From High School to University Algebra

Thorsten Altenkirch

1 Introduction

Wilkie showed that there are identities in Tarski's Highschool Algebra which are not provable from the laws. At the core of his counterexample is the observation that $AD = BC$ implies

$$(A^y + B^y)^x(C^x + D^x)^y = (A^x + B^x)^y(C^y + D^y)^x$$

but this is unprovable with the laws. Di Cosmo et al observed that this identity corresponds to an isomorphism present in all bicartesian closed categories, i.e. typed λ-calculus with $\to, 1, \times, +$ which cannot be decomposed into the basic isomorphisms of biCCCs.

The situation becomes different, if we move to a setting with dependent types, i.e. we consider locally cartesian closed categories with a disjoint boolean object — this corresponds to Type Theory with $\Pi, \Sigma, 1, 2$. We give a collection of isomorphisms for this Type Theory, which we call *University Algebra* and show that Wilke's counterexample can be derived. We conjecture that this collection is complete for deriving isomorphisms and decidable.

By moving from equations to isomorphisms we move from ordinary algebra to a form of 2-algebra, where we don't only state that an equation hold but also use constants (i.e. isomorphisms) witnessing equations on which further equations may depend.

2 University algebra

University algebra is presented in the language of finitary Type Theory with $\Pi, \Sigma, 1, 2$. The Type Theory is extensional, i.e. all the η-rules are present. We conjecture that equality is decidable (in the presence of type variables), the idea is to extend the technique we have explored for simple types with strong sums (e.g. Altenkirch et al) to dependent types. Adding 0 would make the theory undecidable, because the equations $\mathrm{tt} = \mathrm{ff}$ only holds in inconsistent contexts which can be used to encode consistency of first order intuitionistic logic.

Disjointness of 2 corresponds to the existence of a large if, i.e. if there are types $A, B : \mathbf{Type}$ and $b : 2$ we can form a new type: $\mathrm{if}\, b\, A\, B : \mathbf{Type}$.

University algebra is given by the following basic isomorphisms:

$$
\begin{array}{ll}
\Phi_{2C} : & 2 \simeq 2 \\
\Phi_{2A} : & \Sigma x : 2.\mathrm{if}\, x\, A\, \Sigma y : 2.\mathrm{if}\, y\, B\, C \simeq \Sigma x : 2.\mathrm{if}\, x\, (\Sigma y : 2.\mathrm{if}\, y\, A\, B)\, C \\
\Phi_{\Sigma A} : & \Sigma a : A.\Sigma b : B\, a.C\, a\, b \simeq \Sigma (a,b) : (\Sigma a : A.B\, a).C\, a\, b \\
\Phi_{\Pi 1} : & \Pi - : A.1 \simeq 1 \\
\Phi_{1\Pi} : & \Pi x : 1.B\, x \simeq B\, () \\
\Phi_{2\Pi} : & \Pi b : 2.B\, b \simeq (B\, \mathrm{tt}) \times (B\, \mathrm{ff}) \\
\Phi_{1\Sigma} : & \Sigma x : 1.B\, x \simeq B\, () \\
\Phi_{\Sigma\Pi} : & \Pi a : A.\Pi b : B\, a.C\, a\, b \simeq \Pi (a,b) : (\Sigma a : A.B\, a).C\, a\, b \\
\Phi_{\Pi\Sigma} : & \Pi a : A.\Sigma b : B\, a.C\, a\, b \simeq \Sigma f : (\Pi a : A.B\, a).\Pi a : A.C\, a\, (f\, a)
\end{array}
$$

All the isomorphisms are definable in the language and give rise to definitional isomorphisms, i.e. there is $\phi : A \to B$ and $\phi^{-1} : B \to A$ such that $\phi^{-1} \circ \phi = I_A$ and $\phi \circ \phi^{-1} = I_B$, which can be expanded to $\lambda a : A.\phi^{-1}(\phi\, a) = \lambda a : A.a$ and $\lambda b : B.\phi(\phi^{-1}\, b)$. We sketch this construction by giving the left-to-right morphisms:

$$
\begin{array}{ll}
\Phi_{2C}\, \mathrm{tt} & = \mathrm{ff} \\
\Phi_{2C}\, \mathrm{ff} & = \mathrm{tt} \\
\Phi_{2A}\, (\mathrm{tt}, a) & = (\mathrm{tt}, (\mathrm{tt}, a)) \\
\Phi_{2A}\, (\mathrm{ff}, (\mathrm{tt}, b)) & = (\mathrm{tt}, (\mathrm{ff}, b)) \\
\Phi_{2A}\, (\mathrm{ff}, (\mathrm{ff}, c)) & = (\mathrm{ff}, c) \\
\Phi_{\Sigma A}\, (a, (b, c)) & = ((a, b), c) \\
\Phi_{1\Pi}\, f & = f\, () \\
\Phi_{2\Pi}\, f & = (f\, \mathrm{tt}, f \mathrm{ff}) \\
\Phi_{\Sigma 1}\, ((), b) & = b \\
\Phi_{\Pi\Pi}\, f & = \lambda (a, b).f\, a\, b \\
\Phi_{\Pi\Sigma}\, f & = (\lambda a.\pi_0\, (f a), \lambda a.\pi_1\, (f\, a))
\end{array}
$$

We also introduce structural operators which allow us to derive isomorphisms for complex types from isomorphisms for simpler ones:

$$
\frac{\phi : A \simeq A' \quad \psi : \Pi a : A.B\, a \simeq B'\, (\phi\, a)}{\Psi_\Sigma\, \phi\, \psi : \Sigma a : A.B\, a \simeq \Sigma a' : A'.B'\, a'}
\qquad
\frac{\phi : A' \simeq A \quad \psi : \Pi a : A'.B\, (\phi\, a) \simeq B'\, a}{\Psi_\Pi\, \phi\, \psi : \Pi a : A.B\, a \simeq \Pi a' : A'.B'\, a'}
$$

The rules for Π and Σ show that we have to define isomorphims wrt. a context of assumptions. To complete the list we also have:

$$
\frac{}{I_A : A \simeq A}
\qquad
\frac{b : 2 \quad \phi : A \simeq A' \quad \psi : B \simeq B'}{\Psi_{\mathrm{if}}\, b\, \phi\, \psi : \mathrm{if}\, b\, A\, B \simeq \mathrm{if}\, b\, A'\, B'}
$$

3 Deriving High School Algebra

We represent the operations of High School algebra — there are two isomorphic choices (using 2Π) for $\times$:

$$\begin{aligned} A + B &= \Sigma b : 2.\mathrm{if}\, b\, A\, B \\ A \times B &= \Pi b : 2.\mathrm{if}\, b\, A\, B \\ &\simeq \Sigma - : A.B \\ A \to B &= \Pi - : A.B \end{aligned}$$

The isomorphisms of High School Algebra are:

$$\begin{array}{ll} \Phi_{+C} : & A + B \simeq B + A \\ \Phi_{+A} : & A + (B + C) \simeq (A + B) + C \\ \Phi_{\times 1} : & 1 \times A \simeq A \\ \Phi_{\times A} : & B \times A \simeq B \times A \\ \Phi_{\times +} : & A \times (B + C) \simeq (A \times B) + (A \times C) \\ \Phi_{\to 1} : & A \to 1 \simeq 1 \\ \Phi_{1 \to} : & 1 \to A \simeq A \\ \Phi_{+\to} : & (A + B) \to C \simeq (A \to C) \times (B \to C) \\ \Phi_{\to\times} : & A \to (B \times C) \simeq (A \to B) \times (A \to C) \\ \Phi_{\times\to} : & (A \times B) \to C \simeq A \to (B \to C) \end{array}$$

These isomorphisms are easily derivable from the ones in University Algebra: Φ_{+C} and $\Phi_{\times C}$ follow from Φ_{2C}. $\Phi_{+}A$ can be derived from Φ_{2A}. $\Phi_{\times +}$ follows from $\Phi_{\Sigma A}$:

$$\begin{aligned} A \times (B + C) &\simeq \Sigma_{:} A.\Sigma b : 2.\mathrm{if}\, b\, B\, C \\ &\simeq \Sigma b : 2.\Sigma_{:} A.\mathrm{if}\, b\, B\, C && (\Phi_{\Sigma A}) \\ &= \Sigma b : 2.\mathrm{if}\, b\, (\Sigma - : A.B)\, (\Sigma - : A.C) \\ &\simeq (A \to C) \times (B \to C) \end{aligned}$$

$\Phi_{\to 1}$ is $\Phi_{\Pi 1}$ and $\Phi_{1\to}$ is an instance of $\Phi_{\Pi 1}$. $\Phi_{+\to}$ can be constructed from $\Phi_{\Sigma\Pi}$:

$$\begin{aligned} (A + B) \to C &\simeq \Pi - : (\Sigma b : 2.\mathrm{if}\, b\, A\, B).C \\ &\simeq \Pi b : 2.\Pi - : \mathrm{if}\, b\, A\, B.C && (\Phi_{\Sigma\Pi}) \\ &= \Pi b : 2.\mathrm{if}\, b\, (\Pi - : A.B)\, (\Pi - : A.C) \\ &\simeq (A \to C) \times (B \to C) \end{aligned}$$

$\Phi_{\to\times}$ and $\Phi_{\times\to}$ are instances of $\Phi_{\Pi\Sigma}$ and $\Phi_{\Sigma\Pi}$ respectively.

4 Deriving the Wilkie isomorphism

We can derive a number of useful isomorphisms, essential for the Wilkie isomorphism is the following:

$$\begin{aligned} & X \to (Y \to A + Y \to B) \\ & \equiv X \to \Sigma b : 2.\mathrm{if}\, b\, (Y \to A)\, (Y \to B) \\ & \simeq \Sigma f : X \to 2.\Pi x : X.\mathrm{if}\, (f\, x)\, (Y \to A)\, (Y \to B) && (\Psi_{\Sigma\Pi}) \\ & \equiv \Sigma f : X \to 2.\Pi x : X.Y \to \mathrm{if}\, (f\, x)\, A\, B \end{aligned}$$

Writing both sides of the Wilkie equality as types we can *normalize* both sides so that they only differ in the assumed isomorphism $\phi : A \times D \simeq B \times C$:

$$\begin{aligned}
&(X \to (Y \to A + Y \to B)) \times (Y \to (X \to C + X \to D)\\
&\simeq (\Sigma f : X \to 2.\Pi x : X.Y \to \text{if}\,(f\,x)\,A\,B) \times (\Sigma g : Y \to 2.\Pi y : Y.X \to \text{if}\,(g\,y)\,C\,D)\\
&\simeq \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad (\Pi x : X.Y \to \text{if}\,(f\,x)\,A\,B) \times (X \to \Pi y : Y.\text{if}\,(g\,y)\,C\,D) && (\Sigma\Sigma, \times C))\\
&\simeq \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad (\Pi x : X.\Pi y : Y.(\text{if}\,(f\,x)\,A\,B)) \times (\text{if}\,(g\,y)\,C\,D)) && (\Pi\Sigma)\\
&\equiv \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad \Pi x : X.\Pi y : Y.(\text{if}\,(f\,x)\,(\text{if}\,(g\,y)\,(A \times C)\,(A \times D))\,(\text{if}\,(g\,y)\,(B \times C)\,(B \times D))
\end{aligned}$$

$$\begin{aligned}
&(X \to (Y \to A + Y \to B)) \times (Y \to (X \to C + X \to D)\\
&\simeq (\Sigma g : Y \to 2.\Pi y : Y.X \to \text{if}\,(g\,y)\,A\,B) \times (\Sigma f : X \to 2.\Pi x : X.Y \to \text{if}\,(f\,x)\,C\,D)\\
&\simeq \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad (X \to \Pi y : Y \to \text{if}\,(g\,y)\,A\,B) \times (\Pi x : X.Y \to \text{if}\,(f\,x)\,C\,D) && (\Sigma\Sigma, \times C))\\
&\simeq \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad (X \to \Pi y : Y \to \text{if}\,(g\,y)\,A\,B) \times (\Pi x : X.Y \to \text{if}\,(f\,x)\,C\,D) && (\Pi\Sigma)\\
&\equiv \Sigma f : X \to 2.\Sigma g : Y \to 2.\\
&\qquad \Pi x : X.\Pi y : Y.(\text{if}\,(f\,x)\,(\text{if}\,(g\,y)\,(A \times C)\,(B \times C))\,(\text{if}\,(g\,y)\,(A \times D)\,(B \times D))
\end{aligned}$$

This development raises a number ofquestions:

- Can we use dependent types to present a complete collection of isomorphisms, i.e. that we can derive an isomorphism syntactically if it exists semantically wrt. the category of finite sets.
- To derive the isomorphism it seems to be essential that we have used $\Sigma\Pi$ which corresponds to the axiom of choice. Does this lead to an alternative proof of the incompleteness of High School algebra?
- Can we devise a decision procedure to decide whether isomorphisms exist by normalising types? This would also require to decide definitional equality, but it seems plausible that the existent approaches for deciding equality with strong sums can be extended to dependent types.

5 Normal types

We define types in normal form NF by the following grammar:

$$\begin{aligned}
\text{NF} &:: \Sigma x : \text{NF}_\Pi.\text{NF} \mid \text{NF}_\Pi\\
\text{NF}_\Pi &:: \Pi x : \text{NF}.\text{NF}_\Pi \mid \text{NF}_0\\
\text{NF}_0 &:: X|n|\text{T}[\text{NF}]
\end{aligned}$$

where T$[X]$ stands for non-trivial, redundancy-free case trees whose leaves are in X.

We need to show: Every type is isomorphic to one in normal form. Using this lemma we know that we only have to consider isomorphsims between normal types.

To show completeness we have to show that every isomorphism between normal types is of a particular form. It should be possible to show this directly by analyzing normal forms. A semantic alternative (using sheaves ?) would be preferable.