Labelled Variables in Logic Programming:

A First Prototype in tuProlog

Roberta Calegari, Enrico Denti, Andrea Omicini

{roberta.calegari, enrico.denti, andrea.omicini}@unibo.it

DISI

Alma Mater Studiorum—Università di Bologna

Talk @ *14th Conference of the Italian Association for Artificial Intelligence*

University of Ferrara - 23rd September 2015

Outline

1. Scope & Goals
2. The Labelled Variables Model in Logic Programming
   - Unification rules
   - Examples
3. Prototype in tuProlog
   - System architecture
   - The language extension
4. Case studies
   - Interval CLP
   - Dress Selection
5. Conclusions & Further Work

Context and Motivation

Explore models and technologies to face the challenges of *pervasive system*

- complex
- distributed
- situated
- intelligent

From *distributed intelligence* towards *distributed situated intelligence*

↓

situated (intelligent) component for pervasive (intelligent) systems
Logic Programming (LP) extensions for *interacting* with the *environment*

Prodromes of situated LP

- Forms of control which generalises data-driven computation, e.g. CLP [Jaffar and Lassez, 1987]
- Ability to be domain-specific – e.g. CHR [Fruhwirth, 1998]

However they are designed against a landscape different from nowadays pervasive systems.

Archeology

[Jaffar and Lassez, 1987] defines CLP as a merger of two declarative paradigms: constraint solving and logic programming $\rightarrow$ capability for reasoning in the *specific finite domain* of application

[Fruhwirth, 1998] introduces CHR as a high-level and declarative language for implementing constraint solver. *User-defined constraints* led to widely domain applications (constraint solving, type checking, natural language processing, multi-agent systems,...)

[Neumerkel, 1990] presentes metastructures to extend syntactic unification: data types whose (unification) semantics are specified by *user-defined predicates*

[Holzbaur, 1992] introduces attributed variables as mechanism to *support of CLP languages*, yet inside the LP context

[Gabbay, 1996] introduces Labelled Deductive System: general notion of label and a *new unifying methodology* for logic

Goals

Aim

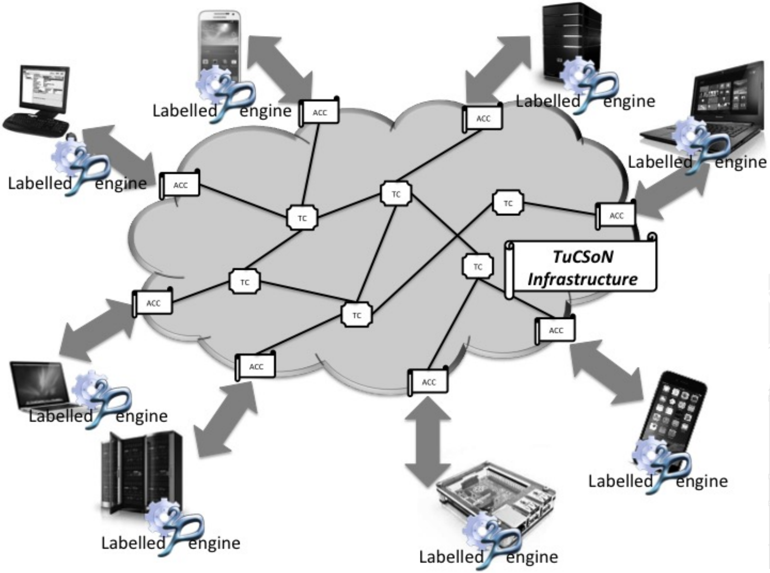
- To devise out simple and expressive *mechanisms* enabling *data-driven and domain specific within a LP framework*
- To acquire information from any source of data (constraint store, data base, sensors,...) $\rightarrow$ infer knowledge

Method

Investigating LP extension based on *labelled variables*

1. To define a *labelled variables logic programming* model where labels are exploited to define computations in *domain-specific contexts*
2. To exploit labelled variables for enabling computations at a separate level, while retaining the general coherence of the LP approach
3. To present a first prototype of Labelled tuProlog

Conceptual Architecture



Outline

The Labelled Variables Model in Logic Programming

Defined by:

- a set of *basic labels* $\mathscr{B}$, where $b \in \mathscr{B}$ is the generic *basic label* here taking the form of a logic term, i.e. $\mathscr{B} \subseteq \mathscr{T}$;
- a set of *labels* $\mathscr{L}$, where $l \in \mathscr{L}$ is a generic *label* defined as a *set of basic labels*, i.e. $l = \{b_1, .., b_n\}$;
- a *labelling association* denoted as $\langle v, l \rangle$ that associates the label l to the variable v.

Unification of two labelled variables

Represented by tuple **(true/false, θ, label)**:

- *true/false* represents if there is an answer
- θ represents the most general substitution
- *label* represents the new label associated to the unified variables

Outline

Unification Rules

Combining function

f_L to synthesise a new label from two given ones. i.e. combine two different labels:

$$f_L : \mathscr{L} \text{x} \mathscr{L} \rightarrow \mathscr{L}$$

The function embeds the *scenario-specific criterion* for defining a new label $\rightarrow$ *domain specific* computation

T_2 \ T_1	constant C_2	variable X_2	labelled variable X_2^{l2}	compund term S_2
constant C_1	true if $C_1 = C_2$	true - $\{X_2/C_1\}$	false	false
variable X_1	true - $\{X_1/C_2\}$	true - $\{X_1/X_2\}$	true - $\{X_1/X_2\} - l_2$	true - $\{X_1/S_2\}$
labelled variable X_1^{l1}	false	true - $\{X_1/X_2\}$ - l_1	true if not $\{\,\}$ - $\{X_1/X_2\} - f_L(l_1, l_2)$	false
compund term S_1	false	true - $\{X_2/S_1\}$	false	true if S_1 and S_2 unify

Outline

Examples I

Arithmetic Inequalities 1/2

- Basic labels are arithmetic intervals: $b \in \mathscr{B}$ takes one of the following forms (where a_3, a_5 can also be $-\infty$, while a_4, a_8 can also be $+\infty$)

$$b = [a_1, a_2] = \{x \in \mathbb{R} \mid a_1 \leq x \leq a_2, a_1, a_2 \in \mathbb{R}\}$$
$$b =]a_3, a_4[= \{x \in \mathbb{R} \mid a_3 < x < a_4, a_3, a_4 \in \mathbb{R}\}$$
$$b =]a_5, a_6] = \{x \in \mathbb{R} \mid a_5 < x \leq a_6, a_5, a_6 \in \mathbb{R}\}$$
$$b = [a_7, a_8[= \{x \in \mathbb{R} \mid a_7 \leq x < a_8, a_7, a_8 \in \mathbb{R}\}$$

- The *combining function* f_L intersects intervals
  Given $l_1 = \{b_{11}, ..b_{1n}\}$ and $l_2 = \{b_{21}, ..b_{2m}\}$, the resulting label defined as the union of all the possible intersections:

$$f_L : (l_1, l_2) = f_L(\{b_{11}, ..b_{1n}\}, \{b_{21}, ..b_{2m}\}) =$$
$$\{\{(b_{11} \cap b_{21}) \cup .. \cup (b_{11} \cap b_{2m})\}, ..., \{(b_{1n} \cap b_{21}) \cup .. \cup (b_{1n} \cap b_{2m})\}\}$$

Examples II

Arithmetic Inequalities 2/2

For instance, let X and Y two variables involved in two inequalities, with the following constraints:

$$X^{\{[3..5],[9..10]\}} \qquad Y^{\{[8..10]\}}$$

If these labels are eventually combined, the new label l_3 obtained by applying f_L results:

$$l_1 = \{[3..5],[9..10]\}$$
$$l_2 = \{[8..10]\}$$
$$l_3 = f_L(l_1, l_2) = f_L(\{[3..5],[9..10]\},[8..10]) =$$
$$= \{([3..5] \cap [8..10]) \cup ([9..10] \cap [8..10])\} = [9..10]$$

Examples III

RGB colour space 1/2

- Basic labels assume the syntax of RGB triplet, so that

$$b = (r, g, b) = \{r, g, b \in \mathbb{N} \mid 0 \leq r, g, b \leq 255\}$$

- Label $l \in \mathscr{L}$ becomes the singleton

$$l = (r, g, b)$$

- The *combining function* f_L is supposed to blend colours according to the specific application needs: computing the arithmetic mean of the colour component labels

$$f_L(l_1, l_2) = f_L((r_1, g_1, b_1), (r_2, g_2, b_2)) = \left(\frac{r_1 + r_2}{2}, \frac{g_1 + g_2}{2}, \frac{b_1 + b_2}{2}\right) = l_3$$

Examples IV

RGB colour space 2/2

Let X represent a red object, Y a yellow one and Z a blue one:

$$X^{(255,0,0)} \quad Y^{(255,255,0)} \quad Z^{(0,0,255)}$$

Combining X with Y according to the combine criterion embedded in f_L leads to:

$$l_1 = (255, 0, 0) \qquad l_2 = (255, 255, 0)$$

$$l_3 = f_L(l_1, l_2) = ((255, 0, 0), (255, 255, 0)) = (255, 128, 0) = l_3$$

Combining X with Z, instead, would lead to a violet label

$$l_4 = (128, 0, 128)$$

Outline

Outline

System architecture

tuProlog Architecture

- minimal Prolog virtual machine
- available as a self-contained object
- featuring a simple interface
- dynamically linkable and dischargeable libraries of predicates

Labelled tuProlog extension

- libraries of Prolog predicates
- suitable Java methods

⇒

Extend the built-in unification by user-definition semantics

↓

enabling data-driven & domain specific computation

Outline

The language extension I

The Prolog language level

Variable/Label association — `label_associate(+Var, +Label)`
to associate variable *Var* to label *Label* — as °/2 operator

Function f_L — `label_generate(+Label1, +Label2, -Label3)`
to specify how to build a new label from two given ones

Label-interpreted terms — `label_interpret(+Label, +Term)`
to enable terms to influence label computation, i.e. to be interpreted in the labels world
success if *Term* can potentially be unified with *Label* in the label world, fail otherwise – only if this check succeeds, the labelled variable is actually unified with the term.

The language extension II

The Java language level

Variable/Label association — `Label`
is a just a new property of the existing `Variable` class

Function f_L — `Label labelGenerate(Label l1, Label l2)`
Java method to embed f_L behaviour

Label-interpreted terms — `boolean labelInterpret(Label l, Term t)`
to enable terms to be interpreted in labels world

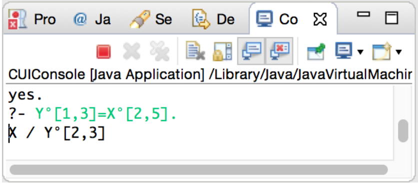
The abstract class `LabelUnifier` is provided for the user's convenience: to define a new Labelled Variable Model, the user can just extend it.

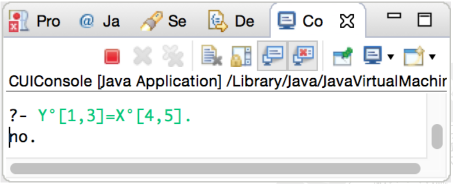
Outline

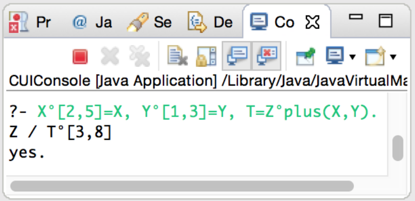
Outline

Interval CLP

- Variables are labelled with their admissible numeric interval—that is, `X°[A,B]` means that `X` can span over the range $[A..B]$
- Unification succeeds if two numeric intervals overlap, in which case the intersected interval is the newly-computed label.







Outline

Dress Selection Colour Constraints I

- The goal is to select from the wardrobe all the shirts that respect some given colour constraints: domain of labels → shirts and colours
- `shirt(`*Colour*`,`*Description*`)` represents a shirt with of colour *Colour* (in RGB `rgb(`*Red*`,`*Green*`,`*Blue*`)`), described by *Description*
- Wardrobe content representation:
  - shirt($A^{rgb(255,240,245)}$, pink_blouse).
  - shirt($B^{rgb(255,222,173)}$, yellow_tshirt).
  - shirt($C^{rgb(119,136,153)}$, army_tshirt).
  - shirt($D^{rgb(188,143,143)}$, periwinkle_blouse).
  - shirt($E^{rgb(255,245,238)}$, cream_blouse).
- Two RGB colours are considered similar if their distance is below a given threshold

Dress Selection Colour Constraints II

- f_L function, checking if two given labels are to be considered similar:

$$f_L(c_1, c_2) = \begin{cases} c_2 & \text{if } \mathit{euclidean\_distance}(c_1, c_2) \leq \mathit{threshold} \in [0..100] \\ \{\} & \text{if } \mathit{euclidean\_distance}(c_1, c_2) > \mathit{threshold} \in [0..100] \end{cases}$$

No constraints on target colour

```
CUIConsole [Java Application] /Library/Java/JavaVirtu
?- shirt(Colour, Desc).
Desc / pink_blouse
 ? ;
Desc / yellow_tshirt
 ? ;
Desc / army_tshirt
 ? ;
Desc / periwinkle_blouse
 ? ;
Desc / cream_blouse
 ? ;
no.
```

Target colour papaya

```
CUIConsole [Java Application] /Library/Java/JavaVirtualMa
?- shirt(Colour°rgb(255,239,213), Desc).
Desc / pink_blouse
 ? ;
Desc / yellow_tshirt
 ? ;
Desc / cream_blouse
 ? ;
no.
```

Outline

Conclusions

- We presented the new Labelled tuProlog
- Our first step for making logic-based technologies fit the requirements of nowadays pervasive system

Labelled tuProlog

Making it possible to spread intelligence via a light-weight Prolog engine supporting domain-specific situated computations via labelled variables

Further Work

Several extensions and further comparisons, in different directions, are worth considering, we intend to:

- formalise, with Labelled tuProlog, some well-known scenarios like Constraints Logic Programming, Attributed Variable and Modal Logic
- extend labelling to formulas $\rightarrow$ we believe the issue of applying labels to formulas, as suggested by [Gabbay, 1996], can open the way to more complex scenarios like fuzzy logic and stochastic computing
- analyse and integrate other LP frameworks
- explore and model, with Labelled tuProlog, domain specific application (e.g. health-care, home-based devices, sensor networks,...)

Thanks

Thank you for your attention

References I

Fruhwirth, T. (1998).
Theory and practice of constraint handling rules.
The Journal of Logic Programming, 37(13):95 – 138.

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 *LNCS*, pages 260–268. Springer.

Jaffar, J. and Lassez, J.-L. (1987).
Constraint logic programming.
In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, pages 111–119, New York, NY, USA. ACM.

Neumerkel, U. (1990).
Extensible Unification by Metastructures.
In *META90*.

Labelled Variables in Logic Programming:

A First Prototype in tuProlog

Roberta Calegari, Enrico Denti, Andrea Omicini

{roberta.calegari, enrico.denti, andrea.omicini}@unibo.it

DISI

Alma Mater Studiorum—Università di Bologna

Talk @ *14th Conference of the Italian Association for Artificial Intelligence*

University of Ferrara - 23rd September 2015