Differences of Types[*]

Subtitle[†]

JOHN LEO[‡], Halfaya Research
FIRST2 LAST2[§], Institution2a and Institution2b

See Introduction.

CCS Concepts: •**Software and its engineering** → **General programming languages;** •**Social and professional topics** → *History of programming languages;*

Additional Key Words and Phrases: Agda, Coq, dependent types, algebra of programming

ACM Reference format:
John Leo and First2 Last2. 2017. Differences of Types. *PACM Progr. Lang.* 1, 1, Article 1 (January 2017), 2 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This document is a place to record my thoughts on the Coq proof repair project headed by Talia Ringer, Nate Yazdani, and Dan Grossman. Everything in it is preliminary.

For now I am using Agda Lite as described in the PhD thesis of Jesper Cockx (https://lirias.kuleuven.be/bitstream/123456789/583556/1/thesis-final-digital.pdf). I am doing this for a number of practical reasons: Agda Lite is much simpler than full Coq, but includes everything we use so far; Jesper is particularly careful and detailed in his presentation; and the thesis contains particular information and results (such as on datatype eliminators and unification) that we might be able to take advantage of.

I also have a selfish reason for using Agda Lite here: I'm also working on another project with Jesper and others to specify "Core Agda", and want to learn the material in the thesis in detail. I'm also personally more familiar and comfortable with Agda over Coq. I am always keeping Coq in mind as well, however, and it should be easy to translate all of this to Coq when we need to.

2 DIFFERENCES OF TYPES

Dependently-typed functional programs and mathematical objects which can be manipulated algebraically. One algebraic operation that can be performed is to take the difference of two types. Given types $A : T$ and $B : T$ of the same sort T this difference $B - A$ can be expressed as a function from A to B:

$$f : A \to B$$

[*] with title note
[†] with subtitle note
[‡] with author1 note
[§] with author2 note

with paper note.
2017. 2475-1421/2017/1-ART1 $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

This is really only interesting if both types are inhabited, and we can also look at the difference between two terms of different types. Given $a : A$ and $b : B$, express $b - a$ as

$$f_{ab} : A \to B$$
$$f(a) = b$$

This function is again not unique in general, and in fact always has the trivial solution $f_{ab}(\_) = b$).

The existence of a should provide some help in calculating b. We would like to capture somehow the notion of using a "maximally", and then defining $b - a$ to be the function that thus takes the minimal amount of effort to go from a to b.

Given that a and b are themselves algebraic structures, it should be possible to quantify the size of a term as either the size or depth of its AST. Denote $|b|$ as the size of b. Our definition of $b - a$ could then be a function $f_{ab} : A \to B$ such that $|f_{ab}(a)|$ is minimal, where we set $|a|$ to be 1.

Note that we do not say what happens when the argument to f_{ab} is not a. We could arbitrarily set the output to b always in this case. More interesting would be to handle any term of type A, using only information about the types A and B. In this case we could define $B - A$ to be any function $f : A \to B$ that minimizes $|f(a)|$ for all $a \in A$, if such a function exists.

Note that this is all very vague and preliminary and might not make much sense. Certainly it seems doubtful $B - A$ would behave as a true difference, so it might be better to use other notation and terminology.

3 EXAMPLES

3.1 Strengthened Conclusions

All eight of the examples in the CSE 503 project, as well as most later examples (such as oldMinimal/newMinimal in email) are of the same form, which could be characterized as strengthening or in general modifying the conclusion of a theorem given a set of hypthoses. The types involved are pi types, and we are given terms $a : (x : X) \to A$ and $b : (x : X) \to B$ where A and B can depend on x. Note that X is identical for both a and b. We view it here as a single sigma type, but in the original form of the it is curried so that a and b have pure pi types.

3.2 Alternate Datatype Formulations