Homework 5: Leap of Truncation

15-819 Homotopy Type Theory

Out: 19/Nov/13
Due: 4/Dec/13

1 Blah, Blah, Blah (a.k.a. Introduction)

Alright. So, the goal of this assignment is to show that the universe of n-types is itself an $(n+1)$-type. Every task in this assignment is a part of the proof of this theorem. Let us begin right away with the definition of *universes of n-types*:

Definition 1. The *universe of n-types* is $\mathcal{U}^n :\equiv \sum_{A:\mathcal{U}} \texttt{is-}n\texttt{-type}(A)$.[1]

To formally talk about the n, the *truncation level*, let us have a separate data type, `tlevel`, which is isomorphic to `nat` but starts with -2. We write $\texttt{rec}_{\texttt{tlevel}}$ as the recursor, and reuse `suc` as the successor constructor. For example, the $(n+1)$ is really $\texttt{suc}(n)$ in the formal statement, and -1 is really $\texttt{suc}(-2)$. Anyway, here is the theorem:

Theorem 1. $\prod_{n:\texttt{tlevel}} \texttt{is-suc}(n)\texttt{-type}(\mathcal{U}^n)$.

The key insight is that the type of equivalences between two n-types is itself an n-type. By univalence, it means that the type of paths between two n-types is also an n-type. The intuition behind the key insight is that equivalences are maps with special properties, and the structure of parallel maps are, to some degree, limited by that of the codomoin in the presence

[1] $\mathcal{U}_i$ is used for the universe level so I have to find another corner. Sorry.

of functional extensionality. You will fill in the missing details in your solution.

As a side note, this theorem is actually tight for $n \geq -1$, in the sense that $\mathcal{U}^n$ is not a n-type for $n \geq -1$.[2] We will not cover this result in this assignment.

> CLARIFICATION: The bound was $n \geq 0$ but it works for $n = -1$ too.

To make programming easier, you are allowed to use pattern matching for Σ types in λ. The dedicated syntax $\lambda\langle x, y\rangle.M$ means $\lambda p.[\mathtt{fst}(p), \mathtt{snd}(p)/x, y]M$ for some fresh variable $p \notin M$.

2 Cumulativeness

In this section we will show that the truncation hierarchy is cumulative, along with some other useful lemmas. You may assume the following lemma that was proved in class.

$$\mathtt{prop\text{-}is\text{-}set} : \prod_{A:\mathcal{U}} \mathtt{isProp}(A) \to \mathtt{isSet}(A)$$

As usual, you do not have to justify your code.

Task 1. $\mathtt{prop\text{-}level} : \prod_{A:\mathcal{U}} \mathtt{isProp}(A) \to \mathtt{is\text{-}(-1)\text{-}type}(A)$

Solution. $\lambda A \alpha x y.\langle \alpha(x)(y), \mathtt{prop\text{-}is\text{-}set}(A)(\alpha)(x)(y)(\alpha(x)(y))\rangle$

Task 2. $\mathtt{contr\text{-}is\text{-}prop} : \prod_{A:\mathcal{U}} \mathtt{isContr}(A) \to \mathtt{isProp}(A)$.

Solution. $\lambda A \langle a, \alpha\rangle x y.\alpha(x)^{-1} \cdot \alpha(y)$

Task 3. $\mathtt{raise\text{-}level} : \prod_{(A:\mathcal{U})} \prod_{(n:\mathtt{tlevel})} \mathtt{is\text{-}}n\mathtt{\text{-}type}(A) \to \mathtt{is\text{-}suc}(n)\mathtt{\text{-}type}(A)$

[2] It has been known that $\mathcal{U}^{-1}$ is not a (-1)-type (i.e., not a proposition) and $\mathcal{U}^0$ is not a 0-type (i.e. not a set) for a while, and Nicolai Kraus and Christian Sattler generalized this to $n \geq -1$ without using higher inductive types.

Solution.

$$\lambda A n.\mathtt{rec}_{\mathtt{tlevel}}[n.\prod_{A:\mathcal{U}} \mathtt{is}\text{-}n\text{-}\mathtt{type}(A) \to \mathtt{is}\text{-}\mathtt{suc}(n)\text{-}\mathtt{type}(A)]$$
$$(n, \lambda A.\mathtt{prop}\text{-}\mathtt{level}(A) \circ \mathtt{contr}\text{-}\mathtt{is}\text{-}\mathtt{prop}(A), n.f.\lambda A \alpha x y.f(x=y)(\alpha(x)(y)))(A)$$

(Note that to be fully consistent with the lecture notes, $\mathtt{rec}_{\mathtt{tlevel}}$ *should be* $\mathtt{ind}_{\mathtt{level}}$*.)*

Task 4. $\mathtt{prop}\text{-}\mathtt{has}\text{-}\mathtt{level}\text{-}\mathtt{suc} : \prod_{(A:\mathcal{U})} \prod_{(n:\mathtt{tlevel})} \mathtt{isProp}(A) \to \mathtt{is}\text{-}\mathtt{suc}(n)\text{-}\mathtt{type}(A)$

Solution.

$$\lambda A n \alpha.\mathtt{rec}_{\mathtt{tlevel}}[n.\mathtt{is}\text{-}\mathtt{suc}(n)\text{-}\mathtt{type}(A)]$$
$$(n, \mathtt{prop}\text{-}\mathtt{level}(A)(\alpha), n.\beta.\mathtt{raise}\text{-}\mathtt{level}(A)(\mathtt{suc}(n))(\beta))$$

(Note that to be fully consistent with the lecture notes, $\mathtt{rec}_{\mathtt{tlevel}}$ *should be* $\mathtt{ind}_{\mathtt{level}}$*.)*

3 Equivalences

This section is to show the "key insight", i.e., that the type of equivalences between n-types is an n-type.

$$\prod_{(A,B:\mathcal{U})} \prod_{(n:\mathtt{tlevel})} \mathtt{is}\text{-}n\text{-}\mathtt{type}(A) \to \mathtt{is}\text{-}n\text{-}\mathtt{type}(B) \to \mathtt{is}\text{-}n\text{-}\mathtt{type}(A \simeq B)$$

The strategy is to exploit the fact that $\mathsf{isEquiv}(f)$ is a proposition. In general we call $\sum_{x:A} B(x)$ a *subtype* of A when $\prod_{x:A} \mathtt{isProp}(B(x))$. The idea is that the truncation level of a subtype should be bounded by that of its original type. This is true except one special case mentioned below. In the case of equivalences, this means that you can disregard the second component $\mathsf{isEquiv}(f)$ and only focus on the function f. Moreover, we know the truncation level of the function type is bounded by that of the codomain, and hence complete the proof. Unfortunately this strategy can fail when A is contractible, because a subtype of a contractible type might not be contractible. To address, we will handle the case $n = -2$ separately. You may assume these lemmas:

- Some facts derived from the function extensionality. Suppose $f, g : \prod_{x:A} B(x)$ for some $A : \mathcal{U}$ and $B : A \to \mathcal{U}$.

$$\texttt{funext} : \left(\prod_{x:A} f\, x = g\, x \right) \to (f = g)$$

  You do not have to write out implicit arguments A, B, f and g.

- Some operations of $\mathsf{isEquiv}(f)$ so that its implementation becomes irrelevant. Suppose $A, B, C : \mathcal{U}$.

$$\begin{gathered} \texttt{equiv-inv} : A \simeq B \to B \simeq A \\ \texttt{equiv-compose} : (B \simeq C) \to (A \simeq B) \to (A \simeq C) \\ \texttt{is-equiv-is-prop} : \prod_{f:A\to B} \mathsf{isProp}(\mathsf{isEquiv}(f)) \end{gathered}$$

  Note that `equiv-compose` is in the function composition order, which can also be written as $e_1 \circ_\simeq e_2$, mimicking the function composition $f \circ g$. You do not have to write out implicit arguments A, B and C.

- Truncation levels are closed under $\prod$ and $\sum$ formations. Suppose $A : \mathcal{U}$, $B : A \to \mathcal{U}$ and $n : \texttt{tlevel}$.

$$\texttt{product-level} : \left(\prod_{x:A} \texttt{is-}n\texttt{-type}(B(x)) \right) \to \texttt{is-}n\texttt{-type} \left(\prod_{x:A} B(x) \right)$$

$$\begin{aligned} \texttt{sigma-level} : \texttt{is-}n\texttt{-type}(A) \to \left(\prod_{x:A} \texttt{is-}n\texttt{-type}(B(x)) \right) \\ \to \texttt{is-}n\texttt{-type} \left(\sum_{x:A} B(x) \right) \end{aligned}$$

  You do not have to write out implicit arguments A, B and n.

- Finally, you may disregard the second component while handling

paths in subtypes. Suppose $A : \mathcal{U}$ and $B : A \to \mathcal{U}$.

$$\texttt{subtype-path} : \left(\prod_{x:A} \texttt{isProp}(B(x))\right) \to \prod_{m,n:\sum_{x:A} B(x)} (\texttt{fst}(m) = \texttt{fst}(n)) \to (m = n)$$

$$\texttt{subtype-path-equiv} : \left(\prod_{x:A} \texttt{isProp}(B(x))\right) \to \prod_{m,n:\sum_{x:A} B(x)} (\texttt{fst}(m) = \texttt{fst}(n)) \simeq (m = n)$$

Again, you do not have to write out implicit arguments A and B.

3.1 The Special Case

Let us begin with the special case $n = -2$.

Task 5. $\texttt{contr-equiv} : \prod_{A,B:\mathcal{U}} \texttt{isContr}(A) \to \texttt{isContr}(B) \to (A \simeq B)$

> ERRATUM: There was a typo that $B : A \to \mathcal{U}$. It should be $B : \mathcal{U}$.

Solution. $\lambda AB\langle a, \alpha\rangle\langle b, \beta\rangle.\langle \lambda\_.b, \langle \lambda\_.a, \beta\rangle, \langle \lambda\_.a, \alpha\rangle\rangle$

Task 6. *Implement*

$$\texttt{equiv-is-contr} : \prod_{A,B:\mathcal{U}} \texttt{isContr}(A) \to \texttt{isContr}(B) \to \texttt{isContr}(A \simeq B)$$

Warning: this can be mind-blowing. **(Hint)** *Bob has office hours.*

Solution.

$$\lambda AB\alpha\beta.\langle \texttt{contr-equiv}(A)(B)(\alpha)(\beta), \lambda\gamma.\texttt{subtype-path}(\texttt{is-equiv-is-prop}) \\ (\texttt{contr-equiv}(A)(B)(\alpha)(\beta))(\gamma)(\texttt{funext}(\lambda a.\texttt{snd}(\beta)(\texttt{fst}(\gamma)(a)))\rangle$$

3.2 Other Cases

Now let us consider the cases $n \geq -1$. The strategy outlined in the beginning of the section critically depends on the following lemma.

Task 7. *Implement*

$$\texttt{subtype-level} : \prod_{(A:\mathcal{U})} \prod_{(B:A\to\mathcal{U})} \prod_{(n:\texttt{tlevel})} \texttt{is-suc}(n)\texttt{-type}(A) \to \left(\prod_{x:A} \texttt{isProp}(B(x))\right) \to \texttt{is-suc}(n)\texttt{-type}\left(\sum_{x:A} B(x)\right)$$

Solution.

$$\lambda ABn\alpha\beta.\texttt{sigma-level}(\alpha)(\lambda x.\texttt{prop-has-level-suc}(B(x))(n)(\beta(x)))$$

Bonus Task 1. *Find a counterexample of the above task if* $\texttt{is-suc}(n)\texttt{-type}$ *was* $\texttt{is-}n\texttt{-type}$*. That is, find* A *and* B *such that* $\texttt{isContr}(A)$ *but* $\neg\texttt{isContr}(\sum_{x:A} B(x))$*.*

Solution. *Let* $A = \top$ *and* $B = \lambda\_.\bot$*. We have*

$$\langle\langle\rangle, \lambda\_.\texttt{refl}(\langle\rangle)\rangle : \texttt{isContr}(A)$$

and

$$\texttt{snd} \circ \texttt{fst} : \texttt{isContr}\left(\textstyle\sum_{x:A} B(x)\right) \to \bot$$

Finally the theorem of this section.

Task 8. *Implement*

$$\texttt{equiv-level} : \prod_{(A,B:\mathcal{U})} \prod_{(n:\texttt{tlevel})} \texttt{is-}n\texttt{-type}(A) \to \texttt{is-}n\texttt{-type}(B) \to \texttt{is-}n\texttt{-type}(A \simeq B)$$

Solution.

$$\lambda ABn.\texttt{rec}_{\texttt{tlevel}}[n.\texttt{is-}n\texttt{-type}(A) \to \texttt{is-}n\texttt{-type}(B) \to \texttt{is-}n\texttt{-type}(A \simeq B)]$$
$$(n, \texttt{equiv-is-contr}(A)(B), n.\_.\lambda\_\beta.\texttt{subtype-level}(A \to B)(\mathsf{isEquiv})(n)$$
$$(\texttt{product-level}(\lambda\_.\beta))(\texttt{is-equiv-is-prop}))$$

(Note that to be fully consistent with the lecture notes, $\texttt{rec}_{\texttt{tlevel}}$ *should be* $\texttt{ind}_{\texttt{level}}$*.)*

4 Universes

We are reaching the ultimate goal. Feel free to use any lemma or task in previous sections, in addition to these:

- $\texttt{is-}n\texttt{-type}(A)$ is also a proposition.

$$\texttt{is-type-is-prop} : \prod_{(n:\texttt{tlevel})} \prod_{(A:\mathcal{U})} \texttt{isProp}(\texttt{is-}n\texttt{-type}(A))$$

- Some facts derived from the univalence axiom. Suppose $A, B : \mathcal{U}$.

$$\texttt{ua} : A \simeq B \to A =_{\mathcal{U}} B$$
$$\texttt{ua-equiv} : (A \simeq B) \simeq (A =_{\mathcal{U}} B)$$

  As usual, you do not have to write out implicit arguments A and B.

4.1 Real Work

Task 9. *Everything respects equivalences, thanks to the univalence axiom.*

$$\texttt{equiv-preserves-level} : \prod_{(A,B:\mathcal{U})} \prod_{(n:\texttt{tlevel})} (A \simeq B) \to \texttt{is-}n\texttt{-type}(A) \to \texttt{is-}n\texttt{-type}(B)$$

Solution. $\lambda ABne\alpha.\texttt{tr}[x.\texttt{is-}n\texttt{-type}(x)](\texttt{ua}(e))(\alpha)$

Task 10. *Implement*

$$\texttt{ntype-path-equiv} : \prod_{(n:\texttt{tlevel})} \prod_{(A,B:\mathcal{U}^n)} (\texttt{fst}(A) =_{\mathcal{U}} \texttt{fst}(B)) \simeq (A =_{\mathcal{U}^n} B)$$

(Hint) *The lemma name is suggestive. Maybe too suggestive.*

Solution. $\lambda n.\texttt{subtype-path-equiv}(\texttt{is-type-is-prop}(n))$

Task 11. *Show the ultimate theorem and finish this assignment.*

Solution.

$$\lambda n\langle A,\alpha\rangle\langle B,\beta\rangle.\texttt{equiv-preserves-level}(A \simeq B)(\langle A,\alpha\rangle = \langle B,\beta\rangle)(n)$$
$$(\texttt{ntype-path-equiv}(n)(\langle A,\alpha\rangle)(\langle B,\beta\rangle) \circ_{\simeq} \texttt{ua-equiv})$$
$$(\texttt{equiv-level}(A)(B)(n)(\alpha)(\beta))$$