AMIDOL Milestone 13 and 14 Report

Eric Davis, Alec Theriault, and Ryan Wright

Galois, Inc.
421 SW 6th Ave., Ste. 300
Portland, OR 97204

Contents

1. Introduction

Part I

Milestone 13

2. Recent Extensions to the AMIDOL Framework

The UI for comparing results of multiple models has been extended to support a rich language for combining traces. This is particularly useful for constructing derived measures without needing to re-run the model. For instance, this might be used in a situation where an epidemiological model tracks multiple strains of a virus separately, but the hospital data only keeps aggregate data. Another similar case is when the scientist doing modelling is trying to quickly line up peaks of infected populations from model result and real world data. Our language contains primitives like shift which simplify this sort of transformation.

A completely new VDSOL that is designed to take a system of differential equations written in LaTeX format has been added. The idea behind this new language is to simplify the work of going from a model written in an academic paper to an AMIDOL model. Scientists enter LaTeX equations, constants, and initial conditions in a text box. Beside this input, they get a real-time LaTeX preview of their equations. From there, the model is compiled into the AMIDOL IR so that it can be executed and compared to results from other models (which may have been designed in completely different VDSOLs).

Finally, we are in the process of experimenting with model composition in the backend, with the goal of finding a minimal intuitive language for combining models. To this end, we've been refining our existing state-sharing composition operators and have started to experiment with other techniques revolving around substitution. Most of these changes are still not exposed to end users, since we are not yet sure what a good visual representation would look like.

2.1. VDSOLs for Mathematical Languages

Adding a VDSOL for LaTeX input equations: The VDSOL renders LaTeX equations using KaTeX, a fast browser-based Javascript library designed for this purpose. Once a user submits a system of differential equations, the backend tries to parse each line as a differential equation, initial condition, or constant. This step is complicated by the

|galois|

fact that the LaTeX source for equations can sometimes be interpreted in multiple ways (for instance: is '\fracpsi2' the variable 'psi' divided by two, or the product of 'p', 's', 'i', and '0.5'). To solve this ambiguity, we require that implied multiplication include at least a space to separate the factors (eg. 'p s i' vs 'psi'). The final step is to convert the system of differential equations into an AMIDOL model. This is fairly straightforward: variables in the equations turn into states and the right-hand side of the equations turns into events.

2.2. Definition of Derived Measures

Updating the comparison UI to support a richer mathematical language for derived measures: In order to implement this, we've moved the work of combining data traces from the UI to the backend. The language for describing derived measures supports translations, linear distortions, component-wise arithmetic, as well as a couple built-in math functions. The backend contains a parser for this language as well as an interpreter. One of the challenges here is around how to interpolate when combing data traces whose measures had different time ranges or step sizes.

3. AMIDOL Demo Instructions

4. AMIDOL Performance for Real-World Systems and Processes

	Model	**SIR**	**SIIR**	**Predator-Prey**	**SIR with vital dynamics**
Time	Graph VDSOL to IR compilation	1.7ms	1.7ms	1.1ms	1.6ms
	Julia to Graph VDSOL (including ontology grounding search)	1.8s - 6s	2.1s - 6s	n / a	n / a
	IR compilation to Python backend	190ms	160ms	30ms	160ms
	IR compilation to Julia backend	24ms	25ms	15ms	52ms
	Execution of Python backend output	2.5s	2.4s	1.1s	2.5s
	Execution of Julia backend output	9.8s	12s	7.5s	7.0s
Lines of Code	Python backend output	40	45	39	47
	Julia backend output	44	62	51	60

|galois|

Part II

Milestone 14

5. Final Prototype Development

6. AMIDOL as a Service

7. Model Algebras and Transformations

7.1. Composing Models in AMIDOL

7.2. Substituting Models in AMIDOL