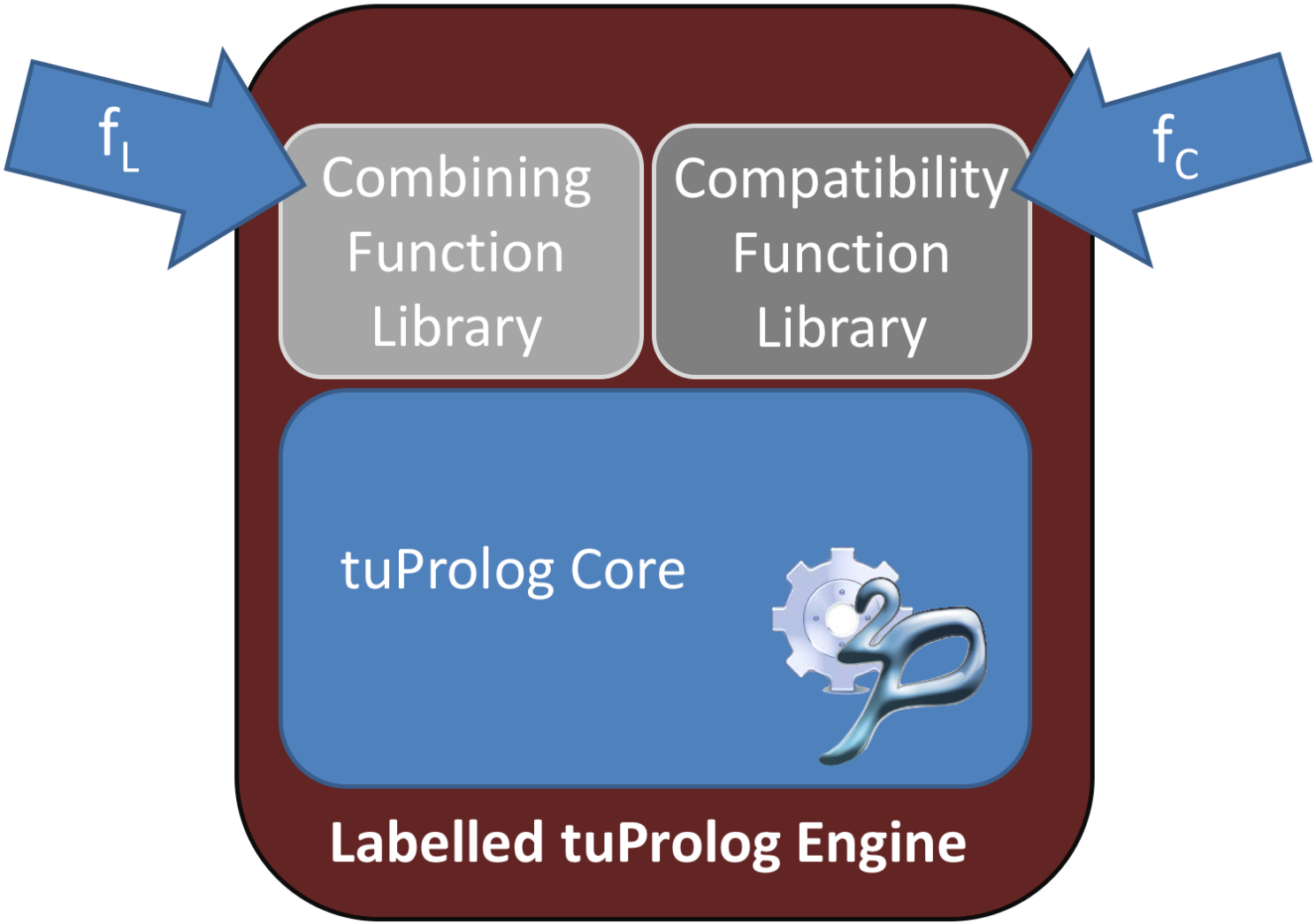
Proposition

Let $p(\widetilde{x})$ be an atom, v a valuation on $\mathscr{H}$ such that $v(\widetilde{x}) = \widetilde{t}$ where $\widetilde{t}$ are ground terms, and $\widetilde{\ell}$ a list of labels

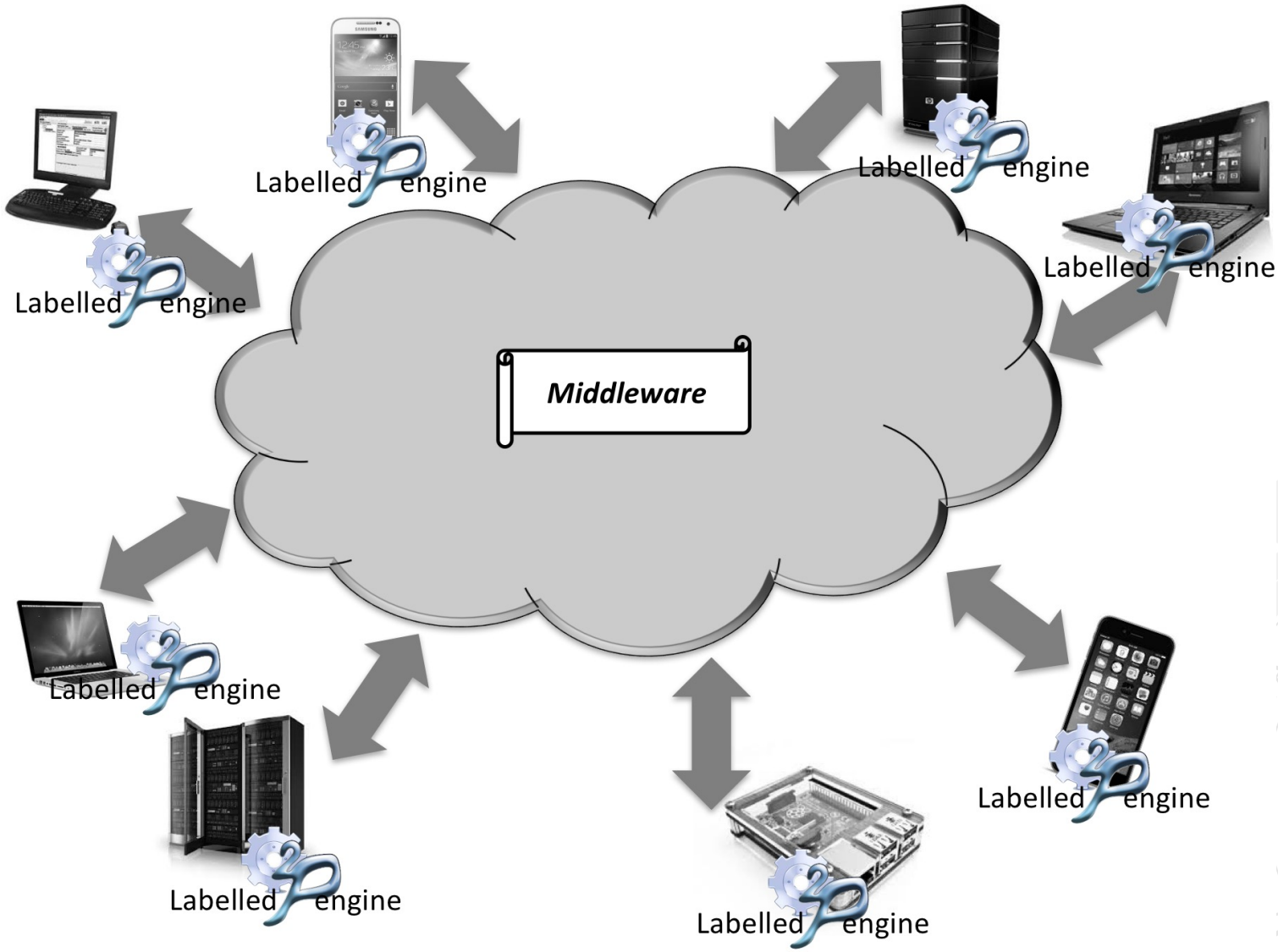
Then there is a successful derivation for $\langle p(\widetilde{t}), v, \langle v(\widetilde{x}), \widetilde{\ell}\rangle\rangle$ iff $\langle p(\widetilde{t}), \widetilde{\ell}\rangle \in T_P \uparrow \omega$

Outline

Labelled tuProlog Engine



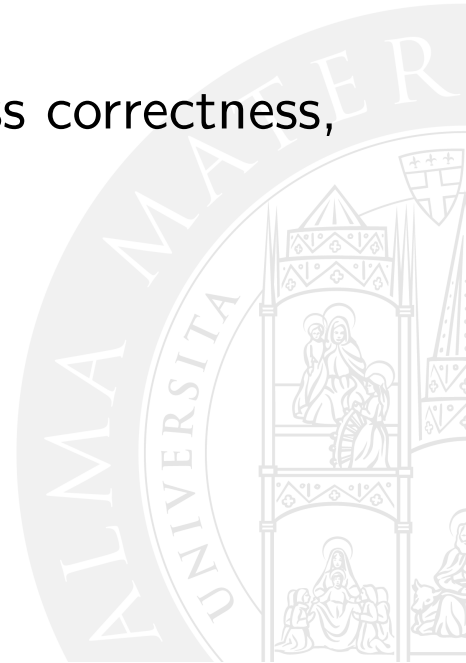
tuProlog Middleware Architecture



Outline

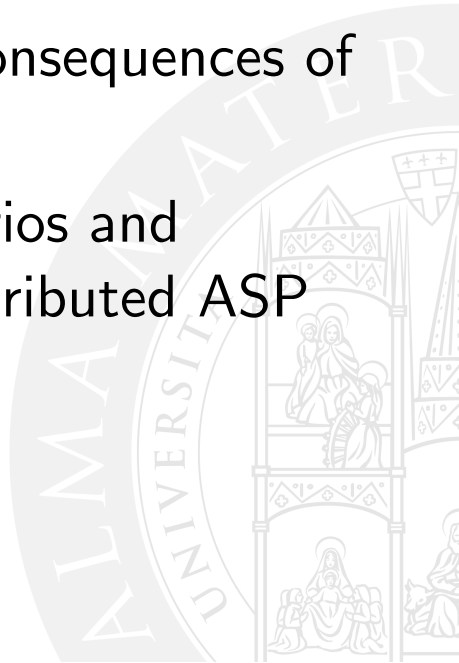
Conclusions

- Definition of the LVLP theoretical framework
- Domain-specific computational models can be expressed via *labelled variables*
- Present the fixpoint and operational semantics, discuss correctness, completeness, and equivalence



Future Work

- Further development of the Labelled tuProlog [Denti et al., 2005] prototype
- Design and implementation of a full-fledged *middleware* for tuProlog and LVLP
- Deeper exploration and better understanding of the consequences of applying labels to formulas
- Application of the LVLP framework to different scenarios and approaches (probabilistic [Skarlatidis et al., 2015], distributed ASP reasoning [Dovier and Pontelli, 2009],...)



Thanks

Thank you for your attention

References I

Denti, E., Omicini, A., and Ricci, A. (2005).
Multi-paradigm Java-Prolog integration in tuProlog.
Science of Computer Programming, 57(2):217–250.

Dovier, A. and Pontelli, E. (2009).
Present and future challenges for ASP systems.
In Erdem, E., Lin, F., and Schaub, T., editors, *Logic Programming and Nonmonotonic Reasoning. 10th International Conference, LPNMR 2009, Potsdam, Germany, September 14-18, 2009. Proceedings*, pages 622–624. Springer.

Gabbay, D. M. (1996).
Labelled Deductive Systems, Volume 1.
Clarendon Press, Oxford Logic Guides 33.

Holzbaur, C. (1992).
Metastructures vs. attributed variables in the context of extensible unification.
In Bruynooghe, M. and Wirsing, M., editors, *Programming Language Implementation and Logic Programming*, volume 631 of *Lecture Notes in Computer Science*, pages 260–268. Springer.
4th International Symposium (PLILP'92) Leuven, Belgium, August 26–28, 1992 Proceedings.

References II


Mariani, S. and Omicini, A. (2015).
Coordinating activities and change: An event-driven architecture for situated MAS.
Engineering Applications of Artificial Intelligence, 41:298–309.


Skarlatidis, A., Artikis, A., Filippou, J., and Paliouras, G. (2015).
A probabilistic logic programming event calculus.
Theory and Practice of Logic Programming, 15(Special Issue 02 (Probability, Logic and Learning)):213–245.

URLs

Slides

- On APICe

→ `http://apice.unibo.it/xwiki/bin/view/Talks/Label2pCilc016`

- On SlideShare

→ `http://www.slideshare.net/rcalegari1/labelled-variables-in-logic-programming-foundations`

Paper

- On APICe

→ `http://apice.unibo.it/xwiki/bin/view/Publications/Label2pCilc016`

- On CEUR

→ `http://ceur-ws.org/Vol-1645/paper_7.pdf`

Labelled Variables in Logic Programming: Foundations

Roberta Calegari[1], Enrico Denti[1], Agostino Dovier[2] Andrea Omicini[1]
{roberta.calegari, enrico.denti, andrea.omicini}@unibo.it
agostino.dovier@uniud.it

[1]Dipartimento di Informatica, Scienza e Ingegneria—Università di Bologna
[2]Dipartimento di Scienze Matematiche, Informatiche e Fisiche—Università di Udine

Talk @ *31th Annual Conference of GULP (CILC 2016)*
Università di Milano-Bicocca, Italy
20 June 2016