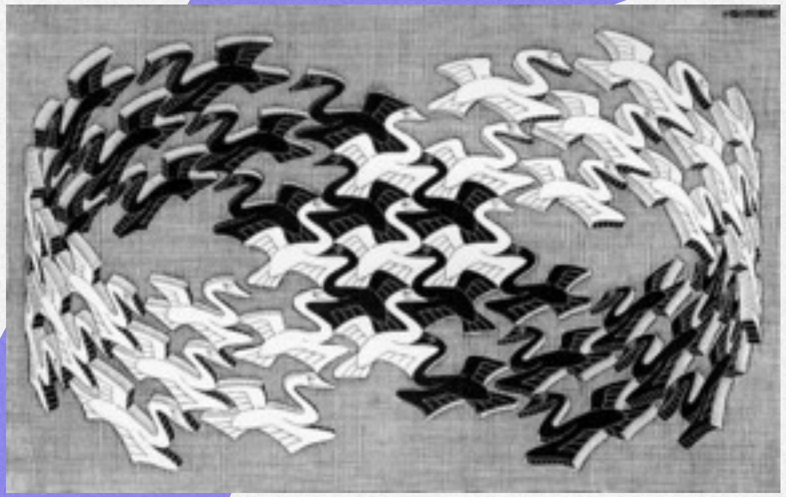
Two-Faced Data

One data fragment has several alternative structural representations tailored toward specific data manipulation approaches.

Vadim Zaytsev



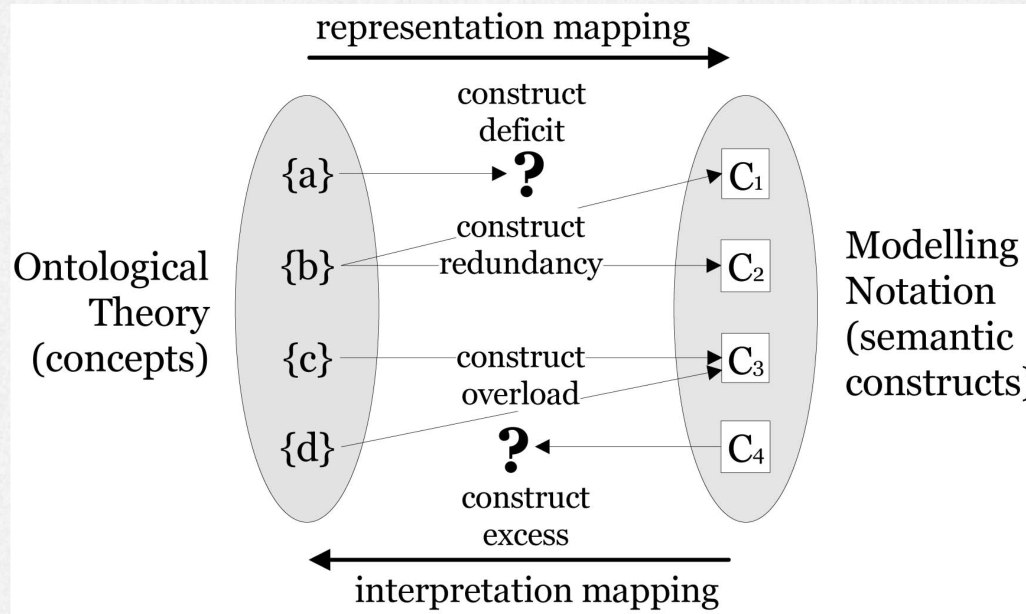
Foreword by Eugene Syriani
Richard Paige
Steffen Zschaler
Huseyin Ergin

2015

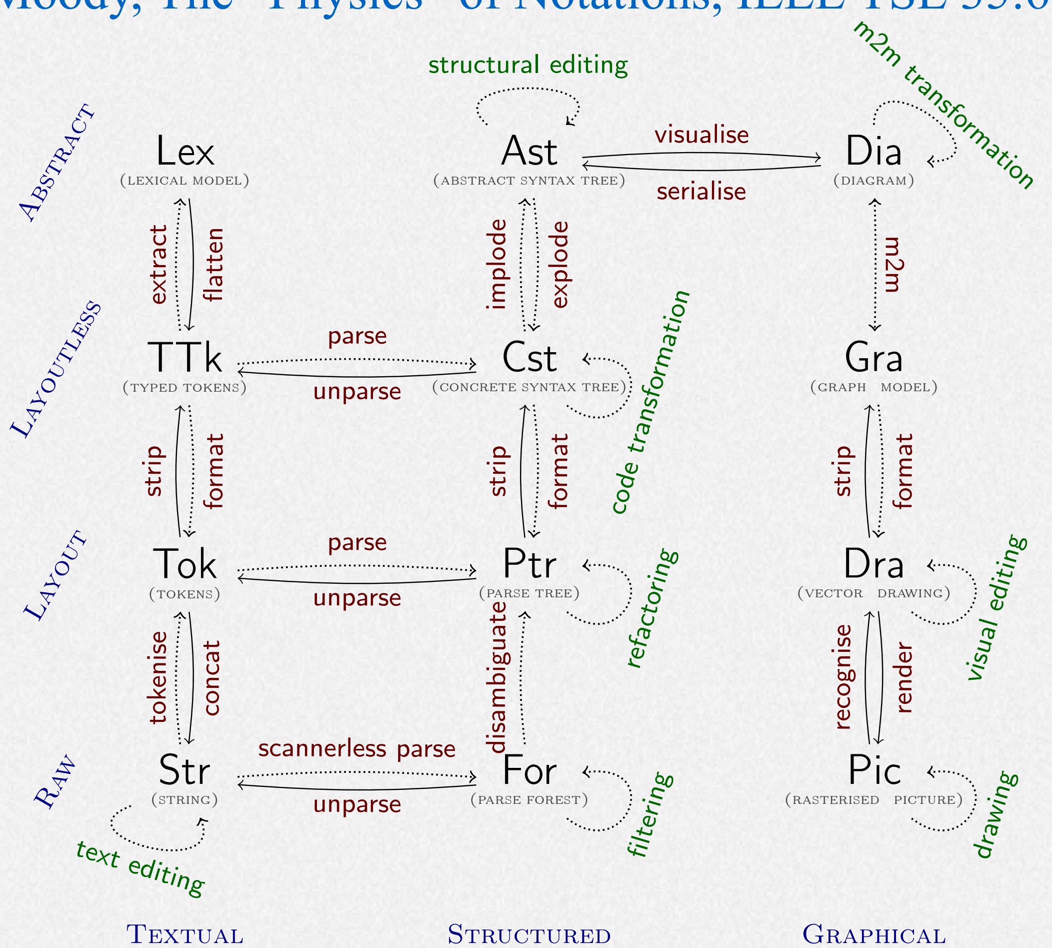
Patterns in Model Engineering at STAF 2015

Two-Faced Data

Motivation



D. L. Moody, The "Physics" of Notations, IEEE TSE 35:6, 2009



V. Zaytsev, A.-H. Bagge, Parsing in a Broad Sense, MoDELS, 2014

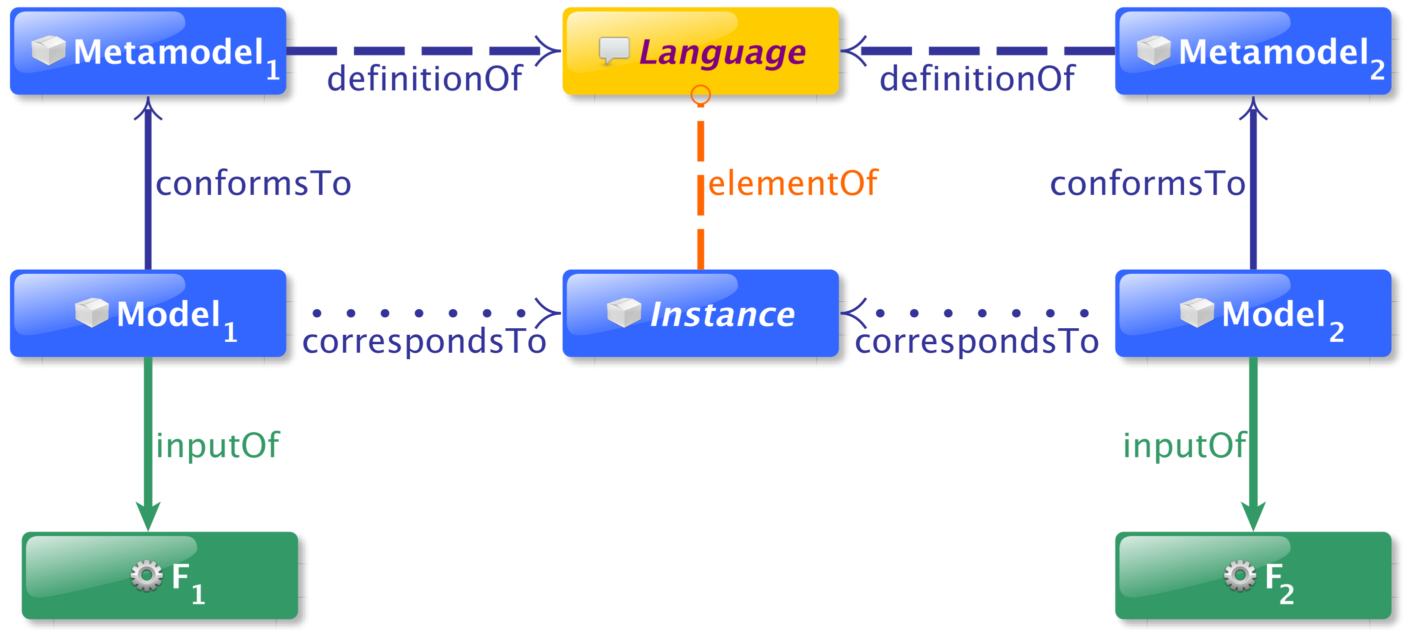
Two-Faced Data

Applicability

- ✓ software language processing
  - ✓ language definition
- ✓ multiple syntaxes
  - ✓ visual and textual
- ✓ (un)serialisation
  - ✓ lossless
- ✓ data-specific algorithms
  - ✓ multiple views
- ✓ multiple tools
  - ✓ interoperability & comfort
- ✓ FORBIDDEN EXAMPLE

Two-Faced Data

Structure



Participants

- ✓ Language
- ✓ Instance
- ✓ Left side
  - ✓ Model
  - ✓ Metamodel
  - ✓ Mapping
- ✓ Right side
  - ✓ Model
  - ✓ Metamodel
  - ✓ Mapping

Two-Faced Data

Collaborations

- ✓ I is an instanceOf L
  - ✓ elementOf
- ✓ Mx models I
  - ✓ correspondsTo
- ✓ MMx models L
  - ✓ representationOf
- ✓ Fx maps Mx
  - ✓ inputOf
- ✓ if Mx is updatable, need BX
- ✓ can have more than two faces

Two-Faced Data

Sample Code

```
data A = foo(bool)
       | bar(list[A] xs)
       ;
```

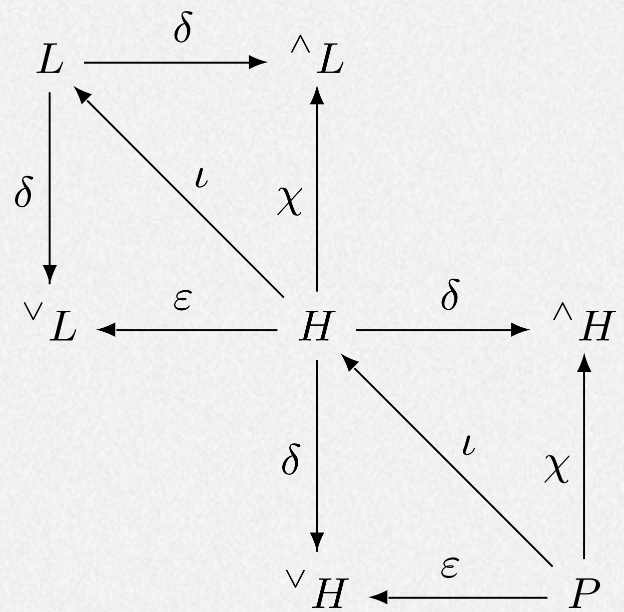
```
syntax A = “foo”?
         | “bar” “(“ A+ “)”;
```

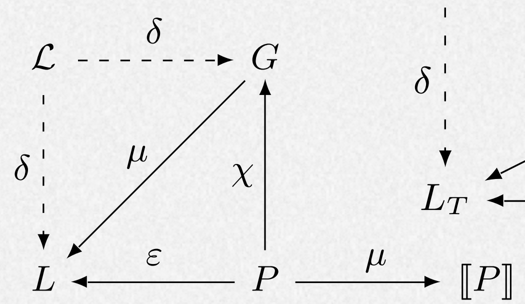
```
T = implode(parse(#A, input)
```

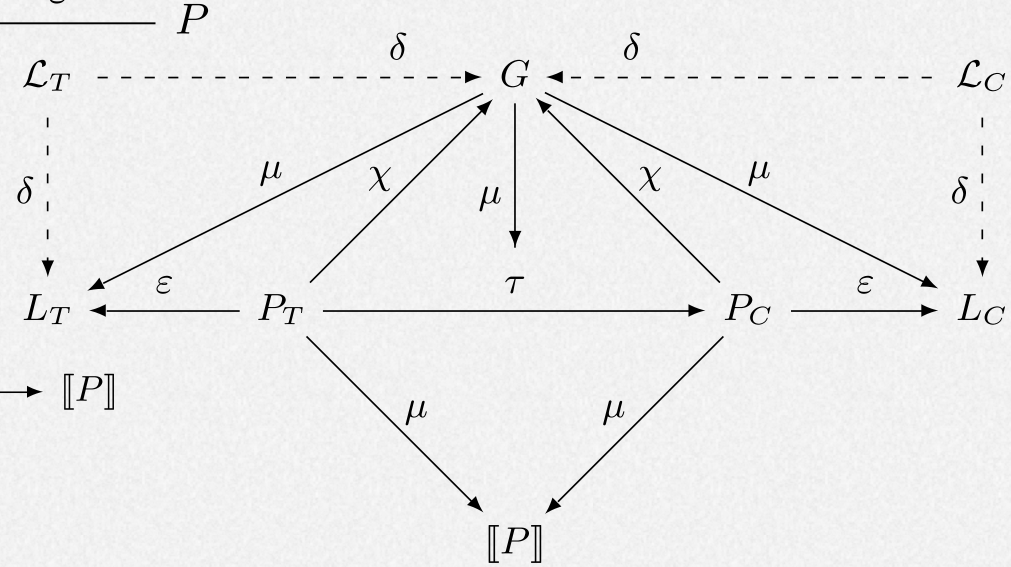
```
visit(T)
{
  foo(True) : cx += 1
  bar([]) => foo(False)
}
```

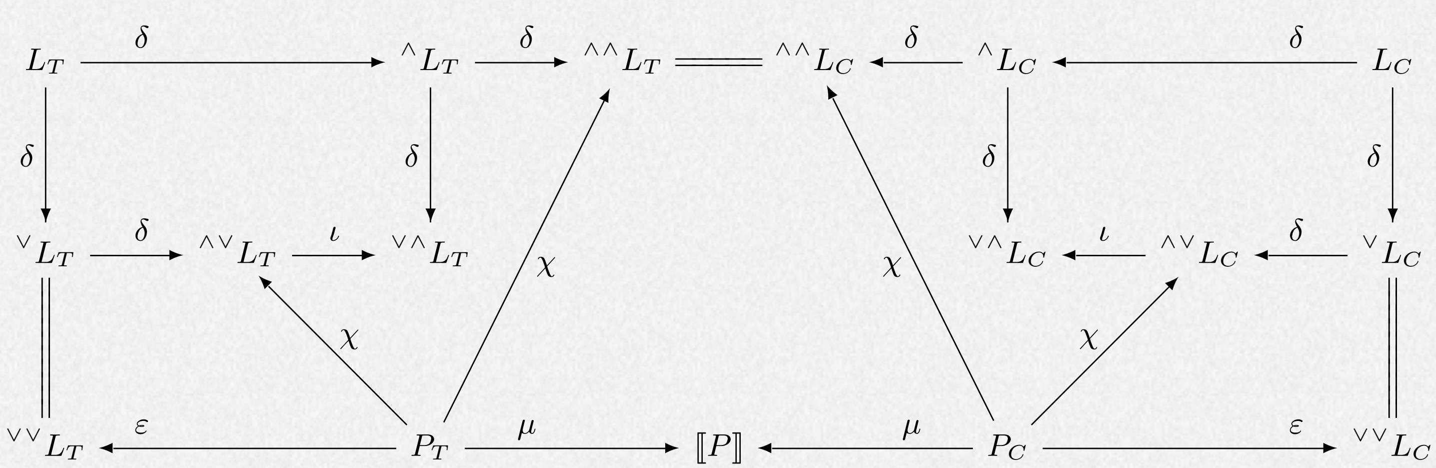
Two-Faced Data

Implementation









Multi-Language Modelling with Higher Order Intensions

Vadim Zaytsev

Universiteit van Amsterdam, The Netherlands, vadim@grammarware.net

Abstract. In the last decade, there have been several fundamental advances in the field of ontological and linguistic metamodelling. They proposed the use of megamodels to link abstract, digital and physical systems with a particular set of useful relations; the distinction between ontological and linguistic layers, identification and separation of them; even formalised the act of modelling and the sense and denotation of a language. In this paper, we propose second order intensions and extensions to more closely model linguistical and ontological conformance and mapping.

1 Formal modelling of languages

In the classic theory of formal languages, a *language* L is defined as a set of sequences of alphabet symbols: $L \subseteq \Sigma^*$ [8]. This definition is easily applicable to textual languages (traditionally associated with programming) and visual languages (traditionally associated with modelling). It is also almost trivially generalisable to graph languages by substituting the reflexive transitive closure in the definition by another operation that (usually recursively) constructs all possible valid language instances from symbols of the alphabet. Even then, all manipulations with the language are done as if it were a set of language instances. For example, a *parser* is generally considered a mapping from the textual language to the tree language in that it assigns a valid parse tree to each valid textual input [17]. Hence, the only relation that is needed to formally describe such processes is an *element of* ($\in$ or ε) relation with rare exceptions like generalised parsers [15] that associate one textual input with a set of several possibly valid parse trees. Since in practice such mapping usually gets implemented to output a *representation* of a set instead of the actual set itself, such cases are more sidesteps than real exceptions: the range of a general parser is a set of parse forests, and each output in an element of this set.

Traditional metamodelling abandons the concept of a language in favour of a *modelling layer* [13,14]. The formal arsenal is expanded to a strict hierarchical structure: the lowest layer is too close to real life to formally decompose and study (e.g., raw data, real life objects, concrete systems), the highest layer is so abstract that it is self-descriptive, and the middle layers M_i contain entities that model entities from the layer below (M_{i-1}) and are expressed in languages defined by entities from the layer above (M_{i+1}). Thus, user data is expressed

Consequences

✓ Questions?