The Language of Languages Research Project: Unifying Concepts Expressed Across Different Notations

James R. Douglass
The Boeing Company
jamie.douglass@boeing.com

Nicholas Chen Ralph E. Johnson
University of Illinois at Urbana-Champaign
{nchen, rjohnson}@illinois.edu

Abstract

Maintaining the consistency of multiple notations used in large projects is daunting. Language of Languages(LoLs) is our experimental language workbench that fulfills a frequently overlooked but important role: *unify* the different notations so developers can better understand and evolve a project. Due to the impossibility of anticipating all the notations that may be used in a project, LoLs adopts a language agnostic view and supports different notations from free-form text to graphical forms and shapes. Our demo begins by illustrating the fundamental ideas of LoLs through building a calculator that supports multiple notations; the demo concludes with more advanced projects that exemplify the extent of our multi-notation support.

Categories and Subject Descriptors D.2.2 [*Software Engineering*]: Design Tools and Techniques; D.3.3 [*Programming Languages*]: Language Constructs and Features

General Terms Design, Languages

Keywords Language Workbench, Modeling, Smalltalk

1 Introduction

Different programming languages, libraries and frameworks provide **different** notations for representing components. For instance, state machines, a common component used in many engineering disciplines, can be expressed using VHDL code (text) or state diagrams (graphics). Large projects seldom, if ever, commit to just a single notation. Instead, multiple notations are used, with developers choosing the most expressive one for the task at hand.

Using multiple notations creates several challenges: (i) **communication** – how would my colleagues understand the notations that I have chosen? (ii) **correctness** – how would we verify that notations are consistent with one another? LoLs [1] has an ambitious goal: distill the core concepts from each component and attempt to unify these underlying concepts across different notations.

This unification maintains consistency and facilitates *virtual integration* of components as the project evolves. Virtual integration, done early and continuously, enables prompt detection of problems and reduces the cost of fixing them. Virtual integration is a key component of the System Architecture Virtual Integration (SAVI) project [3], and we see LoLs as one feasible approach for supporting it.

SPLASH'11 Companion, October 22–27, 2011, Portland, Oregon, USA.
ACM 978-1-4503-0940-0/11/10.

2 Language of Languages Approach

Consider the simple expression "3 + 4". There are different notations to represent this expression: in Western Numerals, Roman Numerals, or even using graphics. Regardless of the notation, the underlying *concept* remains the same: we are adding two numbers together. How can we represent this concept across different notations?

The fundamental building block of LoLs is the `Language Element`. Everything has a corresponding `Language Element`. In Figure 1, the symbols "3", "+" and "4" have corresponding `Language Elements`. These `Language Elements` form a `Language Element Tree`. Each `Language Element` has a corresponding `Concept`. In our example, we have two `Concepts`: Addition and Number. Each `Concept`, in turn, has several `Language Definitions` that it can select from. In our example, we have three `Language Definitions`: Western, Roman and Graphical. The Western Definition is being used in our example (indicated by the check mark).

The fundamental idea of LoLs is thinking in terms of reusable `Concepts` that apply across different notations. Each notation is supported by a particular `Language Definition`. `Language Defintions` do not stand alone; they can reference other `Concepts`. For instance, it is necessary for the `Addition Concept` to reference and use the `Number Concept`. Each `Language Definition` supports **three** operations: checking, parsing and projecting.

Checking The checking operation validates if a `Language Definition` is applicable for the current context.

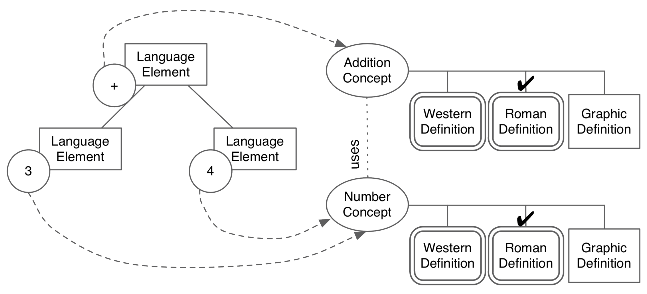


Figure 1. Fundamental building blocks in LoLs for the "3+4" example.

Parsing The parsing operation recognizes the notation that is being used and builds a `Language Element Tree` from each construct of interest.

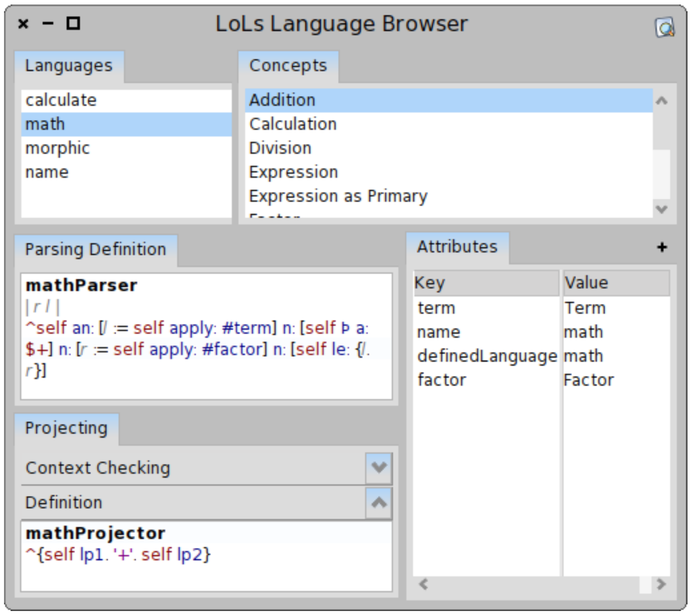
Projecting The projecting operation interprets or transforms `Language Elements`. This operation can take a Language Element Tree and reduce it to a single value e.g. the tree in Figure 1 is reduced to the value 7. Or, it can transform a Language Element Tree from one notation (Western Numerals) to a different notation (Graphical).

Our approach allows us to easily "plug-n-play" different concepts. New `Concepts` and `Language Definitions` can easily be written by the developer or extended from existing ones. Judicious use of `Concepts` and `Language Definitions` in LoLs affords the developer a lot of freedom in expressing the underlying domain knowledge. It is an open research question to catalog the common concepts that apply to various projects.

3 Agenda for Demo

We begin by showing how to build a basic calculator in LoLs. We start with a calculator that only understands Western Numerals and show how to add support for graphical notations in a modular fashion. Starting with a simple example provides the audience with both a high-level overview of LoLs and also a deeper understanding of how our tool is implemented. Figure 2 shows the preliminary user interface for our tool. After familiarizing the audience with the basic ideas, we will demonstrate how to support projects that require the use of multiple notations. These projects illustrate the extent of our support for multiple notations.

The SPLASH community has always been at the forefront of new ideas and would be a befitting audience for LoLs. LoLs is a novel idea that presents not only a new way of thinking about programming but also a minimalistic yet flexible approach that is easily extensible. Our current bootstrap implementation in Squeak Smalltalk [2] makes good use of metacircularity (LoLs is defined in terms of itself) and well-known design patterns for its core. We leverage the Morphic graphic system in Squeak to support graphical notations. LoLs has a small core that is easy to understand and extend for different purposes.



Languages
calculate
math
morphic
name

Concepts
Addition
Calculation
Division
Expression
Expression as Primary

```
mathParser
| r l |
^self an: [l := self apply: #term] n: [self Þ a:
$+] n: [r := self apply: #factor] n: [self le: {l.
r}]
```

```
mathProjector
^{self lp1. '+'. self lp2}
```

Key	Value
term	Term
name	math
definedLanguage	math
factor	Factor

Figure 2. Language Workbench in bootstrap version of LoLs.

The LoLs bootstrap and examples presented will be available for download from `www.LanguageOfLanguages.org` after our demo. LoLs is an open source project under the MIT license and we encourage participation and contributions from the community.

4 Presenters

Jamie Douglass and Nicholas Chen are the main developers of LoLs. Jamie Douglass is an Architect and Associate Technical Fellow within the Office of IT Chief Engineer at the Boeing Company. His research interests include language based integration, modeling and software development. Nicholas Chen is a PhD candidate in the Software Architecture Group at the University of Illinois. His research interest lies in mining patterns of software evolution and creating flexible software engineering tools to support them.

Acknowledgments

The authors thank Jeff Overbey, Kathleen Chalas and Eric Reed for feedback on earlier drafts of this manuscript.

References

[1] J. R. Douglass. Language of Languages for Flexible Development. In *FlexiTools@SPLASH2010*, 2010.

[2] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay. Back to the future: the story of Squeak, a practical Smalltalk written in itself. *SIGPLAN Not.*, 32:318–326, October 1997. ISSN 0362-1340.

[3] D. Redman, D. Ward, J. Chilenski, and G. Pollari. Virtual Integration for Improved System Design. In *The First Analytic Virtual Integratio of Cyber-Physical Systems Workshop*, 2010.