Levitation without the sugar

December 3, 2015

We present the low-level type theory of "The Gentle Art of Levitation" [1] in a more familiar rule-based style, organised as much as possible into the usual categories of formation, introduction, etc. The only real (but inconsequential) difference is that we require type constructor to always be fully applied. Otherwise we follow the paper closely, including the use of colour[1] to distinguish the different categories of syntax. In brief, type formers are blue, constructors red, defined functions green, bound variables purple, and meta-variables stay uncoloured.

1 Standard

We recall the standard rules for unit- and Σ-types, mainly to introduce the slightly non-standard syntax.

1.1 Unit type

Formation

$$\frac{\Gamma \vdash}{\Gamma \vdash 1 : \textsc{Set}}$$

Introduction

$$\frac{\Gamma \vdash}{\Gamma \vdash [] : 1}$$

1.2 Σ-types

Formation

$$\frac{\Gamma \vdash S : \textsc{Set} \qquad \Gamma; x : S \vdash T : \textsc{Set}}{\Gamma \vdash (x : S) \times T : \textsc{Set}}$$

Introduction

$$\frac{\Gamma \vdash S : \textsc{Set} \qquad \Gamma; x : S \vdash T : \textsc{Set}}{\Gamma \vdash [s, t]_{x.T} : (x : S) \times T}$$

To save space, we will drop the type annotation when it is clear from the context. We will also write

$$\left[\begin{array}{c} t_1 \\ \dots \\ t_n \end{array}\right]$$

for the right-nested pair $[t_1, [\dots, [t_n, []] \dots]]$.

[1] See `http://strictlypositive.org/dpm/colour.html` for the rationale behind the choice of colours.

Elimination

$$\frac{\Gamma \vdash p : (x : S) \times T}{\Gamma \vdash \pi_0\ p : S} \qquad \frac{\Gamma \vdash p : (x : S) \times T}{\Gamma \vdash \pi_1\ p : T[\pi_0\ p/x]}$$

Computation

$$\frac{\Gamma \vdash s : S \qquad \Gamma \vdash t : T[s/x]}{\Gamma \vdash \pi_0\ [s,t]_{x.T} \equiv s : S} \qquad \frac{\Gamma \vdash s : S \qquad \Gamma \vdash t : T[s/x]}{\Gamma \vdash \pi_1\ [s,t]_{x.T} \equiv t : T[s/x]}$$

2 Finite sets and small products

2.1 Tags

The purpose of tags is to provide the user with identifiers to be used as constructor names. As this is their only purpose, they only come with an introduction but not elimination rule.

Formation

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Tag} : \textsc{Set}}$$

Introduction

$$\frac{\Gamma \vdash}{\Gamma \vdash \text{`}s : \textsc{Set}}\ s \text{ is a valid identifier}$$

The paper leaves open what it means for an identifier to be "valid".

2.2 Small products indexed by En

En-formation

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{En} : \textsc{Set}}$$

$$\frac{\Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \#E : \textsc{Set}}$$

$$\frac{\Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash P : \#E \to \textsc{Set}}{\Gamma \vdash \pi\, E\, P : \textsc{Set}}$$

En-introduction

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{nE} : \mathsf{En}}$$

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \mathsf{cE}\, t\, E : \mathsf{En}}$$

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En}}{\Gamma \vdash 0 : \#(\mathsf{cE}\, t\, E)}$$

$$\frac{\Gamma \vdash n : \#E}{\Gamma \vdash 1{+}n : \#(\mathsf{cE}\, t\, E)}$$

$$\frac{\Gamma \vdash P : \#\mathsf{nE} \to \textsc{Set}}{\Gamma \vdash \pi\, \mathsf{nE}\, P \equiv 1 : \textsc{Set}}$$

$$\frac{\begin{array}{c}\Gamma \vdash t : \mathsf{Tag} \\ \Gamma \vdash E : \mathsf{En} \\ \Gamma \vdash P : \#(\mathsf{cE}\, t\, E) \to \textsc{Set}\end{array}}{\Gamma \vdash \pi\, (\mathsf{cE}\, t\, E)\, P \equiv P\, 0 \times \pi\, E\, (\lambda_{\#E} x.\, P\, (1{+}x)) : \textsc{Set}}$$

En-elimination

$$\frac{\Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash P : \#E \to \textsc{Set} \qquad \Gamma \vdash b : \pi\, E\, P \qquad \Gamma \vdash x : \#E}{\Gamma \vdash \mathsf{switch}\, E\, P\, b\, x : P\, x}$$

En-computation

$$\frac{\begin{array}{c}\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash P : \#(\mathsf{cE}\, t\, E) \to \textsc{Set} \\ \Gamma \vdash b : P\, 0 \times \pi\, E\, (\lambda_{\#E} x.\, P\, (1{+}x))\end{array}}{\Gamma \vdash \mathsf{switch}\, (\mathsf{cE}\, t\, E)\, P\, b\, 0 \equiv \pi_0\, b : P\, 0}$$

$$\frac{\begin{array}{c}\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash P : \#(\mathsf{cE}\, t\, E) \to \textsc{Set} \\ \Gamma \vdash b : P\, 0 \times \pi\, E\, (\lambda_{\#E} x.\, P\, (1{+}x)) \\ \Gamma \vdash x : \#E\end{array}}{\Gamma \vdash \mathsf{switch}\, (\mathsf{cE}\, t\, E)\, P\, b\, (1{+}x) \equiv \mathsf{switch}\, E\, (\lambda_{\#E} x.\, P\, (1{+}x))\, (\pi_1\, b)\, x : P(1{+}x)}$$

3 Descriptions and their interpretation

3.1 Descriptions

Desc-formation

Again, this is Tarski-style universe construction.

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Desc} : \textsc{Set}} \qquad \frac{\Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \textsc{Set}}{\Gamma \vdash [\![D]\!]\, X : \textsc{Set}}$$

Desc-introduction

We do need any rules introducing Desc-codes as they will be generated by levitation (see below). The resulting constructors are listed on the left-hand side together with suitable abbreviations, which are for our convenience only and not part of the type theory.

$$\underline{\text{`}1} := \mathsf{con}\,[\text{`}1, []] \qquad \frac{\Gamma \vdash X : \textsc{Set}}{\Gamma \vdash [\![\underline{\text{`}1}]\!]\, X \equiv 1 : \textsc{Set}}$$

$$\underline{\text{`}\Sigma}\, S\, D := \mathsf{con}\,[\text{`}\Sigma, [S, [D, []]]] \qquad \frac{\Gamma \vdash X : \textsc{Set} \qquad \Gamma \vdash D : S \to \mathsf{Desc} \qquad \Gamma \vdash S : \textsc{Set}}{\Gamma \vdash [\![\underline{\text{`}\Sigma}\, S\, D]\!]\, X \equiv (s : S) \times [\![D\, s]\!]\, X : \textsc{Set}}$$

$$\underline{\text{`indx}}\, D := \mathsf{con}\,[\text{`indx}, [D, []]] \qquad \frac{\Gamma \vdash X : \textsc{Set} \qquad \Gamma \vdash D : \mathsf{Desc}}{\Gamma \vdash [\![\underline{\text{`indx}}\, D]\!]\, X \equiv X \times [\![D]\!]\, X : \textsc{Set}}$$

$$\underline{\text{`hindx}}\, H\, D := \mathsf{con}\,[\text{`hindx}, [H, [D, []]]] \qquad \frac{\Gamma \vdash X : \textsc{Set} \qquad \Gamma \vdash H : \textsc{Set} \qquad \Gamma \vdash D : \mathsf{Desc}}{\Gamma \vdash [\![\underline{\text{`hindx}}\, H\, D]\!]\, X \equiv (H \to X) \times [\![D]\!]\, X : \textsc{Set}}$$

Desc-levitation

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Desc} \equiv \mu\left(\underline{\text{`}\Sigma}\, \#\underline{\mathsf{E}}\left(\lambda_{\#\underline{\mathsf{E}}} x.\, \mathsf{switchD}\, \underline{\mathsf{E}}\left[\begin{array}{l} \underline{\text{`}1} \\ \underline{\text{`}\Sigma}\, \textsc{Set}\, (\lambda_{\textsc{Set}} S.\, \underline{\text{`hindx}}\, S\, \underline{\text{`}1}) \\ \underline{\text{`indx}}\, \underline{\text{`}1} \\ \underline{\text{`}\Sigma}\, \textsc{Set}\, (\lambda_{\textsc{Set}} \_.\, \underline{\text{`indx}}\, \underline{\text{`}1}) \end{array}\right] x\right)\right) : \textsc{Set}}$$

where $\underline{\mathsf{E}}$ is the enumeration $\mathsf{cE}\,\text{`}1\,(\mathsf{cE}\,\text{`}\Sigma\,(\mathsf{cE}\,\text{`indx}\,(\mathsf{cE}\,\text{`hindx}\,\mathsf{nE})))$.

3.2 Eliminating small products of descriptions

Elimination

$$\frac{\Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \pi\, E\, (\lambda_{\#E} \_.\, \mathsf{Desc}) \qquad \Gamma \vdash x : \#E}{\Gamma \vdash \mathsf{switchD}\, E\, b\, x : \mathsf{Desc}}$$

Computation

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \mathsf{Desc} \times \pi\, E\, (\lambda_{\#E\_}.\, \mathsf{Desc})}{\Gamma \vdash \mathsf{switchD}\, (\mathsf{cE}\, t\, E)\, b\, 0 \equiv \pi_0\, b : P\, 0}$$

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \mathsf{Desc} \times \pi\, E\, (\lambda_{\#E\_}.\, \mathsf{Desc}) \qquad \Gamma \vdash x : \#E}{\Gamma \vdash \mathsf{switchD}\, (\mathsf{cE}\, t\, E)\, b\, (1{+}x) \equiv \mathsf{switchD}\, E\, (\pi_1\, b)\, x : \mathsf{Desc}}$$

3.3 Fixpoint

Formation

$$\frac{\Gamma \vdash D : \mathsf{Desc}}{\Gamma \vdash \mu D : \textsc{Set}}$$

Introduction

$$\frac{\Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash d : [\![D]\!]\, (\mu D)}{\Gamma \vdash \mathsf{con}\, d : \mu D}$$

Elimination

$$\frac{\begin{array}{c}\Gamma \vdash D : \mathsf{Desc} \\ \Gamma \vdash P : \mu D \to \textsc{Set} \\ \Gamma \vdash m : (d : [\![D]\!]\, (\mu D)) \to \mathsf{All}\, D\, (\mu D)\, P\, d \to P\, (\mathsf{con}\, d) \\ \Gamma \vdash x : \mu D\end{array}}{\Gamma \vdash \mathsf{ind}\, D\, P\, m\, x : P\, x}$$

Computation

$$\frac{\begin{array}{c}\Gamma \vdash D : \mathsf{Desc} \\ \Gamma \vdash P : \mu D \to \textsc{Set} \\ \Gamma \vdash m : (d : [\![D]\!]\, (\mu D)) \to \mathsf{All}\, D\, (\mu D)\, P\, d \to P\, (\mathsf{con}\, d) \\ \Gamma \vdash d : [\![D]\!]\, (\mu D)\end{array}}{\Gamma \vdash \mathsf{ind}\, D\, P\, m\, (\mathsf{con}\, d) \equiv m\, d\, (\mathsf{all}\, D\, (\mu D)\, P\, (\mathsf{ind}\, D\, P\, m)\, d) : P\, x}$$

3.4 All predicate

Formation

$$\frac{\Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set} \qquad \Gamma \vdash xs : [\![D]\!]\, X}{\Gamma \vdash \mathsf{All}\, D\, X\, P\, xs : \mathsf{Set}}$$

Introduction

$$\frac{\Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set}}{\Gamma \vdash \mathsf{All}\, \underline{\text{`1}}\, X\, P\, [] \equiv 1 : \mathsf{Set}}$$

$$\frac{\Gamma \vdash S : \mathsf{Set} \qquad \Gamma \vdash D : S \to \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set}}{\Gamma \vdash \mathsf{All}\, (\underline{\text{`}\Sigma}\, S\, D)\, X\, P\, [s, d] \equiv \mathsf{All}\, (D\, s)\, X\, P\, d : \mathsf{Set}}$$

$$\frac{\Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set}}{\Gamma \vdash \mathsf{All}\, (\underline{\text{`indx}}\, D)\, X\, P\, [x, d] \equiv P\, x \times \mathsf{All}\, D\, X\, P\, d : \mathsf{Set}}$$

$$\frac{\Gamma \vdash H : \mathsf{Set} \qquad \Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set}}{\Gamma \vdash \mathsf{All}\, (\underline{\text{`hindx}}\, H\, D)\, X\, P\, [f, d] \equiv ((h : H) \to P\, (f\, h)) \times \mathsf{All}\, D\, X\, P\, d : \mathsf{Set}}$$

$$\frac{\Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set} \qquad \Gamma \vdash p : (x : X) \to P\, x}{\Gamma \vdash \mathsf{all}\, \underline{\text{`1}}\, X\, P\, p\, [] \equiv [] : 1}$$

$$\frac{\Gamma \vdash S : \mathsf{Set} \qquad \Gamma \vdash D : S \to \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set} \qquad \Gamma \vdash p : (x : X) \to P\, x \qquad \Gamma \vdash s : S \qquad \Gamma \vdash d : D\, s}{\Gamma \vdash \mathsf{all}\, (\underline{\text{`}\Sigma}\, S\, D)\, X\, P\, p\, [s, d] \equiv \mathsf{all}\, (D\, s)\, X\, P\, p\, d : \mathsf{All}\, (D\, s)\, X\, P\, d}$$

$$\frac{\Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set} \qquad \Gamma \vdash p : (x : X) \to P\, x \qquad \Gamma \vdash x : X \qquad \Gamma \vdash d : D}{\Gamma \vdash \mathsf{all}\, (\underline{\text{`indx}}\, D)\, X\, P\, p\, [x, d] \equiv [p\, x, \mathsf{all}\, D\, X\, P\, p\, d] : P\, x \times \mathsf{All}\, D\, X\, P\, d}$$

$$\frac{\Gamma \vdash H : \mathsf{Set} \qquad \Gamma \vdash D : \mathsf{Desc} \qquad \Gamma \vdash X : \mathsf{Set} \qquad \Gamma \vdash P : X \to \mathsf{Set} \qquad \Gamma \vdash p : (x : X) \to P\, x \qquad \Gamma \vdash f : H \to X \qquad \Gamma \vdash d : D}{\Gamma \vdash \mathsf{all}\, (\underline{\text{`hindx}}\, H\, D)\, X\, P\, p\, [f, d] \equiv [\lambda_H h.\, p\, (f\, h), \mathsf{all}\, D\, X\, P\, p\, d] : ((h : H) \to P\, (f\, h)) \times \mathsf{All}\, D}$$

References

[1] J. Chapman et al. "The Gentle Art of Levitation". In: *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*. ICFP '10. 2010, pp. 3–14.

[2] P. Martin-Löf. *Intuitionistic Type Theory: Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*. Studies in Proof Theory. Bibliopolis, 1984.

A Small products of descriptions

We spent a lot of effort to define small products over arbitrary types, only to then introduce a special-cased projection function, switchD, restricted to products of descriptions. Below we give an alternative definition of switchD using small products of descriptions from the start.

En-formation

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{En} : \textsc{Set}} \qquad \frac{\Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \#E : \textsc{Set}} \qquad \frac{\Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \pi\mathsf{D}\,E : \textsc{Set}}$$

En-introduction

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{nE} : \mathsf{En}} \qquad \frac{}{\Gamma \vdash \pi\mathsf{D}\,\mathsf{nE} \equiv 1 : \textsc{Set}}$$

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \mathsf{cE}\,t\,E : \mathsf{En}} \qquad \frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En}}{\Gamma \vdash 0 : \#(\mathsf{cE}\,t\,E)} \qquad \frac{\Gamma \vdash t : \mathsf{Tag} \quad \Gamma \vdash E : \mathsf{En}}{\Gamma \vdash \pi\mathsf{D}\,(\mathsf{cE}\,t\,E) \equiv \mathsf{Desc} \times \pi\mathsf{D}\,E : \textsc{Set}}$$

$$\frac{\Gamma \vdash n : \#E}{\Gamma \vdash 1{+}n : \#(\mathsf{cE}\,t\,E)}$$

En-elimination

$$\frac{\Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \pi\mathsf{D}\,E \qquad \Gamma \vdash x : \#E}{\Gamma \vdash \mathsf{switchD}\,E\,b\,x : \mathsf{Desc}}$$

En-computation

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \mathsf{Desc} \times \pi\mathsf{D}\,E}{\Gamma \vdash \mathsf{switchD}\,(\mathsf{cE}\,t\,E)\,b\,0 \equiv \pi_0\,b : P\,0}$$

$$\frac{\Gamma \vdash t : \mathsf{Tag} \qquad \Gamma \vdash E : \mathsf{En} \qquad \Gamma \vdash b : \mathsf{Desc} \times \pi\mathsf{D}\,E \qquad \Gamma \vdash x : \#E}{\Gamma \vdash \mathsf{switchD}\,(\mathsf{cE}\,t\,E)\,b\,(1{+}x) \equiv \mathsf{switchD}\,E\,(\pi_1\,b)\,x : \mathsf{Desc}}$$

B Is En really necessary?

Given a type with exactly four inhabitants, it ought to be possible to eliminate the use of En in the levitation rule defining Desc, using each of the four inhabitants to name one constructor of Desc.

To encode such a type, we add binary sums to the type system. Another approach would be to introduce a Bool type and encode binary sums as $A + B \triangleq (b : \mathsf{Bool}) \times \mathsf{if}\ b\ \mathsf{then}\ A\ \mathsf{else}\ B$. Yet another would be to add a four-element type as a primitive.

B.1 +-types

Formation

$$\frac{\Gamma \vdash S : \textsc{Set} \qquad \Gamma \vdash T : \textsc{Set}}{\Gamma \vdash S + T : \textsc{Set}}$$

Introduction

$$\frac{\Gamma \vdash s : S \qquad \Gamma \vdash T : \textsc{Set}}{\Gamma \vdash \iota_0\, s : S + T} \qquad\qquad \frac{\Gamma \vdash S : \textsc{Set} \qquad \Gamma \vdash t : T}{\Gamma \vdash \iota_1\, s : S + T}$$

Elimination

$$\frac{\Gamma \vdash f : S \to X \qquad \Gamma \vdash g : T \to X \qquad \Gamma \vdash e : S + T}{\Gamma \vdash \langle f, g \rangle\, e : X}$$

Computation

$$\frac{\Gamma \vdash f : S \to X \qquad \Gamma \vdash g : T \to X \qquad \Gamma \vdash s : S}{\Gamma \vdash \langle f, g \rangle\, (\iota_0\, s) \equiv f\, s : X} \qquad \frac{\Gamma \vdash f : S \to X \qquad \Gamma \vdash g : T \to X \qquad \Gamma \vdash t : T}{\Gamma \vdash \langle f, g \rangle\, (\iota_1\, t) \equiv g\, t : X}$$

B.2 A type with exactly four inhabitants

Let $\underline{4}$ be the type give by $1 + (1 + (1 + 1))$. Its only inhabitants are

$$\begin{aligned}
\underline{0} &:= \iota_0\ [] \\
\underline{1} &:= \iota_1\ (\iota_0\ []) \\
\underline{2} &:= \iota_1\ (\iota_1\ (\iota_0\ [])) \\
\underline{3} &:= \iota_1\ (\iota_1\ (\iota_1\ (\iota_0\ [])))
\end{aligned}$$

Furthermore, we can define an eliminator

$$\underline{\mathsf{switch4}}\ a\ b\ c\ d\ x := \langle \lambda_{\_}.\, a, \lambda_{1+1+1} x.\, \langle \lambda_{\_}.\, b, \lambda_{1+1} x.\, \langle \lambda_{\_}.\, c, \lambda_{\_}.\, d \rangle\, x \rangle\, x \rangle\, x$$

that exhibits the desired computational behaviour.

Using this encoding, we can express the levitation rules without use of En:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{Desc} \equiv \mu\big(\underline{\text{`}\Sigma}\ \underline{4}\ \ (\lambda_{\underline{4}} x.\, \underline{\mathsf{switch4}}\ \underline{\text{`}1}\ (\underline{\text{`}\Sigma}\ \textsc{Set}\ (\lambda_{\textsc{Set}} S.\, \underline{\text{`hindx}}\ S\ \underline{\text{`}1}))\ (\underline{\text{`indx}}\ \underline{\text{`}1})\ (\underline{\text{`}\Sigma}\ \textsc{Set}\ (\lambda_{\textsc{Set}\_}.\, \underline{\text{`indx}}\ \underline{\text{`}1}))\ x\big) : \textsc{Set}}$$

where

$$\begin{aligned}
\underline{\text{`}1} &:= \mathsf{con}\ [\underline{0}, []] \\
\underline{\text{`}\Sigma}\ \ S\ D &:= \mathsf{con}\ [\underline{1}, [S, [D, []]]] \\
\underline{\text{`indx}}\ \ D &:= \mathsf{con}\ [\underline{2}, [D, []]] \\
\underline{\text{`hindx}}\ \ H\ D &:= \mathsf{con}\ [\underline{3}, [H, [D, []]]]
\end{aligned}$$

C Can we levitate 1?

Instead of having a built-in unit type, can we introduce it using levitation just like we did with Desc? In other words, can we replace the introduction rule of 1 by a computation rule $\Gamma \vdash 1 \equiv \mu\text{`}\underline{1} : \textsc{Set}$?

Using this alternative rule, let us try to construct its (hopefully) unique inhabitant using the μ-introduction rule

$$\frac{\Gamma \vdash d : [\![\text{`}\underline{1}]\!]\,(\mu\text{`}\underline{1}) \equiv \mu\text{`}\underline{1}}{\Gamma \vdash \mathsf{con}\,d : \mu\text{`}\underline{1}}$$

which tells us that any inhabitant must be of shape $\mathsf{con}\,d$, where d is again of type $\mu\text{`}\underline{1}$. We have hit an infinite regress, meaning that no inhabitant can be constructed.