Presenting a Type System for Typed Presentations of Type System Representations

Jason Reed*
jcreed@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

March 22, 2007

Abstract

Merely doing research is well and good, but one finds that there inevitably comes the time when one must present the fruits of one's diligent work to other, generally more ignorant, persons. This task is further complicated by, on the one hand, the proliferation of incompatible presentation software packages, and on the other, the ubiquitous possibility of making errors in the presentation. We propose an abstraction intended as an underlying type-theoretic framework to both capture the common features of diverse presentation formalisms, and to provide dynamic guarantees of static specifications. Examples of errors we would like to rule out include violating community-negotiated conventions that a talk should have a certain minimum number of graphs, and spurious duplication of material.

Keywords: Powerpoint, Types, Static Analysis

1 Introduction

Merely doing research is well and good, but one finds that there inevitably comes the time when one must present the fruits of one's diligent work to other, generally more ignorant, persons. This task is further complicated by, on the one hand, the proliferation of incompatible presentation software packages, and on the other, the ubiquitous possibility of making errors in the presentation. We propose an abstraction intended as an underlying type-theoretic framework to both capture the common features of diverse presentation formalisms, and to provide dynamic guarantees of static specifications. Examples of errors we would like to rule out include violating community-negotiated conventions that a talk should have a certain minimum number of graphs, spurious duplication of material, and spurious duplication of material.

*This research partially supported by Grant WEH-4625 "Pile of money I found in Wean 4625"

2 Language

For space reasons, herein we present a rather simplified version of our type system for presentations. We assume familiarity with the (...)-Calculus [**?**] developed by, you know, those people who invented it. The syntax is as follows.

Presentations P	::=	$\vec{S}$
Slides S	::=	$\mathsf{intro} \mid \mathsf{outline} \mid \mathsf{diag}$
		$\mid \mathsf{cont}(\vec{\beta}) \mid \mathsf{conc}$
Bullets β	::=	$\pi \mid \epsilon \mid \mu \mid \nu \mid \phi$
		$\mid \iota \mid \alpha \mid \gamma \mid \lambda(\vec{\beta})$
Refinements ρ	::=	$\rho_1 \rightarrow \rho_2 \mid X \mid \top \mid @ \mid \$$
Types A	::=	$\mathsf{slide} \mid \bullet \mid \ldots$
Kinds K	::=	$\mathrm{type} \mid \ldots$
Hyperkinds H	::=	$\mathrm{kind} \mid \ldots$
Sorts σ	::=	$a \mid k \mid h \mid \square \mid \bigcirc \mid * \mid \star$
Universes L	::=	$A \mid K \mid H \mid \ldots$
Cosmologies $\mathfrak{C}$	::=	$\ldots$
Philosophies $\mathfrak{P}$	::=	$\ldots$
WTFs $[\![\overline{\mathfrak{M}}^{\circ}]\!]$	::=	$[\![\overline{\mathfrak{M}}^{\circ}]\!] \mid [\![\overline{\mathfrak{M}}^{\circ}]\!]$

Figure 1: Pile of Money

2.1 Slides

Given the uniformity of presentation styles, we can easily classify slides by their type.

Slide	Meaning
intro	Introduction Slide
outline	Outline Slide
diag	Intimidating Diagram
cont($\vec{\beta}$)	Slide With Actual Content
conc	Conclusion Slide

2.2 Bullet Points

We similarly divide bullet points by their function.

Bullet	Meaning
π	Plain Prose
ϵ	Excuses
μ	Misdirection
ϕ	Formula
ι	Inference Rule
ν	Non Sequitur
α	Impenetr. Mass of Abbrevs.
γ	Impenetrable Mass of Greek
$\lambda(\vec{\beta})$	Nested Bullet List

2.3 Typing

Nobody really looks at most of the obvious, run-of-the-mill, completely standard judgments and inference rules such as

$$\Gamma \vdash S : \mathsf{slide}$$

or

$$\frac{\Gamma \vdash \beta_i : \bullet \qquad (\forall i)}{\Gamma \vdash \mathsf{intro} + \mathsf{outline} + \mathsf{diag} + \mathsf{cont}(\vec{\beta}) + \mathsf{conc} : \mathsf{presentation}}$$

anyway, so we decided not to waste space on any more of them except for the most alarming ones. On the other hand, the remaining inference rules have been determined to be unsuitable for human consumption, and we prudently omit them also. We expect them to be used in animal feed, and common garden mulch.

2.4 Refinements

The essential aim of our proposal is to enable the description of refinements of the type of presentations. A presentation that contains too few graphs, too many or too few inference rules, lacks a cute animation at the end, or completely fails to explain how to achieve the research result claimed and uses feeble misdirection to avoid drawing attention to this fact should be rejected.

2.5 Aspects

On the other hand, aspects, as any OOP researcher worth his or her salt knows, are a feature of verbs use to indicate how the action of a verb takes place over time. For example, there are the following aspects.

Aspect	Meaning
Imperfect	Action still going on
Perfect	Action completed
Pluperfect	Sounds funny
Aorist	Sounds even funnier
Simple(r) Past	Hmph! Kids these days!
Progressive	Drives Prius, Eats Granola
"Around" Advice	For debugging
"Before" Advice	For logging
"After" Advice	Too late, therefore useless

Our contribution to Aspect-Oriented Presentation is the **powerpoint-cut**, by which a slide can be inserted at the last minute just as question is asked. This works by *[XXX remember to write this section before submitting!]*.

3 Related Work

While no previous work has presented an exact and complete formal definition of a type system for presentations there has been occasional interest in the intersection between computer science and theatrical presentation construed more broadly. Most of this work has focused on practical systems that are immediately deployable in the field. For example, Hopper, Murger, and Snivelin invented the so-called 'HMS Semaphore' [HMS01] to prevent audience members from beginning concurrent computations ('impertinent questions') and thereby preventing the talk from achieving hard real-time bounds. The Myfair Scheduler [HH64] attempts to provide the same guarantees by refining the language.

More type-theoretic approaches to the problem include the invention of the *phantom type of the operad* due to by Andrew Lloyd Weppél. [Wep92]

Håvard Bringinda and his students at the University of Oslo have been working trying to unify statistical mechanics and category theory. We hope the study of Bringinda Noise, and Bringinda Functors to be fruitful.

4 Future Work

We hope to extend the current type system to include other expositional components of computer science research, in particular conference and journal papers, and technical reports. This should be quite reasonable if we use for document preparation a modern functional programming language. The only obstacle to ensuring of such documents that they contain no errors or other anomalies is `*** ERROR: Redundant Hypokind in SECTION $0xFE 'future work' at ptsptspts.mtx`

5 Conclusion

References

[HH64] A. Hepburn and R Harrison. The Myfair scheduler. In J. Halpern, editor, *Proceedins of th' Haighth Annual Symposium on the Linguistics what is used in Computer Science (LICS'01)*, pages 221–230, Boston, Massachusetts, May 1964. IEEE Computer Society Press.

[HMS01] B. Hopper, B. Murger, and B. Snivelin. A concrete proposal for eliminating interruptions. Technical report, Adobe Systems, 2001.

[Wep92] Andrew Lloyd Weppél. Phantom type of the operad. Unpublished manuscript, 1992.