

# Manual de Hue

Nicolas Cabral

29 de enero de 2017

## **Resumen**

The abstract text goes here.

## **1. Hue**

Es un asistente de pruebas basados en el Calculo de Construccions  $\lambda C$

## 2. PTS: Pure Type System

**Definición 2.1** (PTS). Un PTS  $\mathcal{P}$  esta definido por  $\{\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}\}$  donde

- $\mathcal{S}$  es un conjunto sorts
- $\mathcal{V}$  es un conjunto de variables
- $\mathcal{A}$  es un conjunto no vacio de  $\mathcal{S} \times \mathcal{S}$  llamados axiomas
- $\mathcal{R}$  es un conjunto de  $\mathcal{S} \times \mathcal{S} \times \mathcal{S}$  llamadas  $\Pi$  – Reglas

**Definición 2.2** (Pseudotérminos). El conjunto  $\mathcal{T}$  de pseudotérminos de un PTS  $\mathcal{P} = \{\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}\}$  es el menor conjunto que satisface lo siguiente:

- $\mathcal{S} \cup \mathcal{V} \subset \mathcal{T}$
- Si  $a \in \mathcal{T}$  y  $b \in \mathcal{T}$  entonces  $ab \in \mathcal{T}$
- Si  $A \in \mathcal{T}$ ,  $B \in \mathcal{T}$  y  $x \in \mathcal{V}$  entonces  $(\lambda x : A.B) \in \mathcal{T}$
- Si  $A \in \mathcal{T}$ ,  $B \in \mathcal{T}$  y  $x \in \mathcal{V}$  entonces  $(\Pi x : A.B) \in \mathcal{T}$

**Definición 2.3** (PTS funcional). Un PTS se dice *funcional* si

- $\langle s_1, s \rangle \in \mathcal{A}$  y  $\langle s_1, s' \rangle \in \mathcal{A}$  implica  $s = s'$ ;
- $\langle s_1, s_2, s \rangle \in \mathcal{R}$  y  $\langle s_1, s_2, s' \rangle \in \mathcal{R}$  implica  $s = s'$ .

**Definición 2.4** (PTS full). Un PTS se dice *full* si para todo  $s_1, s_2 \in \mathcal{S}$  existe un  $s_3 \in \mathcal{S}$  con  $\langle s_1, s_2, s_3 \rangle \in \mathcal{R}$ .

**Definición 2.5** (PTS semifull). Un PTS se dice *semi-full* si para todo  $s_1 \in \mathcal{S}$   $\exists s_2, s_3 [\langle s_1, s_2, s_3 \rangle \in \mathcal{R}] \Rightarrow \forall s_2 \exists s_3 [\langle s_1, s_2, s_3 \rangle \in \mathcal{R}]$

**Lema 2.1.** PTS full  $\Rightarrow$  PTS semi full ??

PTS funcional  $\Rightarrow$  PTS full o semi full??

**Definición 2.6** (Relación  $\vdash$ ). Definimos a la relacion  $\vdash$ , con  $\vdash \subseteq \mathcal{C} \times \mathcal{T} \times \mathcal{T}$  como la menor relación que cumple:

$(Srt)$	$\overline{\emptyset \vdash s_1 : s_2}$	$(s_1, s_2) \in \mathcal{A}$
$(Var)$	$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}$	
$(Wk)$	$\frac{\Gamma \vdash b : B \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash b : B}$	$b \in \mathcal{S} \cup \mathcal{V}$
$(Pi)$	$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A. B : s_3}$	$(s_1, s_2, s_3) \in \mathcal{R}$
$(Lda)$	$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash b : B \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \lambda a : A. b : \Pi x : A. B}$	$(s_1, s_2, s_3) \in \mathcal{R}$
$(App)$	$\frac{\Gamma \vdash a : \Pi x : B. A \quad \Gamma \vdash b : B}{\Gamma \vdash ab : A[x := b]}$	
$(Cnv)$	$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : s}{\Gamma \vdash a : B}$	$A \simeq B$

## 2.1. Ejemplos

Sea  $\mathcal{S} = \{\text{Prop}, \text{Type}\}$  y  $\mathcal{A} = \{(\text{Prop} : \text{Type})\}$  tenemos que

- $\lambda^\rightarrow$  es el PTS con  $(\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}_1)$
- $\lambda\Pi$  es el PTS con  $(\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}_2)$
- $\lambda C$  es el PTS con  $(\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}_3)$

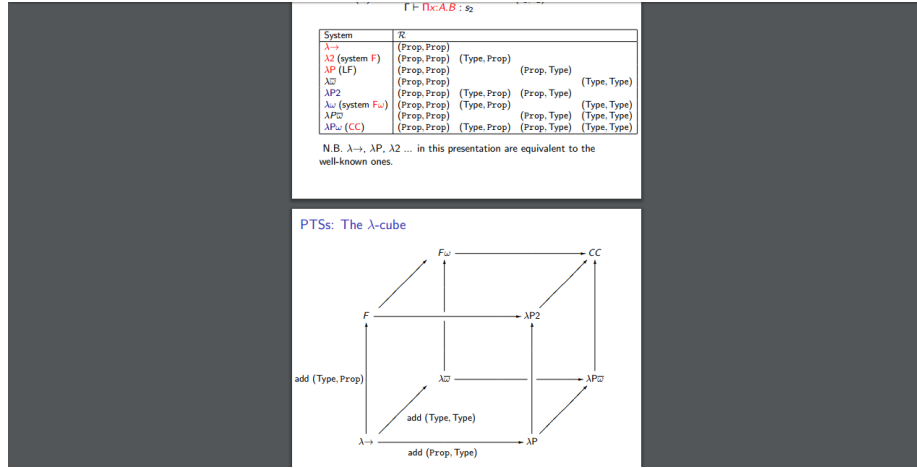
## 2.2. Optimización

Podemos introducir una modificación a  $\vdash$  orientada a la implementación. En lugar de tener que usar  $Wk$  para obtener un contexto válido podemos asumir que el mismo es válido. De esta forma las reglas  $\vdash_{vtyp}$  se definiría de la siguiente manera:

**Definición 2.7** (Relación  $\vdash_{vtyp}$ ). Definimos a la relación  $\vdash_{vtyp}$ , con  $\vdash_{vtyp} \subseteq \mathcal{C} \times \mathcal{T} \times \mathcal{T}$  como la menor relación que cumple:

$$\begin{array}{ll}
(Srt - vtyp) & \frac{}{\Gamma \vdash_{vtyp} s_1 : s_2} \quad (s_1, s_2) \in \mathcal{A} \\
(Var - vtyp) & \frac{}{\Gamma \vdash_{vtyp} x : A} \quad x : A \in \Gamma \\
(Pi - vtyp) & \frac{\Gamma \vdash_{vtyp} A : s_1 \quad \Gamma, x : A \vdash_{vtyp} B : s_2}{\Gamma \vdash_{vtyp} \Pi x : A. B : s_3} \quad (s_1, s_2, s_3) \in \mathcal{R} \\
(Lda - vtyp) & \frac{\Gamma \vdash_{vtyp} A : s_1 \quad \Gamma, x : A \vdash_{vtyp} b : B \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash_{vtyp} \lambda a : A. b : \Pi x : A. B} \quad \begin{array}{l} x \notin dom(\Gamma) \\ (s_1, s_2, s_3) \in \mathcal{R} \end{array} \\
(App - vtyp) & \frac{\Gamma \vdash_{vtyp} a : \Pi x : B. A \quad \Gamma \vdash_{vtyp} b : B}{\Gamma \vdash_{vtyp} ab : A[x := b]} \quad x \notin dom(\Gamma) \\
(Cnv - vtyp) & \frac{\Gamma \vdash_{vtyp} a : A \quad \Gamma \vdash_{vtyp} B : s}{\Gamma \vdash_{vtyp} a : B} \quad A \simeq B
\end{array}$$

#### PASAR A LATEX



**Definición 2.8** (Relación  $\vdash_{vctx}$ ). Definimos a la relacion  $\vdash_{vctx}$ , con el menor predicado sobre  $\mathcal{C}$  que satisface:

$$\begin{array}{ll}
(Nil - vctx) & \frac{}{\emptyset \vdash_{vctx}} \\
(Cons - vctx) & \frac{\Gamma \vdash_{vctx} \quad \Gamma \vdash_{vtyp} A : s}{\Gamma, x : A \vdash_{vctx}}
\end{array}$$

## Some other Pure Type Systems

$\lambda\text{HOL}$	
$S$	$\text{Prop}, \text{Type}, \text{Type}'$
$A$	$\text{Prop} : \text{Type}, \text{Type} : \text{Type}'$
$R$	$(\text{Prop}, \text{Prop}), (\text{Type}, \text{Type}), (\text{Type}, \text{Prop})$
$\lambda U^-$	
$S$	$\text{Prop}, \text{Type}, \text{Type}'$
$A$	$\text{Prop} : \text{Type}, \text{Type} : \text{Type}'$
$R$	$(\text{Prop}, \text{Prop}), (\text{Type}, \text{Type}), (\text{Type}', \text{Type}), (\text{Type}, \text{Prop})$
$\lambda*$	
$S$	$*$
$A$	$* : *$
$R$	$(*, *)$

- ▶  $\lambda\text{HOL}$  corresponds to **constructive Higher Order Logic** under the Curry-Howard isomorphism
- ▶  $\lambda U^-$  is Higher Order Logic over **impredicative domains** and is inconsistent (Girard's paradox)
- ▶  $\lambda*$  is the system with 'Type : Type', which is also inconsistent.

**Lema 2.2.** Equivalencia de  $\vdash$  y  $\vdash_{vtyp}$

$$\Gamma \vdash a : A \Leftrightarrow \Gamma \vdash_{vctx} \wedge \Gamma \vdash_{vtyp} a : A \quad (1)$$

### 3. Teoría del calculo de $\lambda C$

**Definición 3.1.**  $\lambda C$  es un PTS definido por  $\{\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{R}\}$  tales que:

- $\mathcal{S} = \{\text{Prop}, \text{Type}\}$
- $\mathcal{V} = \dots$
- $\mathcal{A} = \{(\text{Prop}, \text{Type})\}$
- $\mathcal{R} = \{(\text{Prop}, \text{Prop}, \text{Prop}), (\text{Prop}, \text{Type}, \text{Type}), (\text{Type}, \text{Prop}, \text{Prop}), (\text{Type}, \text{Type}, \text{Type})\}$

**Lema 3.1.**  $\lambda C$  es un PTS *full*.

*Demostración.*

□

## 4. Hue: Typechecker

**Definición 4.1.** Los terminos del  $\lambda$ -calculo esta definidos por:

```
data CoCT =  IdT Int
            | VarT String
            | LamT CoCT CoCT
            | AppT CoCT CoCT
            | PiT CoCT CoCT
            | PropT
            | TypeT deriving (Eq)
```

```
type Type = CoCT
type Term = CoCT
```

**Definición 4.2.** Los contextos en Hue estan definidos de la siguiente manera:

```
type Context = [ (String, (Maybe Term, Type)) ]
```

**Definición 4.3.** Definimos *whnf* como el operador que reduce un termino a su forma normal a la cabeza.

```
whnf :: Context -> CoCT -> NTerm
whnf c (IdT n)      = NAtom (AId n)
whnf c (VarT s)     = maybe (NAtom (AVar s)) (whnf c) (getDef c s)
whnf c (PiT s t)    = NAtom (APi s t)
whnf c PropT        = NAtom (AProp)
whnf c TypeT        = NAtom (AType)
whnf c (LamT s t)   = NLam s t
whnf c (AppT m n)   = case whnf c m of
    NLam _ t -> whnf c (app t n)
    NAtom t  -> NAtom (AApp t n)
```

**Definición 4.4.** Definimos la relacion  $A \simeq_{\delta\beta} B$  de la siguiente manera

```
conv :: Context -> CoCT -> CoCT -> Bool
conv c s t = convT c (whnf c s) (whnf c t)
```

**Definición 4.5.** Definimos el operador  $\rightarrow^* \Pi x : A.B$  de la siguiente manera

```
getPi :: Context -> CoCT -> Maybe (CoCT, CoCT)
getPi c t = case whnf c t of
    NAtom (APi m n) -> Just (m, n)
    _                -> Nothing
```

Definimos el operador  $\rightarrow^* s$  donde  $s \in \mathcal{S}$  de la siguiente manera

```

getSort :: Context -> CoCT -> Maybe CoCT
getSort c t = case whnf c t of
    NAtom AProp -> Just PropT
    NAtom AType -> Just TypeT
    _             -> Nothing

```

**Definición 4.6** (Relación  $\vdash_{hue}$ ). Definimos a la relacion  $\vdash_{hue}$ , con  $\vdash_{hue} \subseteq \mathcal{C} \times \mathcal{T} \times \mathcal{T}$  como la menor relación que cumple:

$$\begin{array}{lll}
(Srt - hue) & \overline{\Gamma \vdash_{hue} s_1 : s_2} & (s_1, s_2) \in \mathcal{A} \\
(Var - hue) & \overline{\Gamma \vdash_{hue} x : A} & x : A \in \Gamma \\
(Pi - hue) & \frac{\Gamma \vdash_{hue} A : \rightarrow^* s_1 \quad \Gamma, x : A \vdash_{hue} B : \rightarrow^* s_2}{\Gamma \vdash_{hue} \Pi x : A. B : s_3} & (s_1, s_2, s_3) \in \mathcal{R} \\
(Lda - hue) & \frac{\Gamma \vdash_{vtyp} A : \rightarrow^* s_1 \quad \Gamma, x : A \vdash_{vtyp} b : B}{\Gamma \vdash_{vtyp} \lambda a : A. b : \Pi x : A. B} & x \notin dom(\Gamma) \\
(App - hue) & \frac{\Gamma \vdash_{hue} a : \rightarrow^* \Pi x : B. A \quad \Gamma \vdash_{hue} b : B'}{\Gamma \vdash_{hue} ab : A[x := b]} & B \notin \mathcal{S} \vee \mathcal{A}(B) \\
& & B \simeq B'
\end{array}$$

**Lema 4.1.** Equivalencia de  $\vdash$  y  $\vdash_{hue}$  Sea  $\vdash$  la relación del PTS asociado a  $\lambda C$  entonces:

$$\Gamma \vdash a : A \Leftrightarrow \Gamma \vdash_{vctx} \wedge \Gamma \vdash_{hue} a : A \quad (2)$$

*Demostración.* chan! □



**4.1. Terminos**  
**4.2. Reduccion**  
**4.3. Contextos**  
**4.4. Typechecker**

[3] [1] [2]

$$\frac{}{\vdash s_1 : s_2} \text{Srtnsd} \quad (3)$$

$$\frac{\Gamma \vdash A : \Rightarrow s}{\Gamma, x : A \vdash_{nsd} x : A} \text{Varnsd} \quad (4)$$

$$\frac{\Gamma \vdash_{nsd} b : B \quad \Gamma \vdash A : \Rightarrow s}{\Gamma, x : A \vdash_{nsd} b : B} \text{Wknsd} \quad (5)$$

$$\frac{\Gamma \vdash A : \Rightarrow s_1 \quad \Gamma, x : A \vdash_{nsd} B : \Rightarrow s_2}{\Gamma \vdash_{nsd} \Pi x : A. B : s_3} \text{Pinsd} \quad (6)$$

$$\frac{\Gamma \vdash A : \Rightarrow s_1 \quad \Gamma, x : A \vdash_{nsd} b : \Rightarrow B \quad \Gamma, x : A \vdash_{nsd} B : \Rightarrow s_2}{\Gamma \vdash_{nsd} \lambda x : A. b : \Pi x : A. B} \text{Ldansd} \quad (7)$$

$$\frac{\Gamma \vdash a : \Rightarrow \Pi x : A. B \quad \Gamma \vdash_{nsd} b : \Rightarrow B}{\Gamma \vdash_{nsd} a \ b : A[x := b]} \text{Appnsd} \quad (8)$$

## 5. Conclusion

Write your conclusion here.

## Referencias

- [1] L. S. van Benthem Jutting, J. McKinna, and R. Pollack. Checking algorithms for pure type systems. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programs*, pages 19–61. Springer, Berlin, Heidelberg, 1993.
- [2] L. S. van Benthem Jutting, James McKinna, and Robert Pollack. Checking algorithms for pure type systems. In Henk Barendregt and Tobias Nipkow, editors, *Types for Proofs and Programs, International Workshop TYPES'93, Nijmegen, The Netherlands, May 24-28, 1993, Selected Papers*, volume 806 of *Lecture Notes in Computer Science*, pages 19–61. Springer, 1993.
- [3] L.S. van Benthem Jutting, J. McKinna, and R. Pollack. Checking algorithms for pure type systems, 1993.