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

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

- J. Cartmell, D. Phil. thesis, Oxford University, 1978.
- J. Cartmell, *Generalised Algebraic theories and Contextual Categories*, APAL 32:209-243, 1986.
- P. Taylor, Ph.D. thesis, Cambridge University, 1986.
- M. Makkai, *First Order Logic with Dependent Sorts, with Applications to Category Theory*, preprint, McGill University, 1995.
- P. Dybjer, *Internal Type Theory*, **Types for Proofs and Programs**, (S. Berardi and M. Coppo, editors), LNCS 1158, Springer, (120-134) 1996.

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.
- N. Gambino and P. Aczel, *The Generalised Type-Theoretic Interpretation of Constructive Set Theory*, JSL 71:67-103, 2006.
- J. Belo, *Dependently Sorted Logic*, **TYPES'07**, (M. Miculan et al., editors) LNCS 4941, Springer, (33-50), 2008.
- J. Belo, Ph.D. thesis, Manchester University, 2009.
- R. Adams and Z. Luo, *Classical predicative logic-enriched type theories*, APAL 161:1315-1345, 2010.

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 $x, y : Obj$,

Obj

$Hom(x, y)$

Terms: For $x, y, z : Obj, f : Hom(x, y), g : Hom(y, z)$,

$id(x) : Hom(x, x)$

$comp(x, y, z, f, g) : Hom(x, z)$

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:

$$\begin{aligned} x \to y &:= Hom(x, y) \\ f \bullet g &:= comp(x, y, z, f, g) \end{aligned}$$

Axioms: For $x, y, z, w : Obj, f : x \to y, g : y \to z, h : z \to w$

$id(x) \bullet f =_{x \to y} f$ and $f \bullet id(y) =_{x \to y} f$

$f \bullet (g \bullet h) =_{x \to w} (f \bullet g) \bullet h$

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:

$$x \to y \quad := Hom(x, y)$$
$$f \bullet g \quad := comp(x, y, z, f, g)$$

Axioms: For $x, y, z, w : Obj, f : x \to y, g : y \to z, h : z \to w$

$id(x) \bullet f =_{x \to y} f$ and $f \bullet id(y) =_{x \to y} f$

$f \bullet (g \bullet h) =_{x \to w} (f \bullet g) \bullet h$

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.

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.

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, Γ, the Γ-sorts, the Γ-terms, and the Γ-substitutions are simultaneously inductively generated and substitution action on sorts and terms is recursively defined at the same time.

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, Γ, the Γ-sorts, the Γ-terms, and the Γ-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 (Δ)sort and the arity of each term constructor has the form $(\Delta)A$ where Δ is a context and A is a Δ-sort.

Generalised Algebraic (GA) Theories, 3

- Each context Γ 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 Γ-sort for $i = 1, \ldots, n$.

Generalised Algebraic (GA) Theories, 3

- Each context Γ 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 Γ-sort for $i = 1, \ldots, n$.

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

Generalised Algebraic (GA) Theories, 3

- Each context Γ 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 Γ-sort for $i = 1, \ldots, n$.

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

- Each Γ-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 Δ-term of sort $A_i\sigma$, for $i = 1, \ldots, n$.

Generalised Algebraic (GA) Theories, 3

- Each context Γ 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 Γ-sort for $i = 1, \ldots, n$.

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

- Each Γ-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 Δ-term of sort $A_i\sigma$, for $i = 1, \ldots, n$.

- $\sigma : \Delta \to \Gamma$ acts on sorts and terms so that

$$\begin{array}{lcl} \Gamma\text{-sort } A & \mapsto & \Delta\text{-sort } A\sigma \\ \Gamma\text{-term } a & \mapsto & \Delta\text{-term } a\sigma \end{array}$$

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 Γ-free and A is a Γ-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 Γ-free and A is a Γ-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 $\Delta \to ()$.

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

- If a is a Γ-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 $\Delta \to (\Gamma, x : A)$.

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 (Γ)sort where Γ is a context.

- $F(a_1, \ldots, a_n)$ is a Δ-sort.

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 (Γ)sort where Γ is a context.

- $F(a_1, \ldots, a_n)$ is a Δ-sort.

Terms:

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

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

- $f(a_1, \ldots, a_n)$ is a Δ-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 (Γ)sort where Γ is a context.

- $F(a_1, \ldots, a_n)$ is a Δ-sort.

Terms:

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

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

- $f(a_1, \ldots, a_n)$ is a Δ-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

$$\begin{array}{rll} y_j\tau & := b_j & \quad \text{for } i = 1, \ldots, n \\ f(a_1, \ldots, a_n)\tau & := f(a_1\tau, \ldots, a_n\tau) & \\ F(a_1, \ldots, a_n)\tau & := F(a_1\tau, \ldots, a_n\tau) & \end{array}$$

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 (Γ)sort. If $B \equiv F(a_1, \ldots, a_n)$ is a Δ-sort and b, b' are Δ-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 (Γ)sort. If $B \equiv F(a_1, \ldots, a_n)$ is a Δ-sort and b, b' are Δ-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 $\Delta \to \Gamma$ 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 (Γ)sort. If $B \equiv F(a_1, \ldots, a_n)$ is a Δ-sort and b, b' are Δ-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 GA^- **signature** is a GA signature that only has sort constructors. So there are no individual constants or function symbols and the only possible Γ-terms are the variables declared in the context Γ.

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 Γ-terms are the variables declared in the context Γ.

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

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 Γ-terms are the variables declared in the context Γ.

• 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 Γ-terms are the variables declared in the context Γ.

• 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.

Generalised Type Setups (GTSs), 1

A Category with Types and Terms (CTT) consists of the following.

- A category, $\mathcal{C}$, of contexts Γ and substitution maps $\sigma : \Delta \to \Gamma$.
- An assignment of a set $\mathit{Type}(\Gamma)$ of Γ-types to each context Γ and a set $\mathit{Term}(\Gamma, A)$ of Γ-terms of type A to each Γ-type.

Generalised Type Setups (GTSs), 1

A Category with Types and Terms (CTT) consists of the following.

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

$$\begin{array}{ll} A \in Type(\Gamma) & \mapsto A\sigma \in Type(\Delta), \\ a \in Term(\Gamma, A) & \mapsto a\sigma \in Term(\Gamma, A). \end{array}$$

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

  - $A\, id_\Gamma = A$ 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.

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 Γ has a Γ-free variable; i.e. a variable that is not a Γ-term of any Γ-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 Γ has a Γ-free variable; i.e. a variable that is not a Γ-term of any Γ-type.
Associated with each triple (Γ, x, A) consisting of a context Γ, a Γ-free variable x and a Γ-type A is a comprehension extension; i.e. a substitution $\pi : \Gamma' \to \Gamma$, satisfying the following.

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

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 Γ has a Γ-free variable; i.e. a variable that is not a Γ-term of any Γ-type.

Associated with each triple (Γ, x, A) consisting of a context Γ, a Γ-free variable x and a Γ-type A is a comprehension extension; i.e. a substitution $\pi : \Gamma' \to \Gamma$, satisfying the following.

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

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

Type Setups

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

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

Type Setups

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

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

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

$$((\cdots(\,(), x_1 : A_1), \ldots), x_n : A_n) \text{ for some } n \geq 0,$$

naturally abbreviated $(x_1 : A_1, \ldots, x_n : A_n)$, and every substitution $\Delta \to \Gamma$ has uniquely the form

$$[[\cdots[\,[]_\Delta, x_1 := a_1], \ldots], x_n := a_n] \text{ for some } n \geq 0,$$

naturally abbreviated $[x_1 := a_1, \ldots, x_n := a_n]$, where $[]_\Delta : \Delta \to ()$.

Formulae over a GTS with relation symbols

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 $(\Gamma)\ \phi$, for contexts Γ, expressing that ϕ is a Γ-**formula**, are inductively generated using the following rules.

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

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 Γ, expressing that ϕ is a Γ-**formula**, are inductively generated using the following rules.

- If R is a relation symbol of arity Λ and $\tau : \Gamma \to \Lambda$ then $(\Gamma)\ R{<}\tau{>}$.
- If A is an equality Γ-sort and a, a' are Γ-terms of type A then $(\Gamma)\ a =_A a'$.
- If $\diamond := \top, \bot$ then $(\Gamma)\ \diamond$.
- If $\Box := \wedge, \vee, \to$ then $(\Gamma)\ \phi_i$, for $i = 1, 2$, implies $(\Gamma)\ (\phi_1 \Box \phi_2)$.
- If $\nabla := \forall, \exists$ and A is a Γ-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 $R{<}\tau{>}$.

Action of substitutions on GTS formulae

The action of substitutions $\sigma : \Delta \to \Gamma$ on each Γ-formula ϕ to give a Δ-formula $\phi\sigma$ is defined by structural recursion using the following table.

ϕ	$\phi\sigma$
$R\!<\!\tau\!>$	$R\!<\!\tau \circ \sigma\!>$
$(a =_A a')$	$(a\sigma =_{A\sigma} a'\sigma)$
$\diamond$	$\diamond$
$(\phi_1 \,\square\, \phi_2)$	$(\phi_1\sigma \,\square\, \phi_2\sigma)$
$(\nabla x : A)\ \phi_0$	$(\nabla x' : A)\ \phi_0[\sigma, x := x']$

where x' is x if x is Δ-fresh, but is the first Δ-fresh variable otherwise.

The predicate logic rules of inference for a GTS

- A sequent has the form $(\Gamma)\ \Phi \Rightarrow \phi$ where Φ is a list $\phi_1, \ldots, \phi_m$ of Γ-formulae and ϕ is a Γ-formula.

The predicate logic rules of inference for a GTS

• A sequent has the form $(\Gamma)\ \Phi \Rightarrow \phi$ where Φ is a list $\phi_1, \ldots, \phi_m$ of Γ-formulae and ϕ is a Γ-formula.

• The predicate logic rules of inference for deriving such sequents are essentially as expected. We just give those for the quantifiers and equality.

$$\frac{(\Gamma, x : A)\ \Phi \Rightarrow \theta}{(\Gamma)\ \Phi \Rightarrow (\forall x : A)\theta} \qquad \frac{(\Gamma)\ \Phi \Rightarrow (\forall x : A)\theta}{(\Gamma)\ \Phi \Rightarrow \theta[a/x]}$$

$$\frac{(\Gamma)\ \Phi \Rightarrow \theta[a/x]}{(\Gamma)\ \Phi \Rightarrow (\exists x : A)\theta} \qquad \frac{\begin{cases} (\Gamma)\ \Phi \Rightarrow (\exists x : A)\theta \\ (\Gamma, x : A)\ \Phi, \theta \Rightarrow \phi \end{cases}}{(\Gamma)\ \Phi \Rightarrow \phi}$$

$$\frac{}{(\Gamma)\ \Phi \Rightarrow (a =_A a)} \qquad \frac{(\Gamma)\ \Phi \Rightarrow (a =_A a')}{(\Gamma)\ \Phi, \theta[a/x] \Rightarrow \theta[a'/x]}$$

where Φ is a list of Γ-formulae, ϕ is a Γ-formula, θ is a $(\Gamma, x : A)$-formula, a, a' are Γ-terms of type A and $[a/x]$ is the substitution $[id_\Gamma, x := a] : \Gamma \to (\Gamma, x : A)$.

Some References, 1

- J. Cartmell, D. Phil. thesis, Oxford University, 1978.
- J. Cartmell, *Generalised Algebraic theories and Contextual Categories*, APAL 32:209-243, 1986.
- P. Taylor, Ph.D. thesis, Cambridge University, 1986.
- M. Makkai, *First Order Logic with Dependent Sorts, with Applications to Category Theory*, preprint, McGill University, 1995.
- P. Dybjer, *Internal Type Theory*, **Types for Proofs and Programs**, (S. Berardi and M. Coppo, editors), LNCS 1158, Springer, (120-134) 1996.

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.
- N. Gambino and P. Aczel, *The Generalised Type-Theoretic Interpretation of Constructive Set Theory*, JSL 71:67-103, 2006.
- J. Belo, *Dependently Sorted Logic*, **TYPES'07**, (M. Miculan et al., editors) LNCS 4941, Springer, (33-50), 2008.
- J. Belo, Ph.D. thesis, Manchester University, 2009.
- R. Adams and Z. Luo, *Classical predicative logic-enriched type theories*, APAL 161:1315-1345, 2010.