Generalised Type Setups for Dependently Sorted Logic
TACL 2011

Peter Aczel

The University of Manchester

July 26, 2011
Motivation for the notion of a Generalised Type Setup

Logic-riched dependent type theories

The Problem  The idea of a logic-enrichment of a dependent type theory is to build a logic on top of the type theory by treating its types and typed terms as the sorts and sorted terms of a dependently sorted logic. The idea was first introduced in [Aczel and Gambino (2002)]. In order to make the general idea of logic-enrichment rigorous we need a precise notion to replace the idea of a dependent type theory.
Motivation for the notion of a Generalised Type Setup

Logic-riched dependent type theories

The Problem  The idea of a logic-enrichment of a dependent type theory is to build a logic on top of the type theory by treating its types and typed terms as the sorts and sorted terms of a dependently sorted logic. The idea was first introduced in [Aczel and Gambino (2002)]. In order to make the general idea of logic-enrichment rigorous we need a precise notion to replace the idea of a dependent type theory.

A Solution  The notion of a Generalised Type Setup (GTS) is a precise notion that has abstracted away from the details concerning the inductive generation of the types, terms and contexts of a dependent type theory while keeping an explicit treatment of variable declarations, \( x : A \).
Motivation for the notion of a Generalised Type Setup

Logic-riched dependent type theories

The Problem  The idea of a logic-enrichment of a dependent type theory is to build a logic on top of the type theory by treating its types and typed terms as the sorts and sorted terms of a dependently sorted logic. The idea was first introduced in [Aczel and Gambino (2002)]. In order to make the general idea of logic-enrichment rigorous we need a precise notion to replace the idea of a dependent type theory.

A Solution  The notion of a Generalised Type Setup (GTS) is a precise notion that has abstracted away from the details concerning the inductive generation of the types, terms and contexts of a dependent type theory while keeping an explicit treatment of variable declarations, \( x : A \).

Background  There are a variety of abstract notions of category for dependent type theories that are more concerned with the algebraic semantics of type dependency than the idea of a type theory; e.g. CwFs [Dybjer, 1996].
Some References, 1


P. Aczel and N. Gambino, *Collection Principles in Dependent Type Theory*, *Types for Proofs and Programs* (P. Callaghan et al., editors), LNCS 2277, Springer, (1-23), 2002.


PLAN of TALK

- Generalised Algebraic (GA) Theories (6)
- First Order Logic with Dependent Sorts (FOLDS) (1)
- Generalised Type Setups (GTSs) (3)
- First Order Logic over a GTS (3)
- The references again (2)
Generalised Algebraic (GA) Theories, 1

Example: the GA theory of categories:

**Sorts:** For \(x, y : \text{Obj},\)

- \(\text{Obj}\)
- \(\text{Hom}(x, y)\)

**Terms:** For \(x, y, z : \text{Obj}, f : \text{Hom}(x, y), g : \text{Hom}(y, z),\)

- \(\text{id}(x) : \text{Hom}(x, x)\)
- \(\text{comp}(x, y, z, f, g) : \text{Hom}(x, z)\)

Abbreviations:

\(x \rightarrow y := \text{Hom}(x, y)\)

\(f \circ g := \text{comp}(x, y, z, f, g)\)

Axioms: For \(x, y, z, w : \text{Obj}, f : \text{Hom}(x, y), g : \text{Hom}(y, z), h : \text{Hom}(z, w)\):

- \(\text{id}(x) \circ f = x \rightarrow y f\)
- \(f \circ \text{id}(y) = x \rightarrow y f\)
- \(f \circ (g \circ h) = (f \circ g) \circ h\)
Generalised Algebraic (GA) Theories, 1

Example: the GA theory of categories:

**Sorts:** For $x, y : \text{Obj}$,
- $\text{Obj}$
- $\text{Hom}(x, y)$

**Terms:** For $x, y, z : \text{Obj}, f : \text{Hom}(x, y), g : \text{Hom}(y, z),$
- $\text{id}(x) : \text{Hom}(x, x)$
- $\text{comp}(x, y, z, f, g) : \text{Hom}(x, z)$

**Abbreviations:**
- $x \rightarrow y := \text{Hom}(x, y)$
- $f \bullet g := \text{comp}(x, y, z, f, g)$

**Axioms:** For $x, y, z, w : \text{Obj}, f : x \rightarrow y, g : y \rightarrow z, h : z \rightarrow w$
- $\text{id}(x) \bullet f =_{x \rightarrow y} f$ and $f \bullet \text{id}(y) =_{x \rightarrow y} f$
- $f \bullet (g \bullet h) =_{x \rightarrow w} (f \bullet g) \bullet h$
Generalised Algebraic (GA) Theories, 1

Example: the GA theory of categories:

**Sorts:** For $x, y : Obj$,
- $Obj$
- $Hom(x, y)$

**Terms:** For $x, y, z : Obj, f : Hom(x, y), g : Hom(y, z)$,
- $id(x) : Hom(x, x)$
- $comp(x, y, z, f, g) : Hom(x, z)$

**Abbreviations:**
- $x \rightarrow y := Hom(x, y)$
- $f \bullet g := comp(x, y, z, f, g)$

**Axioms:** For $x, y, z, w : Obj, f : x \rightarrow y, g : y \rightarrow z, h : z \rightarrow w$
- $id(x) \bullet f =_{x\rightarrow y} f$ and $f \bullet id(y) =_{x\rightarrow y} f$
- $f \bullet (g \bullet h) =_{x\rightarrow w} (f \bullet g) \bullet h$

In a GA theory only equations between terms are allowed as formulae. In this GA theory of categories there is no equality between objects, only between arrows.
A pre-signature for a GA theory has sort constructors and term constructors, each of some arity. Certain sort constructors are labelled as equality-forming.
Generalised Algebraic (GA) Theories, 2
Pre-signatures and signatures

- A pre-signature for a GA theory has sort constructors and term constructors, each of some arity. Certain sort constructors are labelled as equality-forming.

- Given a pre-signature, the contexts, $\Gamma$, the $\Gamma$-sorts, the $\Gamma$-terms, and the $\Gamma$-substitutions are simultaneously inductively generated and substitution action on sorts and terms is recursively defined at the same time.
A pre-signature for a GA theory has sort constructors and term constructors, each of some arity. Certain sort constructors are labelled as equality-forming.

Given a pre-signature, the contexts, $\Gamma$, the $\Gamma$-sorts, the $\Gamma$-terms, and the $\Gamma$-substitutions are simultaneously inductively generated and substitution action on sorts and terms is recursively defined at the same time.

A pre-signature is a signature if the arity of each sort constructor has the form $(\Delta)\text{sort}$ and the arity of each term constructor has the form $(\Delta)A$ where $\Delta$ is a context and $A$ is a $\Delta$-sort.
Generalised Algebraic (GA) Theories, 3

- Each context $\Gamma$ will have the form of a list

\[(x_1 : A_1, \ldots, x_n : A_n)\]

of $n \geq 0$ variable declarations of the distinct variables $x_1, \ldots, x_n$ and $A_i$ will be a $\Gamma$-sort for $i = 1, \ldots, n$. 

A variable $x$ is $\Gamma$-free if $x \notin \{x_1, \ldots, x_n\}$.

Each $\Gamma$-substitution $\sigma : \Delta \rightarrow \Gamma$ will have the form of a list

\[[x_1 := a_1, \ldots, x_n := a_n]\]

$\Delta$ of variable assignments where $a_i$ is a $\Delta$-term of sort $A_i \sigma$, for $i = 1, \ldots, n$. 

$\sigma : \Delta \rightarrow \Gamma$ acts on sorts and terms so that $\Gamma$-sort $A \mapsto \Delta$-sort $A \sigma$, $\Gamma$-term $a \mapsto \Delta$-term $a \sigma$. 

P. Aczel (The University of Manchester)  Generalised Type Setups  July 26 8 / 20
Each context $\Gamma$ will have the form of a list

$$(x_1 : A_1, \ldots, x_n : A_n)$$

of $n \geq 0$ variable declarations of the distinct variables $x_1, \ldots, x_n$ and $A_i$ will be a $\Gamma$-sort for $i = 1, \ldots, n$.

A variable $x$ is $\Gamma$-free if $x \not\in \{x_1, \ldots, x_n\}$. 

Each context $\Gamma$ will have the form of a list

$$(x_1 : A_1, \ldots, x_n : A_n)$$

of $n \geq 0$ variable declarations of the distinct variables $x_1, \ldots, x_n$ and $A_i$ will be a $\Gamma$-sort for $i = 1, \ldots, n$.

A variable $x$ is $\Gamma$-free if $x \notin \{x_1, \ldots, x_n\}$.

Each $\Gamma$-substitution $\sigma : \Delta \to \Gamma$ will have the form of a list

$$[x_1 := a_1, \ldots, x_n := a_n]^\Delta$$

of variable assignments where $a_i$ is a $\Delta$-term of sort $A_i \sigma$, for $i = 1, \ldots, n$. 
Generalised Algebraic (GA) Theories, 3

- Each context $\Gamma$ will have the form of a list
  \[(x_1 : A_1, \ldots, x_n : A_n)\]
of $n \geq 0$ variable declarations of the distinct variables $x_1, \ldots, x_n$ and $A_i$ will be a $\Gamma$-sort for $i = 1, \ldots, n$.

- A variable $x$ is $\Gamma$-free if $x \notin \{x_1, \ldots, x_n\}$.

- Each $\Gamma$-substitution $\sigma : \Delta \rightarrow \Gamma$ will have the form of a list
  \[[x_1 := a_1, \ldots, x_n := a_n]^\Delta\]
of variable assignments where $a_i$ is a $\Delta$-term of sort $A_i\sigma$, for $i = 1, \ldots, n$.

- $\sigma : \Delta \rightarrow \Gamma$ acts on sorts and terms so that
  \[
  \Gamma\text{-sort } A \leftrightarrow \Delta\text{-sort } A\sigma \\
  \Gamma\text{-term } a \leftrightarrow \Delta\text{-term } a\sigma
  \]
Contexts:

- ( ) is a context.

Let $\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ be a context.

- If $x$ is $\Gamma$-free and $A$ is a $\Gamma$-sort then
  
  $(\Gamma, x : A) := (x_1 : A_1, \ldots, x_n : A_n, x : A)$ is a context.
Generalised Algebraic (GA) Theories, 4

Contexts and substitutions

Contexts:

- () is a context.

Let $\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ be a context.

- If $x$ is $\Gamma$-free and $A$ is a $\Gamma$-sort then
  $(\Gamma, x : A) := (x_1 : A_1, \ldots, x_n : A_n, x : A)$ is a context.

Substitutions: Let $\Delta \equiv (y_1 : B_1, \ldots, y_m : B_m)$ also be a context.

- $[]^\Delta$ is a substitution $\Delta \rightarrow ()$.

Let $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta$ be a substitution $\Delta \rightarrow \Gamma$.

- If $a$ is a $\Gamma$-term of sort $A$ then
  $[\sigma, x := a]^\Delta \equiv [x_1 := a_1, \ldots, x_n := a_n, x := a]^\Delta$ is a substitution $\Delta \rightarrow (\Gamma, x : A)$. 
Generalised Algebraic (GA) Theories, 5

Sorts, terms and substitution action

Let \( \sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta \) be a substitution \( \Delta \to \Gamma \).

**Sorts:** Let \( F \) be a sort constructor of arity \((\Gamma)\)sort where \( \Gamma \) is a context.
- \( F(a_1, \ldots, a_n) \) is a \( \Delta \)-sort.

**Terms:** \( y_j \) is a \( \Delta \)-term for \( j = 1, \ldots, m \).

Let \( f \) be a term constructor of arity \((\Gamma)A\) where \( \Gamma \) is a context and \( A \) is a \( \Gamma \)-sort.
- \( f(a_1, \ldots, a_n) \) is a \( \Delta \)-term of sort \( A \sigma \).
Generalised Algebraic (GA) Theories, 5

Sorts, terms and substitution action

Let $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta$ be a substitution $\Delta \to \Gamma$.

**Sorts:** Let $F$ be a sort constructor of arity $(\Gamma)\text{sort}$ where $\Gamma$ is a context.

- $F(a_1, \ldots, a_n)$ is a $\Delta$-sort.

**Terms:**

- $y_j$ is a $\Delta$-term for $j = 1, \ldots m$.

Let $f$ be a term constructor of arity $(\Gamma)A$ where $\Gamma$ is a context and $A$ is a $\Gamma$-sort.

- $f(a_1, \ldots, a_n)$ is a $\Delta$-term of sort $A\sigma$. 
Generalised Algebraic (GA) Theories, 5
Sorts, terms and substitution action

Let $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta$ be a substitution $\Delta \to \Gamma$.

**Sorts:** Let $F$ be a sort constructor of arity $(\Gamma)$sort where $\Gamma$ is a context.
- $F(a_1, \ldots, a_n)$ is a $\Delta$-sort.

**Terms:**
- $y_j$ is a $\Delta$-term for $j = 1, \ldots, m$.

Let $f$ be a term constructor of arity $(\Gamma)A$ where $\Gamma$ is a context and $A$ is a $\Gamma$-sort.
- $f(a_1, \ldots, a_n)$ is a $\Delta$-term of sort $A\sigma$.

**Substitution Action:** Let $\tau \equiv [y_1 := b_1, \ldots, y_m := b_m]^\Lambda$ be a substitution $\Lambda \to \Delta$. By structural recursion on sorts and terms define

$$y_j\tau := b_j \quad \text{for } i = 1, \ldots, n$$

$$f(a_1, \ldots, a_n)\tau := f(a_1\tau, \ldots, a_n\tau)$$

$$F(a_1, \ldots, a_n)\tau := F(a_1\tau, \ldots, a_n\tau)$$
Generalised Algebraic (GA) Theories, 6

The category of contexts: Given a GA theory the contexts form a category where the arrows are the substitutions $\Delta \to \Gamma$ and, if $\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ then $id_{\Gamma} := [x_1 := x_1, \ldots, x_n := x_n]^\Gamma$ and, if $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta : \Delta \to \Gamma$ and $\tau : \Lambda \to \Delta$ then

$$\sigma \circ \tau := [x_1 := a_1\tau, \ldots, x_n := a_n\tau]^\Lambda : \Lambda \to \Gamma.$$
Generalised Algebraic (GA) Theories, 6

The category of contexts: Given a GA theory the contexts form a category where the arrows are the substitutions $\Delta \to \Gamma$ and, if $\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ then $id_{\Gamma} := [x_1 := x_1, \ldots, x_n := x_n]^{\Gamma}$ and, if $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^{\Delta} : \Delta \to \Gamma$ and $\tau : \Lambda \to \Delta$ then

$$\sigma \circ \tau := [x_1 := a_1 \tau, \ldots, x_n := a_n \tau]^{\Lambda} : \Lambda \to \Gamma.$$ 

Equations: Let $F$ be an equality-forming sort constructor of arity $(\Gamma)\text{sort}$. If $B \equiv F(a_1, \ldots, a_n)$ is a $\Delta$-sort and $b, b'$ are $\Delta$-terms of sort $B$ then

$$(\Delta) \ b =_B b'$$

is an equation of the GAT.
Generalised Algebraic (GA) Theories, 6

The category of contexts: Given a GA theory the contexts form a category where the arrows are the substitutions $\Delta \to \Gamma$ and, if $\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ then $id_\Gamma := [x_1 := x_1, \ldots, x_n := x_n]_\Gamma$ and, if $\sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta : \Delta \to \Gamma$ and $\tau : \Lambda \to \Delta$ then

$$\sigma \circ \tau := [x_1 := a_1 \tau, \ldots, x_n := a_n \tau]^\Lambda : \Lambda \to \Gamma.$$

Equations: Let $F$ be an equality-forming sort constructor of arity $(\Gamma)$sort. If $B \equiv F(a_1, \ldots, a_n)$ is a $\Delta$-sort and $b, b'$ are $\Delta$-terms of sort $B$ then

$$(\Delta) \; b =_B b'$$

is an equation of the GAT.

A GA theory consists of a GA signature and a set of equations of the signature.
Generalised Algebraic (GA) Theories, 6

The category of contexts: Given a GA theory the contexts form a category
where the arrows are the substitutions $\Delta \rightarrow \Gamma$ and, if
$\Gamma \equiv (x_1 : A_1, \ldots, x_n : A_n)$ then $id_\Gamma := [x_1 := x_1, \ldots, x_n := x_n]^\Gamma$ and, if
$sigma \equiv [x_1 := a_1, \ldots, x_n := a_n]^\Delta : \Delta \rightarrow \Gamma$ and $tau : \Lambda \rightarrow \Delta$ then

$$sigma \circ tau := [x_1 := a_1 tau, \ldots, x_n := a_n tau]^\Lambda : \Lambda \rightarrow \Gamma.$$ 

Equations: Let $F$ be an equality-forming sort constructor of arity $(\Gamma)$sort.
If $B \equiv F(a_1, \ldots, a_n)$ is a $\Delta$-sort and $b, b'$ are $\Delta$-terms of sort $B$ then

$$(\Delta) \ b =_B b'$$

is an equation of the GAT.

A GA theory consists of a GA signature and a set of equations of the
signature.

Inference Rules: Standard rules for equational reasoning are used to
generate the theorems of the GA theory.
First Order Logic with Dependent Sorts (FOLDS)
[Makkai, 1995]

- A $\text{GA}^-$ signature is a GA signature that only has sort constructors. So there are no individual constants or function symbols and the only possible $\Gamma$-terms are the variables declared in the context $\Gamma$. 
• A GA\(^-\) signature is a GA signature that only has sort constructors. So there are no individual constants or function symbols and the only possible \(\Gamma\)-terms are the variables declared in the context \(\Gamma\).

• A FOLDS (FOLDS\(^+\)) signature consists of a GA\(^-\)(GA) signature together with relation symbols, each of arity some context.
A GA$^-\ $signature is a GA signature that only has sort constructors. So there are no individual constants or function symbols and the only possible $\Gamma$-terms are the variables declared in the context $\Gamma$.

A FOLDS (FOLDS$^+$) signature consists of a GA$^-(GA)$ signature together with relation symbols, each of arity some context.

As we will see, for the more general notion of a Generalised Type Setup (GTS) with relation symbols, we can define predicate logic over a FOLDS$^+$ signature and the notion of a FOLDS$^+$ theory.
First Order Logic with Dependent Sorts (FOLDS)

[Makkai, 1995]

• A GA\(^{-}\) signature is a GA signature that only has sort constructors. So there are no individual constants or function symbols and the only possible \(\Gamma\)-terms are the variables declared in the context \(\Gamma\).

• A FOLDS (FOLDS\(^{+}\)) signature consists of a GA\(^{-}\)(GA) signature together with relation symbols, each of arity some context.

• As we will see, for the more general notion of a Generalised Type Setup (GTS) with relation symbols, we can define predicate logic over a FOLDS\(^{+}\) signature and the notion of a FOLDS\(^{+}\) theory.

• A GTS is an abstract notion of dependent type theory which has types, terms and contexts of variable declarations, but has abstracted away from the rules for inductively generating these.
A Category with Types and Terms (CTT) consists of the following.

- A category, \( \mathcal{C} \), of contexts \( \Gamma \) and substitution maps \( \sigma : \Delta \rightarrow \Gamma \).
- An assignment of a set \( \text{Type}(\Gamma) \) of \( \Gamma \)-types to each context \( \Gamma \) and a set \( \text{Term}(\Gamma, A) \) of \( \Gamma \)-terms of type \( A \) to each \( \Gamma \)-type.
A Category with Types and Terms (CTT) consists of the following.

- A category, $\mathcal{C}$, of contexts $\Gamma$ and substitution maps $\sigma : \Delta \to \Gamma$.
- An assignment of a set $\text{Type}(\Gamma)$ of $\Gamma$-types to each context $\Gamma$ and a set $\text{Term}(\Gamma, A)$ of $\Gamma$-terms of type $A$ to each $\Gamma$-type.
- Each substitution $\sigma : \Delta \to \Gamma$ acts contravariantly on types and terms so that if $\sigma : \Delta \to \Gamma$ then
  
  $$\begin{align*}
  A \in \text{Type}(\Gamma) & \mapsto A\sigma \in \text{Type}(\Delta), \\
  a \in \text{Term}(\Gamma, A) & \mapsto a\sigma \in \text{Term}(\Gamma, A).
  \end{align*}$$

such that, for $A \in \text{Type}(\Gamma)$ and $a \in \text{Term}(\Gamma, A),$

- $A \ id_\Gamma = A$ and $a \ id_\Gamma = a$ and
- for $\sigma : \Delta \to \Gamma$, $\tau : \Lambda \to \Delta,$

  $$A(\sigma \circ \tau) = (A\sigma)\tau \text{ and } a(\sigma \circ \tau) = (a\sigma)\tau.$$
Generalised Type Setups (GTSs), 2

A Generalised Type Setup (GTS) consists of a CTT with variables and comprehension extensions.
A Generalised Type Setup (GTS) consists of a CTT with variables and comprehension extensions. The variables form an infinite set of terms such that every context $\Gamma$ has a $\Gamma$-free variable; i.e. a variable that is not a $\Gamma$-term of any $\Gamma$-type.
Generalised Type Setups (GTSs), 2

A **Generalised Type Setup (GTS)** consists of a CTT with variables and comprehension extensions. The **variables** form an infinite set of terms such that every context $\Gamma$ has a $\Gamma$-free variable; i.e. a variable that is not a $\Gamma$-term of any $\Gamma$-type. Associated with each triple $(\Gamma, x, A)$ consisting of a context $\Gamma$, a $\Gamma$-free variable $x$ and a $\Gamma$-type $A$ is a **comprehension extension**; i.e. a substitution $\pi : \Gamma' \to \Gamma$, satisfying the following.

- The variable $x$ is a $\Gamma'$-term of type $A$,
- For each $\Gamma$-type $A$, $A\pi = A \in Type(\Gamma')$ and $a\pi = a \in Term(\Gamma', A)$ for each $\Gamma$-term $a$ of type $A$.
- For each substitution $\sigma : \Delta \to \Gamma$ and each $a \in Term(\Delta, A\sigma)$ there is a unique substitution $\sigma' : \Delta \to \Gamma'$ such that $\pi \circ \sigma' = \sigma$ and $x\sigma' = a$. 
A Generalised Type Setup (GTS) consists of a CTT with variables and comprehension extensions. The variables form an infinite set of terms such that every context $\Gamma$ has a $\Gamma$-free variable; i.e. a variable that is not a $\Gamma$-term of any $\Gamma$-type. Associated with each triple $(\Gamma, x, A)$ consisting of a context $\Gamma$, a $\Gamma$-free variable $x$ and a $\Gamma$-type $A$ is a comprehension extension; i.e. a substitution $\pi : \Gamma' \rightarrow \Gamma$, satisfying the following.

- The variable $x$ is a $\Gamma'$-term of type $A$,
- For each $\Gamma$-type $A$, $A\pi = A \in Type(\Gamma')$ and $a\pi = a \in Term(\Gamma', A)$ for each $\Gamma$-term $a$ of type $A$.
- For each substitution $\sigma : \Delta \rightarrow \Gamma$ and each $a \in Term(\Delta, A\sigma)$ there is a unique substitution $\sigma' : \Delta \rightarrow \Gamma'$ such that $\pi \circ \sigma' = \sigma$ and $x\sigma' = a$.

We write $(\Gamma, x : A)$ for $\Gamma'$ and $[\sigma, x := a]$ for $\sigma'$. 
Type Setups

A Type Setup is a generalised type setup such that the following.

- For each context $\Gamma$, the set $\text{var}(\Gamma)$ of variables that are $\Gamma$-terms is a finite set such that $\text{var}((\Gamma, x : A)) = \text{var}(\Gamma) \cup \{x\}$.
- There is a terminal context $()$ and, for each other context $\Gamma'$ there is a unique triple $(\Gamma, x, A)$ such that $\Gamma'$ is $(\Gamma, x : A)$.
A Type Setup is a generalised type setup such that the following.

- For each context $\Gamma$, the set $\text{var}(\Gamma)$ of variables that are $\Gamma$-terms is a finite set such that $\text{var}((\Gamma, x : A)) = \text{var}(\Gamma) \cup \{x\}$.
- There is a terminal context $(\cdot)$ and, for each other context $\Gamma'$ there is a unique triple $(\Gamma, x, A)$ such that $\Gamma'$ is $(\Gamma, x : A)$.

It follows that in a type setup every context has uniquely the form

$$(\cdots ( (\cdot), x_1 : A_1), \ldots), x_n : A_n)$$

for some $n \geq 0$, naturally abbreviated $(x_1 : A_1, \ldots, x_n : A_n)$, and every substitution $\Delta \rightarrow \Gamma$ has uniquely the form

$$[[\cdots [ []_{\Delta}, x_1 := a_1], \ldots], x_n := a_n]$$

for some $n \geq 0$, naturally abbreviated $[x_1 := a_1, \ldots, x_n := a_n]$, where $[]_{\Delta} : \Delta \rightarrow (\cdot)$.
Assume given a GTS with relations symbols, each of arity some context.
Formulae over a GTS with relation symbols

Assume given a GTS with relations symbols, each of arity some context.

The judgments \((\Gamma) \varphi\), for contexts \(\Gamma\), expressing that \(\varphi\) is a \(\Gamma\)-formula, are inductively generated using the following rules.

- If \(R\) is a relation symbol of arity \(\Lambda\) and \(\tau : \Gamma \rightarrow \Lambda\) then \((\Gamma) R^{\tau}\).
- If \(A\) is an equality \(\Gamma\)-sort and \(a, a'\) are \(\Gamma\)-terms of type \(A\) then \((\Gamma) a =_A a'\).
- If \(\Diamond := \top, \bot\) then \((\Gamma) \Diamond\).
- If \(\Box := \wedge, \vee, \rightarrow\) then \((\Gamma) \varphi_i\), for \(i = 1, 2\), implies \((\Gamma) (\varphi_1 \Box \varphi_2)\).
- If \(\nabla := \forall, \exists\) and \(A\) is a \(\Gamma\)-sort then \((\Gamma, x : A) \varphi_0\) implies \((\Gamma) (\nabla x : A) \varphi_0\).
Formulae over a GTS with relation symbols

Assume given a GTS with relations symbols, each of arity some context.

The judgments \((\Gamma) \phi\), for contexts \(\Gamma\), expressing that \(\phi\) is a \(\Gamma\)-formula, are inductively generated using the following rules.

- If \(R\) is a relation symbol of arity \(\Lambda\) and \(\tau : \Gamma \rightarrow \Lambda\) then \((\Gamma) R^{<\tau>}\).
- If \(A\) is an equality \(\Gamma\)-sort and \(a, a'\) are \(\Gamma\)-terms of type \(A\) then \((\Gamma) a =_A a'\).
- If \(\Diamond := \top, \bot\) then \((\Gamma) \Diamond\).
- If \(\Box := \land, \lor, \rightarrow\) then \((\Gamma) \phi_i\), for \(i = 1, 2\), implies \((\Gamma) (\phi_1 \Box \phi_2)\).
- If \(\nabla := \forall, \exists\) and \(A\) is a \(\Gamma\)-sort then \((\Gamma, x : A) \phi_0\) implies \((\Gamma) (\nabla x : A) \phi_0\).

If \(\tau \equiv [z_1 := c_1, \ldots, z_r := c_r]\) it is natural to write \(R(c_1, \ldots, c_r)\) rather than \(R^{<\tau>}\).
The action of substitutions $\sigma : \Delta \to \Gamma$ on each $\Gamma$-formula $\phi$ to give a $\Delta$-formula $\phi\sigma$ is defined by structural recursion using the following table.

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\phi\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R &lt; \tau &gt;$</td>
<td>$R &lt; \tau \circ \sigma &gt;$</td>
</tr>
<tr>
<td>$(a =_A a')$</td>
<td>$(a\sigma =_{A\sigma} a'\sigma)$</td>
</tr>
<tr>
<td>$\Diamond$</td>
<td>$\Diamond$</td>
</tr>
<tr>
<td>$(\phi_1 \Box \phi_2)$</td>
<td>$(\phi_1\sigma \Box \phi_2\sigma)$</td>
</tr>
<tr>
<td>$(\nabla x : A) \phi_0$</td>
<td>$(\nabla x' : A) \phi_0[\sigma, x := x']$</td>
</tr>
</tbody>
</table>

where $x'$ is $x$ if $x$ is $\Delta$-fresh, but is the first $\Delta$-fresh variable otherwise.
The predicate logic rules of inference for a GTS

- A sequent has the form \((\Gamma) \Phi \Rightarrow \phi\) where \(\Phi\) is a list \(\phi_1, \ldots, \phi_m\) of \(\Gamma\)-formulae and \(\phi\) is a \(\Gamma\)-formula.
The predicate logic rules of inference for a GTS

- A sequent has the form \((\Gamma) \Phi \implies \phi\) where \(\Phi\) is a list \(\phi_1, \ldots, \phi_m\) of \(\Gamma\)-formulae and \(\phi\) is a \(\Gamma\)-formula.
- The predicate logic rules of inference for deriving such sequents are essentially as expected. We just give those for the quantifiers and equality.

\[
\frac{(\Gamma, x : A) \Phi \implies \theta}{(\Gamma) \Phi \implies (\forall x : A)\theta}
\]
\[
\frac{(\Gamma) \Phi \implies \theta[a/x]}{(\Gamma) \Phi \implies (\exists x : A)\theta}
\]
\[
\frac{(\Gamma) \Phi \implies (a =_A a) \quad (\Gamma) \Phi \implies \theta[a'/x]}{(\Gamma) \Phi \implies \theta[a/x] \implies \theta[a'/x]}
\]

where \(\Phi\) is a list of \(\Gamma\)-formulae, \(\phi\) is a \(\Gamma\)-formula, \(\theta\) is a \((\Gamma, x : A)\)-formula, \(a, a'\) are \(\Gamma\)-terms of type \(A\) and \([a/x]\) is the substitution \([id_{\Gamma}, x := a] : \Gamma \rightarrow (\Gamma, x : A)\).


Some References, 2

P. Aczel and N. Gambino, *Collection Principles in Dependent Type Theory*, *Types for Proofs and Programs* (P. Callaghan et al., editors), LNCS 2277, Springer, (1-23), 2002.


