Functorial programming for strict/lax ω-categories in Lambdapi

https://github.com/hotdocx/emdash

We report on an ongoing experiment, `emdash version 2`, whose goal is a new *type-theoretical* definition of ω-categories that is both *internal* (expressed inside dependent type theory) and *computational* (amenable to normalization by rewriting). The implementation target is the Lambdapi logical framework [2], following [1]. In this sense, the Lambdapi specification itself already behaves like a small programming language/proof assistant for ω-categories: the theory is formulated internally and computations are performed by normalization; what remains is chiefly elaboration and user-facing syntax in TypeScript.

The central construction is similar to the recent "bridge type" approach to internal parametricity without an interval [3] and, at the same time, by type-theoretic accounts of weak higher categories [4]. We use a *dependent hom / comma / arrow construction* that directly organizes "cells over a base arrow" in a simplicial manner.

Concretely, let B be a category and let E be a dependent category over B (informally, a fibration $E\colon B \to \mathbf{Cat}$, or a more general isofibration). Fix a base object $b_0 \in B$ and a fibre object $e_0 \in E(b_0)$. We construct a Cat-valued functor that assigns to a base arrow $b_{01} : b_0 \to b_1$ and a fibre object $e_1 \in E(b_1)$ a category of "morphisms from the transport of e_0 along b_{01} to e_1 in the fibre over b_1":

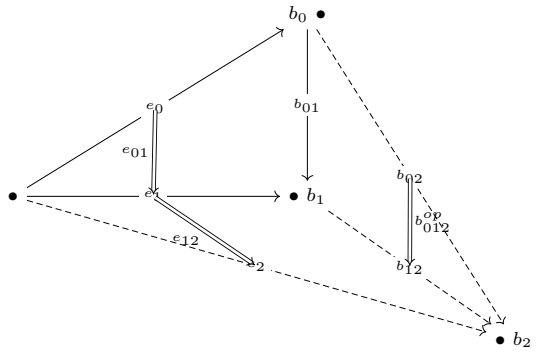
$$\mathrm{Homd}_E(e_0, (-,-)) \;:\; E \times_B \big(\mathrm{Hom}_B(b_0, -)\big)^{\mathrm{op}} \longrightarrow \mathbf{Cat}.$$

$$\mathrm{Homd}_E(e_0, (e_1, b_{01})) \;\coloneqq\; \mathrm{Hom}_{E(b_1)}\big((b_{01})!\, e_0,\; e_1\big),$$

In *internalized* syntax notation, this is

$$\mathrm{Homd}_E \;:\; \Pi b_0,\; E[b_0]^{\mathrm{op}} \to \big(\Sigma b_1,\; E[b_1] \times \mathrm{Hom}_B(b_0, b_1)^{\mathrm{op}}\big) \to \mathbf{Cat}.$$

Iterating it yields a simplicial presentation of higher cells (triangles, "surfaces", and higher simplices) where "stacking" of 2-cells along a 1-cell is expressed over a chosen base edge:



We aim to keep the interface conceptual and type-theoretic rather than explicitly combinatorial [5].

As an application, we outline a computational formulation of adjunctions in which unit and counit are first-class 2-cell data and the triangle identities are oriented as cut-elimination reductions on composites, following [1].

$$\varepsilon_f \circ L(\eta_g) \;\;\rightsquigarrow\;\; f \circ L(g)$$

[1] K. Došen and Z. Petrić, *Cut-Elimination in Categories*.

[2] F. Blanqui *et al.*, *The Lambdapi Logical Framework*, https://github.com/Deducteam/lambdapi.

[3] T. Altenkirch, Y. Chamoun, A. Kaposi, and M. Shulman, *Internal Parametricity, without an Interval*.

[4] E. Finster and S. Mimram, *A Type-Theoretical Definition of Weak ω-Categories*.

[5] H. Herbelin and R. Ramachandra, *A parametricity-based formalization of semi-simplicial and semi-cubical sets*.