Multi-Language Modelling with Second Order Intensions

Vadim Zaytsev aka @grammarware
MULTI 2015
Multi-Level Transformation

SIDE STORY

(a) S ::= a+ b+ c+ & AB c+ & a+ BC;
    AB ::= a AB? b;
    BC ::= b BC? c;
    AP ::= a+;

(b) S ::= AP b+ c+ & AB c+ & AP BC;
    AB ::= a AB? b;
    BC ::= b BC? c;

(c) S ::= ABP c+ & AB c+ & a+ BC;
    AB ::= a AB? b;
    BC ::= b BC? c;
    ABP ::= a+ b+;

V.Zaytsev, Coupled Transformations of Boolean Grammars and Shared Packed Parse Forests. GCM 2015
Multi-Level Transformation

(a)  
S ::= a+ b+ c+ & AB c+ & a+ BC;  
AB ::= a AB? b;  
BC ::= b BC? c;  

(b)  
S ::= AP b+ c+ & AB c+ & AP BC;  
AB ::= a AB? b;  
BC ::= b BC? c;  
AP ::= a+;  

(c)  
S ::= ABP c+ & AB c+ & a+ BC;  
AB ::= a AB? b;  
BC ::= b BC? c;  
ABP ::= a+ b+;
Motivation

- Software languages
- Complex systems
- Different approaches
- Megamodelling
Language Theory

\[ G \rightarrow \mu \rightarrow L \]
\[ \mu \rightarrow X \rightarrow \varepsilon \rightarrow P \]
\[ \mu \rightarrow S \]
Language Theory
Language Theory
Language Theory

\[
\begin{align*}
G & \quad \mu \\
L & \quad \varepsilon \\
P & \quad \mu \\
\end{align*}
\]
Language Theory

\[
\begin{align*}
L & \xrightarrow{\epsilon} P \\
\mu \quad & \quad \quad \quad \quad \quad \quad \quad \\
G & \xrightarrow{\mu} [P]
\end{align*}
\]
Language Theory
Language Theory

\[ L \quad \cdots \cdots \quad G \]

\[ \{P\} \quad \varepsilon \quad P \]
Language Theory

\[ L \xrightarrow{\delta^\wedge} G \]

\[ \delta_v \]

\[ \{P\} \xleftarrow{\epsilon} P \]

\[ \chi \]
Language Theory

\[ L \xrightarrow{\delta^\wedge} \wedge L \]
\[ \wedge L \xrightarrow{\delta_v} L \]
\[ \wedge L \xrightarrow{\varepsilon} P \]
\[ L \xleftarrow{\delta_v} \wedge L \]

\[ L \xrightarrow{\delta^\wedge} \wedge L \]
\[ \wedge L \xrightarrow{\delta_v} L \]
\[ \wedge L \xrightarrow{\varepsilon} P \]
\[ L \xleftarrow{\delta_v} \wedge L \]
The XML schema (xsd file) is a $\mu\alpha$ representation of the system (wrt. a given intention I). The schema is used to generate an EMF model (ecore file). The model and the schema share the same intention I, as shown by $\mu\alpha$ relations. The model is then used to generate a generation model (genmodel), which is also in a $\mu\alpha$ relation with the system. The .genmodel contains additional information (wrt. the model) to drive the code generation process; therefore it is the target of a partial $\mu\gamma$ relation. Three Java projects are generated from the generation model: model, edit, and editor. Edit.java is a Java projection of the model, thus it is a $\mu\alpha/I$ representation of the system as well. Edit.java contains general editing mechanisms (not dependent on the graphical user interface) and uses the java projection of the model (represented with another $\mu\alpha$ relation). Finally, Editor.java provides end-user editing facilities to visualize models, using a tree-based explorator.

6.8 Modeling

This paper is entitled "modeling modeling modelling." This is to reflect the fact that the presented work is about building a formal model (F in Fig. 17) of a language (L in the picture), which in turn is a representation for a set of models of systems (M and S in the picture). This journal paper (modelling modeling modeling) extends the conference paper (modeling modeling) by a third level of modeling. Hence, this third modeling is the contribution of this paper (Fig. 17).

7 Conclusion

This work analyzes various definitions of models, as found in the related works, and proposes a modeling language which can be used as a foundation to represent the various representation relations between models, metamodels and languages.

Our language focuses on representation relations between modeling artifacts, without actually trying to understand the nature of these artifacts. Ignoring the details of their internal structure appears to be very effective because it magnifies the fact that modeling is a matter of relations and roles, and not intrinsic to the artifacts.

We have identified five kinds of representation relation (based on their intention), two natures (analytical and synthetic), and taken causal dependencies and transitivity into account. We have also introduced a formal definition of the domain of modeling as well as precise semantics for relations between things that are manipulated when modeling. We have illustrated our approach with several simple examples, drawn from the software engineering domain.

From a practical point of view, we hope that this step toward a better understanding of representation relations will serve as a basis for intention-aware metamodeling tools, in the same way as relational algebra triggered the development of efficient databases. One step in that direction would consist in formally capturing the operational semantics of the modeling language.

Acknowledgments

This paper is the result of numerous informal discussions we have had with so many people that it is almost impossible to enumerate them all here. We would like to especially thank a few of them: including Jean-Marie Favre, Thomas Kuehne, Colin Atkinson, Marc Pantel, Christophe Gaston, and Régis Fleurquin. We would also like to acknowledge the invaluable comments of anonymous reviewers of an earlier version of this paper.

References

1. Molière: Le Bourgeois gentilhomme (1607)
(Modelling Modelling)
Back to Languages

\[ L \xrightarrow{\delta_\land} \land L \quad \land L \xrightarrow{\delta_\lor} L \\]

\[ \lor L \xrightarrow{\varepsilon} P \quad P \xrightarrow{\chi} \lor L \]
The Rôle of Grammar

- $G \sim \mu \sim \land L$ (analytical/prescriptive/conformance)
- $G \sim \mu \sim L$ (commitment)
- $G \sim \mu \sim \lor L$ (generative/derivational)
- $G \sim \mu \sim \pi$ (transformational/mapping)
- $G \sim \mu \sim \kappa$ (same but “downwards”)
Fig. 1. Bidirectional megamodel of parsing. Dotted lines denote mappings that rely on either lexical or syntactic definitions; solid lines denote universally defined mappings. The loops are examples of transformations.

Let us first introduce the kinds of artefacts we will use for the remainder of the paper:

- **Str** — as string.
- **Tok** — a finite sequence of strings (called tokens) which, when concatenated, yields Str. Includes spaces, line breaks, comments, etc — collectively, layout.
- **TTk** — a finite sequence of typed tokens, with layout removed, some classified as numbers of strings, etc.
- **Lex** — a lexical source model\[28,29\] that adds grouping to typing; in fact possibly incomplete tree connecting most tokens together in one structure.
- **For** — a forest of parse trees, a parse graph or an ambiguous parse tree with sharing; a tree-like structure that models Str according to a syntactic definition.

4 Artifacts and Mappings
Parsing

\[ \text{Lt} \quad \delta_{\wedge} \quad \text{G} \]

\[ \delta_{\lor} \]

\[ \{Pt\} \quad \epsilon \quad Pt \]

\[ \mu \quad \chi \]
Parsing

\[
\begin{align*}
\delta^\wedge & : Lt \rightarrow G \\
\delta^\vee & : Lt \rightarrow \{Pt\} \\
\mu & : Pt \rightarrow \chi \\
\delta^\wedge & : Lc \rightarrow G \\
\delta^\vee & : Lc \rightarrow \{Pc\} \\
\end{align*}
\]
Parsing

\[ Lt \quad \delta_\wedge \quad G \quad \delta_\wedge \quad Lc \]

\[ \{Pt\} \quad \epsilon \quad Pt \quad \tau \quad Pc \quad \epsilon \quad \{Pc\} \]

\[ \delta_\vee \quad \mu \quad \chi \quad \mu \quad \chi \quad \mu \]

\[ \mu \quad \mu \quad \mu \quad \mu \quad \mu \quad \mu \]
First Order
\[
L \xrightarrow{\delta_\wedge} \land L \xrightarrow{\delta_\wedge} \land \land L \\
\delta_\lor \downarrow \quad \delta_\lor \downarrow \\
\lor L \xrightarrow{\delta_\wedge} \land \lor L \\
\downarrow \quad \downarrow \\
\lor \lor L \xrightarrow{\delta_\lor} \land \lor L \\
\downarrow \\
\lor \lor \lor L
\]

Second Order