Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Type Reconstruction

Sven Woop

woop@ps.uni-sb.de

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Goal

- Calculating a principle type of a not type-annotated term. More Formally: Given a pair (Γ,t), compute the most general type T such that Γ > t : T es well typed.
- Example: $\phi \succ f = \lambda x. x(f\ x) \Rightarrow \quad f : (X \to X) \to X$

  $\phi \succ \lambda x. x : X \to X$
- 2 Steps
  - Derive a set of contraints
  - find the principal unifier for these constraints
- We compute principal types, not principal typings.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Index

- Basics
  - Standard Unification
  - Nonstandard Unification
  - Typing-rules for simply typed λ-calculus
- Type Reconstruction
  - Constraint typing rules for λ-calculus
  - CT-Rules and Recursive types
- Polymorphism
  - Let-Polymorphism
- Overview

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Unification

- Unification, [Robinson, 1965]
- Unification in linear space complexity [Martelli, Montanary, 1984]

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Standard Unification

- More precisely: syntactic equational unification
- We define the set of terms as:
  $s,t := x \mid f(t_1,\ldots,t_n)$ with $x \in Var$, $f \in FuncSymbols$
- Given an equation
  $s \approx t$
  we search a substitution σ such that
  $\sigma s = \sigma t$
- σ is called a *unifier* for $s \approx t$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Standard Unification

- We call a unifier σ_1 *more general* than a unifier σ_2 iff there is a substitution σ such that $\sigma \sigma_1 = \sigma_2$.
  We write $\sigma_1 \leq \sigma_2$.

- A *principal unifier* of $s \approx t$ is a unifier σ such that for all unifiers σ‘ of $s \approx t$ we have $\sigma \leq \sigma$‘.

Unification Theorem: Each equation $s \approx t$ has a principal unifier if it is unifiable.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example

- $f(x,y) \approx f(a,y)$
- $\sigma_1 = \{ x := a, y := b \}$ is a unifier as
  $\sigma_1 f(x,y) = \sigma_1 f(a,y)$
  $f(a,b) = f(a,b)$
- $\sigma_2 = \{ x := a \}$ is a principal unifier
  $\sigma_2 f(x,y) = \sigma_2 f(a,y)$
  $f(a,y) = f(a,y)$
- $\{ y := b \} \sigma_2 = \sigma_1$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example

- $f(x) \approx g(a)$ is not unifiable
- $x \approx f(x)$ is not unifiable by *standard unification* !!!!

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Unification by Martelli/Montanari

$t \approx t, R \mid \sigma \ \Rightarrow_{MM} R \mid \sigma$

$f(\ldots) \approx g(\ldots), R \mid \sigma \ \Rightarrow_{MM} \ \perp$ if $f \neq g$ or $\mathrm{Arity}(f) \neq \mathrm{Arity}(g)$

$f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n), R \mid \sigma \ \Rightarrow_{MM} s_1 \approx t_1, \ldots , s_n \approx t_n, R \mid \sigma$

$x \approx t, R \mid \sigma \ \Rightarrow_{MM} [x{:=}\ t]\ R \mid [x{:=}t]\ \sigma$ if $x \notin \mathrm{var}(t)$

(Self Occurence Check)

$x \approx t, R \mid \sigma \ \Rightarrow_{MM} \ \perp$ if $x \in \mathrm{var}(t)$

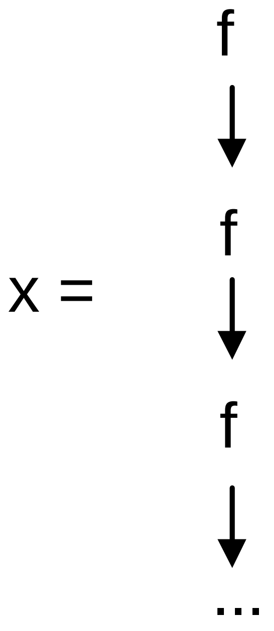
$t \approx x, R \mid \sigma \ \Rightarrow_{MM} x \approx t, R \mid \ \sigma$

$\phi \mid \sigma \Rightarrow_{MM} \sigma$

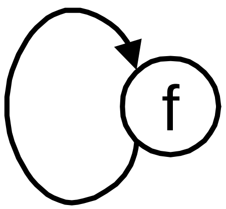
Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Motivation

$x \approx f(x)$ is not unifiable with a **finite** term. But the following regular tree is an **infinite** solution:



finite representation:



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Equivalence Test

```
s := ϕ

fun eq(n,m) =
   if {n,m} ∈ s then
      true
   else if Label(n) ≠ Label(m) or Arity(n) ≠ Arity(m)
      false
   else
      s := s ∪ { {n,m} }
      $\bigwedge_{i=1}^{Arity(n)} eq(n.i,m.i)$
```

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Nonstandard Unification

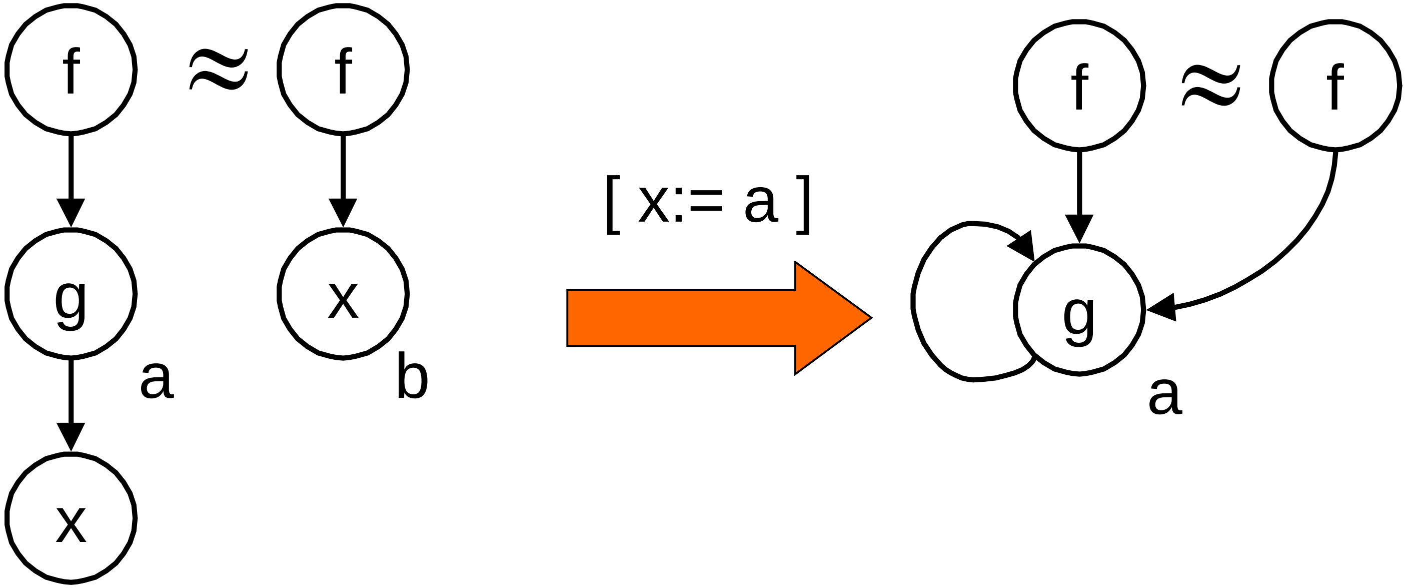
See the unification problem $t_1 \approx t_2$ as a graph unification problem. Let *eq* be a function that computes the equivalence between two nodes in a graph.

```
While not eq(t1,t2) do
   let (n,m) be a pair of nodes with Label(n) ≠ Label(m)
         or Arity(n) ≠ Arity(m)
   if Label(n) = f and Label(m) = g and f ≠ g or
         Arity(n) ≠ Arity(m) then return ⊥
   else if Label(n) = x then subst(x,m)
   else if Label(m) = x then subst(x,n)
```

Note: No occurence check !!!!!!
Solutions are infinite regular trees.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example: f(g(x)) ≈ f(x)



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Typing rules for simply typed lambda calculus

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Typing Rules

$$\frac{x : T \in \Gamma}{\Gamma \succ x : T} \ (\text{Ty-Var})$$

$$\frac{\Gamma, x_1 : T_1 \succ t_2 : T_2}{\Gamma \succ x_1 : T_1 = t_2 : T_2} \ (\text{Ty-Rec})$$

$$\frac{\Gamma, x : T_1 \succ t_2 : T_2}{\Gamma \succ \lambda x : T_1 . t_2 : T_1 \to T_2} \ (\text{Ty-Abs})$$

Note: Abstractions and recursions are type annotated!!!!

$$\frac{\Gamma \succ t_1 : T_2 \to \mathrm{T}_3 \quad \Gamma \succ t_2 : T_2}{\Gamma \succ t_1\ t_2 : T_3} \ (\text{Ty-App})$$

$$\frac{\Gamma \succ t_1 : Bool \quad \Gamma \succ t_2 : T \quad \Gamma \succ t_3 : T}{\Gamma \succ \text{if } t_1 \text{ then } t_2 \text{ else } \mathrm{t}_3 : T} \ (\text{Ty-If})$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example λ x. λ y. x y

λ x: Bool → Bool. λ y: Bool. x y : (Bool → Bool) → Bool → Bool

λ x: Nat → Bool. λ y: Nat. x y : (Nat → Bool) → Nat → Bool

λ x: Bool → Y. λ y: Bool. x y : (Bool → Y) → Bool → Y

λ x: X → Y. λ y: X. x y : (X → Y) → X → Y

Supposition: It exists a principal type annotation.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Constraint typing rules

Principal Types, Curry and Feys [1958]

Algorithm to compute principal types, Hindley [1969]

Type reconstruction, Algorithm W, Damas and Milner [1982]

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Goal

- Calculating a principle type of a not type-annotated term. More Formally: Given a pair (Γ,t), compute the most general type T such that Γ > t : T es well typed.

- Example: $\phi \succ f = \lambda x.\, x\,(f\ x) \Rightarrow\ f : (X \to X) \to X$

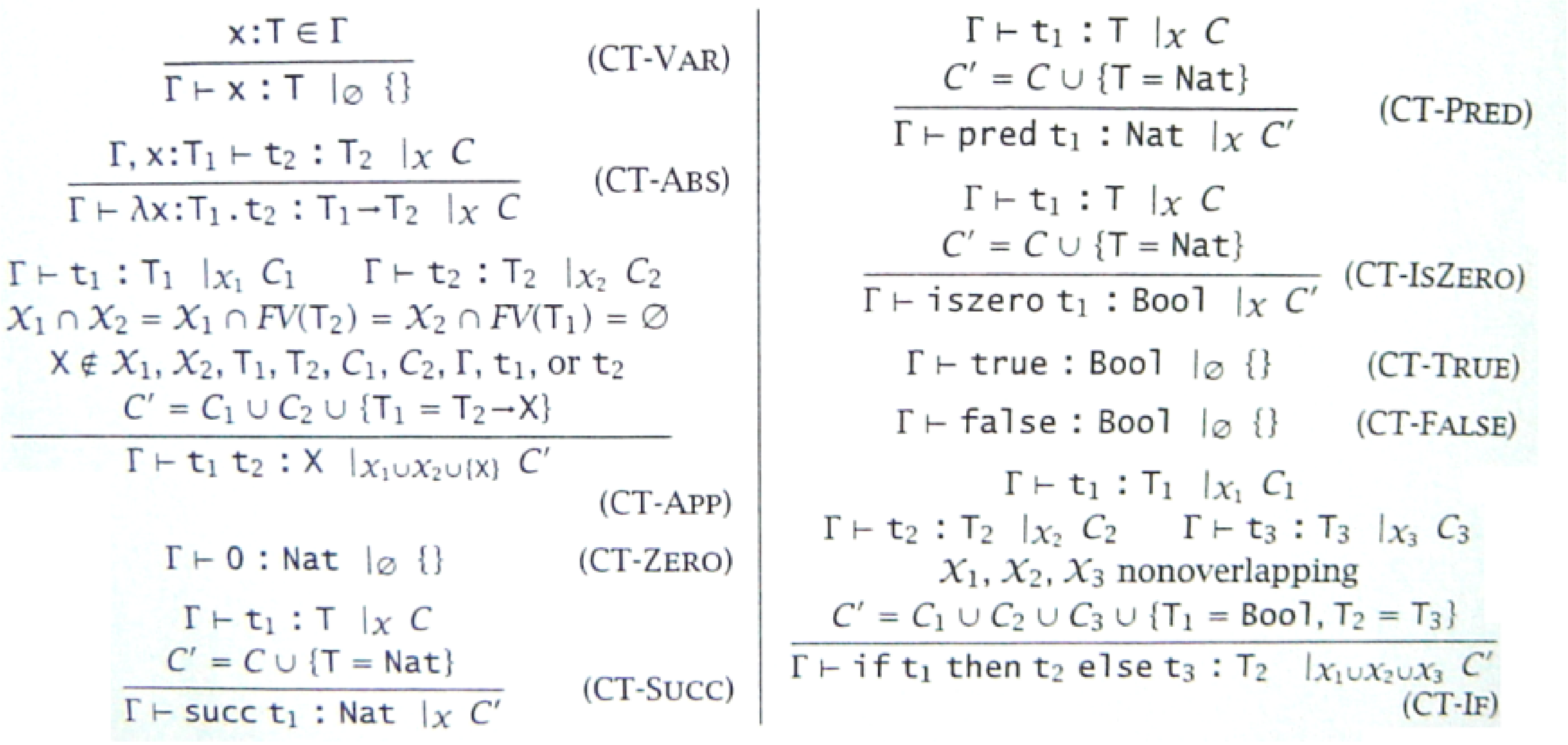
  $\phi \succ \lambda x.\, x : X \to X$

- 2 Steps
  - Derive a set of contraints
  - find the principal unifier for these constraints

- We compute principal types, not principal typings.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

CT-Rules by Pierce



$$\frac{x{:}\mathsf{T} \in \Gamma}{\Gamma \vdash x : \mathsf{T} \mid_{\emptyset} \{\}} \qquad \text{(CT-VAR)}$$

$$\frac{\Gamma, x{:}\mathsf{T}_1 \vdash t_2 : \mathsf{T}_2 \mid_{\mathcal{X}} C}{\Gamma \vdash \lambda x{:}\mathsf{T}_1.t_2 : \mathsf{T}_1 \to \mathsf{T}_2 \mid_{\mathcal{X}} C} \qquad \text{(CT-ABS)}$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : \mathsf{T}_1 \mid_{\mathcal{X}_1} C_1 \qquad \Gamma \vdash t_2 : \mathsf{T}_2 \mid_{\mathcal{X}_2} C_2 \\ \mathcal{X}_1 \cap \mathcal{X}_2 = \mathcal{X}_1 \cap FV(\mathsf{T}_2) = \mathcal{X}_2 \cap FV(\mathsf{T}_1) = \emptyset \\ \mathsf{X} \notin \mathcal{X}_1, \mathcal{X}_2, \mathsf{T}_1, \mathsf{T}_2, C_1, C_2, \Gamma, t_1, \text{ or } t_2 \\ C' = C_1 \cup C_2 \cup \{\mathsf{T}_1 = \mathsf{T}_2 \to \mathsf{X}\} \end{array}}{\Gamma \vdash t_1\ t_2 : \mathsf{X} \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{\mathsf{X}\}} C'} \qquad \text{(CT-APP)}$$

$$\Gamma \vdash 0 : \mathsf{Nat} \mid_{\emptyset} \{\} \qquad \text{(CT-ZERO)}$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : \mathsf{T} \mid_{\mathcal{X}} C \\ C' = C \cup \{\mathsf{T} = \mathsf{Nat}\} \end{array}}{\Gamma \vdash \mathsf{succ}\ t_1 : \mathsf{Nat} \mid_{\mathcal{X}} C'} \qquad \text{(CT-SUCC)}$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : \mathsf{T} \mid_{\mathcal{X}} C \\ C' = C \cup \{\mathsf{T} = \mathsf{Nat}\} \end{array}}{\Gamma \vdash \mathsf{pred}\ t_1 : \mathsf{Nat} \mid_{\mathcal{X}} C'} \qquad \text{(CT-PRED)}$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : \mathsf{T} \mid_{\mathcal{X}} C \\ C' = C \cup \{\mathsf{T} = \mathsf{Nat}\} \end{array}}{\Gamma \vdash \mathsf{iszero}\ t_1 : \mathsf{Bool} \mid_{\mathcal{X}} C'} \qquad \text{(CT-ISZERO)}$$

$$\Gamma \vdash \mathsf{true} : \mathsf{Bool} \mid_{\emptyset} \{\} \qquad \text{(CT-TRUE)}$$

$$\Gamma \vdash \mathsf{false} : \mathsf{Bool} \mid_{\emptyset} \{\} \qquad \text{(CT-FALSE)}$$

$$\frac{\begin{array}{c} \Gamma \vdash t_1 : \mathsf{T}_1 \mid_{\mathcal{X}_1} C_1 \\ \Gamma \vdash t_2 : \mathsf{T}_2 \mid_{\mathcal{X}_2} C_2 \qquad \Gamma \vdash t_3 : \mathsf{T}_3 \mid_{\mathcal{X}_3} C_3 \\ \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3 \text{ nonoverlapping} \\ C' = C_1 \cup C_2 \cup C_3 \cup \{\mathsf{T}_1 = \mathsf{Bool}, \mathsf{T}_2 = \mathsf{T}_3\} \end{array}}{\Gamma \vdash \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 : \mathsf{T}_2 \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} C'} \qquad \text{(CT-IF)}$$

Figure 22-1: Constraint typing rules

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Constraint typing rules

Let all X_i be fresh type variables.

$$\frac{x : T \in \Gamma}{\Gamma \succ x : T \mid \{\}} \ \text{(CT - Var)} \qquad \frac{\Gamma, x_1 : X_1 \succ t_2 : T_2 \mid C}{\Gamma \succ x_1 : X_1 = t_2 : T_2 \mid C \cup \{X_1 = T_2\}} \ \text{(CT - Rec)}$$

$$\frac{\Gamma, x : X_1 \succ t_2 : T_2 \mid C}{\Gamma \succ \lambda x : X_1 . t_2 : X_1 \to T_2 \mid C} \ \text{(CT - Abs)}$$

$$\frac{\Gamma \succ t_1 : T_1 \mid C_1 \quad \Gamma \succ t_2 : T_2 \mid C_2}{\Gamma \succ t_1 \; t_2 : X \mid C_1 \cup C_2 \cup \{T_1 = T_2 \to X\}} \ \text{(CT - App)}$$

$$\frac{\Gamma \succ t_1 : T_1 \mid C_1 \quad \Gamma \succ t_2 : T_2 \mid C_2 \quad \Gamma \succ t_3 : T_3 \mid C_3}{\Gamma \succ \text{if } t_1 \text{ then } t_2 \text{ else } \mathrm{t}_3 : T_2 \mid C_1 \cup C_2 \cup C_3 \cup \{T_1 = Bool, T_2 = T_3\}} \ \text{(CT - If)}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Idea

- Do just the same as the standard typing rules.
- Introduce fresh type variables each time a type can‘t be computed directly.
- Construct constraints consisting of the conditions the typing rules check.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

CT-Var

$$\frac{x : T \in \Gamma}{\Gamma \succ x : T}$$

Ty-Var

$$\frac{x : T \in \Gamma}{\Gamma \succ x : T \,|\, \{\}}$$

CT-Var

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

CT-Rec

$$\frac{\Gamma, x_1 : T_1 \succ t_2 : T_2 \qquad T_1 = T_2}{\Gamma \succ x_1 : T_1 = t_2 : T_2} \qquad \text{Ty-Rec}$$

$$\frac{\Gamma, x_1 : X_1 \succ t_2 : T_2 \mid C}{\Gamma \succ x_1 : X_1 = t_2 : T_2 \mid C \cup \{X_1 = T_2\}}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

CT-Abs

$$\frac{\Gamma, x : T_1 \succ t_2 : T_2}{\Gamma \succ \lambda x : T_1 . t_2 : T_1 \to T_2} \quad \text{Ty-Abs}$$

$$\frac{\Gamma, x : X_1 \succ t_2 : T_2 \,|\, C}{\Gamma \succ \lambda x : X_1 . t_2 : X_1 \to T_2 \,|\, C} \quad \text{CT-Abs}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

CT-App

$$\frac{\Gamma \succ t_1 : T_2 \to T_3 \quad \Gamma \succ t_2 : T_2}{\Gamma \succ t_1\, t_2 : T_3}$$

Ty-App

$$\frac{\Gamma \succ t_1 : T_1 \mid C_1 \quad \Gamma \succ t_2 : T_2 \mid C_2}{\Gamma \succ t_1\, t_2 : X_3 \mid C_1 \cup C_2 \cup \{T_1 = T_2 \to X_3\}}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example $f = \lambda x.\, x\,(f\ x)$

$$
\dfrac{\dfrac{\dfrac{f : X_1, x : X_2 \succ x : X_2 \mid \phi \qquad \dfrac{f : X_1, x : X_2 \succ f : X_1 \mid \phi \quad f : X_1, x : X_2 \succ x : X_2 \mid \phi}{f : X_1, x : X_2 \succ f\ x : X_3 \mid \{X_1 = X_2 \to X_3\} = C_1}}{f : X_1, x : X_2 \succ x\,(f\ x) : X_4 \mid C_1 \cup \{X_2 = X_3 \to X_4\} = C_2}}{f : X_1 \succ \lambda x : X_2 .\, x\,(f\ x) : X_2 \to X_4 \mid C_2}}{\phi \succ f : X_1 = \lambda x : X_2 .\, x\,(f\ x) : X_1 \mid C_2 \cup \{X_1 = X_2 \to X_4\}}
$$

$$
C_3 = \begin{Bmatrix} X_1 = X_2 \to X_3 \\ X_2 = X_3 \to X_4 \\ X_1 = X_2 \to X_4 \end{Bmatrix}
$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example

$$C_3 = \begin{Bmatrix} X_1 = X_2 \to X_3 \\ X_2 = X_3 \to X_4 \\ X_1 = X_2 \to X_4 \end{Bmatrix} \qquad \sigma_1 C_3 = \begin{Bmatrix} X_2 \to X_3 = X_2 \to X_3 \\ X_2 = X_3 \to X_4 \\ X_2 \to X_3 = X_2 \to X_4 \end{Bmatrix}$$

$$\sigma_1 = [X_1 := X_2 \to X_3] \qquad \sigma_2 = [X_1 := X_2 \to X_4, X_3 := X_4]$$

$$\sigma_2 C_3 = \begin{Bmatrix} X_2 \to X_4 = X_2 \to X_4 \\ X_2 = X_4 \to X_4 \\ X_2 \to X_4 = X_2 \to X_4 \end{Bmatrix} \quad \sigma_3 C_3 = \begin{Bmatrix} (X_4 \to X_4) \to X_4 = (X_4 \to X_4) \to X_4 \\ X_4 \to X_4 = X_4 \to X_4 \\ (X_4 \to X_4) \to X_4 = (X_4 \to X_4) \to X_4 \end{Bmatrix}$$

$$\sigma_3 = [X_1 := (X_4 \to X_4) \to X_4, X_3 := X_4, X_2 := X_4 \to X_4]$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Recursive Types

- CT-Rules can be maintained
- Use the Nonstandard Unification Algorithm

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Simple Example $\lambda x.\, x\ x$

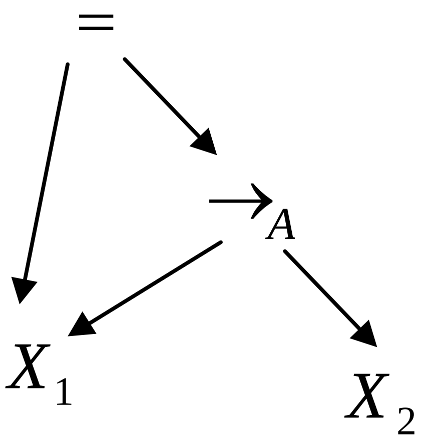
$$\frac{\dfrac{\{x:X_1\} \succ \mathrm{x}:\mathrm{X}_1 \mid \phi \qquad\qquad \{x:X_1\} \succ \mathrm{x}:\mathrm{X}_1 \mid \phi}{\{x:X_1\} \succ x\ x : X_2 \mid \{X_1 = X_1 \to X_2\}}}{\phi \succ \lambda x:X_1.\, x\ x : X_1 \to X_2 \mid \{X_1 = X_1 \to X_2\}}$$

$$C = \{X_1 = X_1 \to X_2\}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Unification

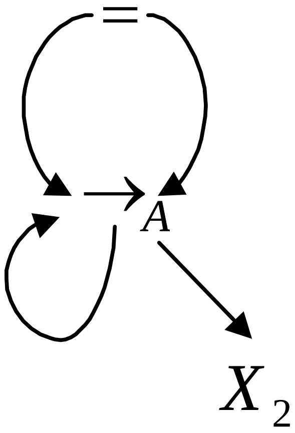
$$\sigma = \phi$$



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Unification

$$\sigma = [X_1 = A]$$



Most general type of $\lambda x.\, x\, x$ is $(\mu\, X_1 : X_1 \to X_2) \to X_2$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Advanced Example

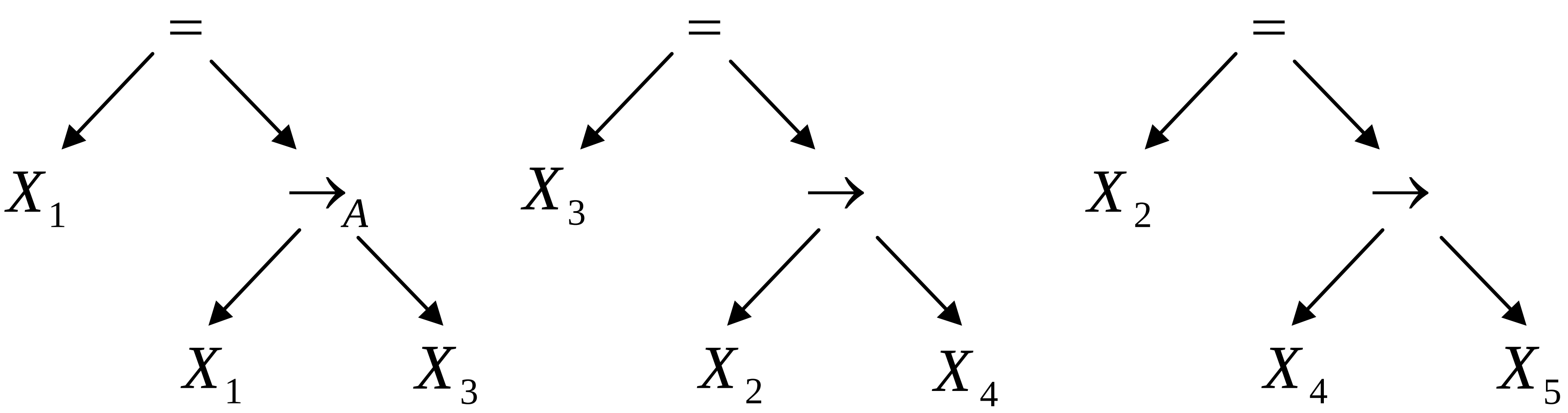
$$F = \lambda x.\lambda y.\, y\,(x\;x\;y) \qquad fix = F\;F$$

$$\dfrac{\dfrac{\Gamma \succ y : X_2 \,|\, \phi \qquad \dfrac{\dfrac{\Gamma \succ x : X_1 \,|\, \phi \qquad \Gamma \succ x : X_1 \,|\, \phi}{\Gamma \succ x\;x : X_3 \,|\, \{X_1 = X_1 \to X_3\} = C_1} \qquad \Gamma \succ y : X_2 \,|\, \phi}{\Gamma \succ x\;x\;y : X_4 \,|\, C_1 \cup \{X_3 = X_2 \to X_4\} = C_2}}{\Gamma = \{x : X_1, y : X_2\} \succ y\,(x\;x\;y) : X_5 \,|\, C_2 \cup \{X_2 = X_4 \to X_5\} = C_3}}{\phi \succ \lambda x : X_1\; \lambda y : X_2.\, y\,(x\;x\;y) : X_1 \to X_2 \to X_5 \,|\, C_3}$$

$$C_3 = \begin{Bmatrix} X_1 = X_1 \to X_3 \\ X_3 = X_2 \to X_4 \\ X_2 = X_4 \to X_5 \end{Bmatrix}$$

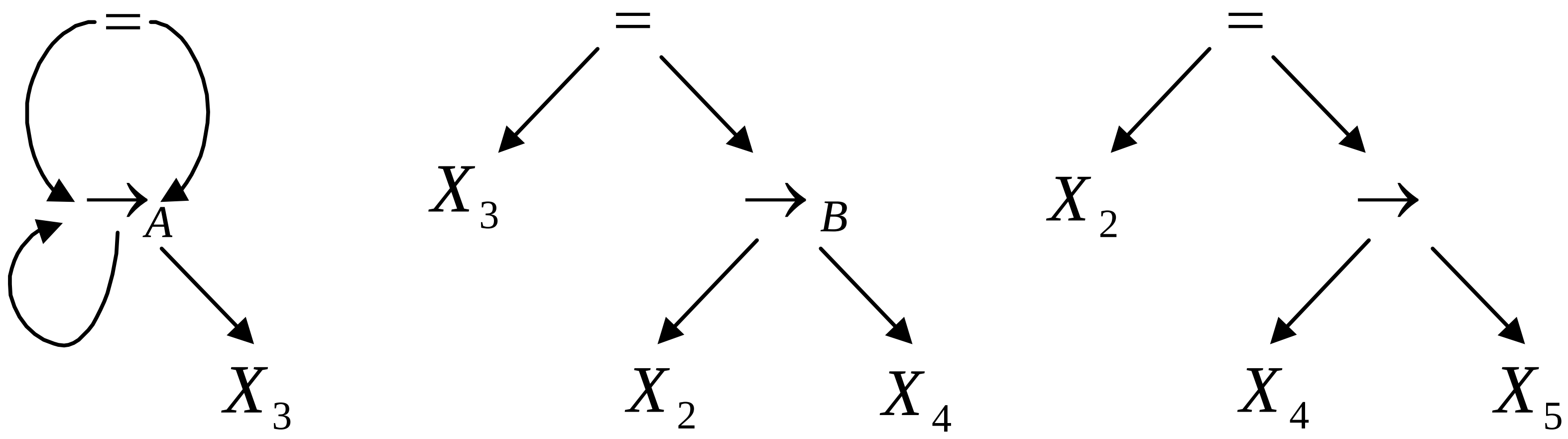
Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

$\sigma = \phi$



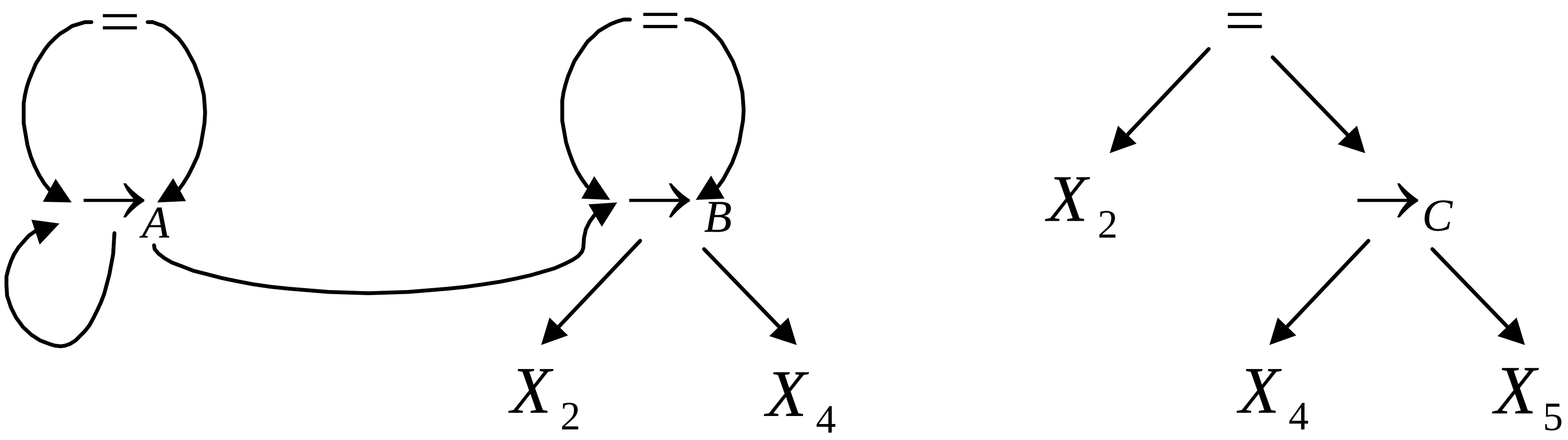
Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

$$\sigma = [X_1 = A]$$



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

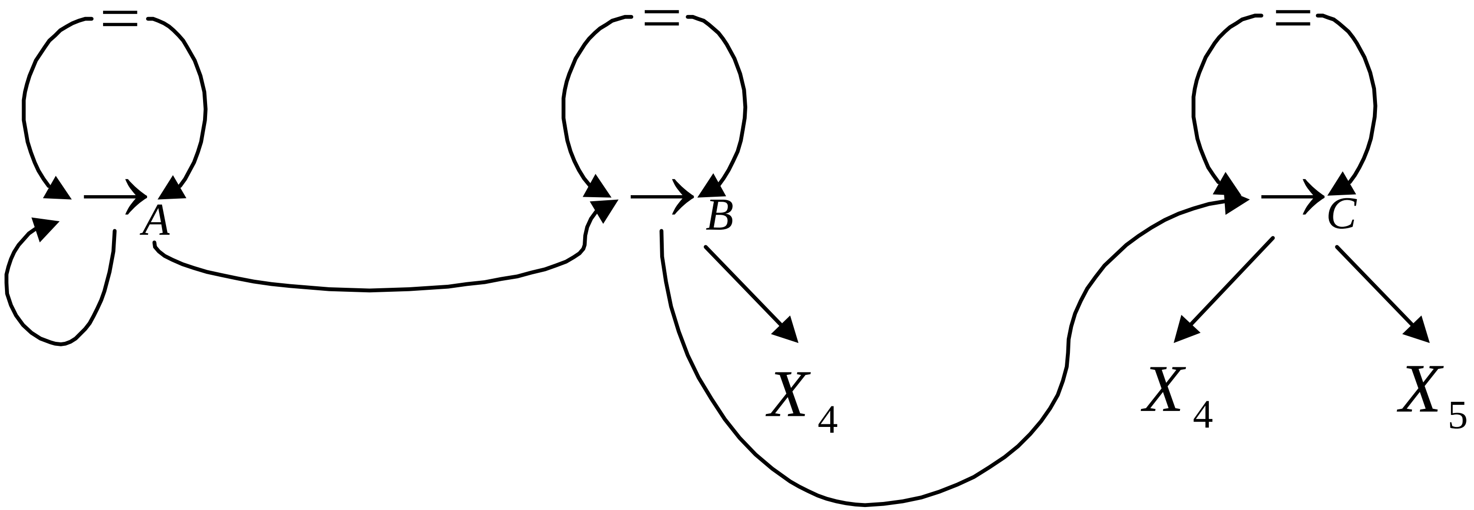
$$\sigma = [X_1 = A, X_3 = B]$$



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

$$\sigma = [X_1 = A, X_3 = B, X_2 = C]$$

$$\lambda x : X_1\, \lambda y : X_2 .\, y\, (x\, x\, y) : X_1 \to X_2 \to X_5$$



Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Correctness

σ satisfies C ⇔ σ *t* is well typed

$$C = \begin{Bmatrix} X_1 = X_2 \to X_3 \\ X_2 = X_3 \to X_4 \\ X_1 = X_2 \to X_4 \end{Bmatrix}$$

$$f : X_1 = \lambda x : X_2.\big(x\,(f\ x) : X_3\big) : X_4$$

Note: It exists a principal type annotation.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Let-Polymorphism

Let-polymorphism, Milner [1978]

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Polymorphism

Polymorphism is a language mechanism that allow a single part of a program to be used with ***different types*** in different contexts.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Let-Polymorphism

Naive Let-Rule:

```
let
  id = λx.x
in
  id 1;
  id true
end
```

$\mathtt{id} : X_1 \to X_1$

$$\underbrace{\begin{array}{l} X_1=\mathtt{Nat} \\ X_1=\mathtt{Bool} \end{array}}_{\text{type clash}}$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Solution: type scheme

Let $X_1,...,X_n$ be the ***free type variables*** of T that do not occur in Γ. We define:

$$\frac{\Gamma \succ x = t_1 : T \mid C \quad \Gamma, x : (\forall X_1,\ldots, X_n : T, C) \succ t_2 : T' \mid C'}{\Gamma \succ \text{let } x = t_1 \text{ in } t_2 \text{ end} : T' \mid C'} \ (\text{CT-Let})$$

$$\frac{x : \left(\forall X_1,\ldots, X_n : T, C\right) \in \Gamma \quad \sigma = [X_1 = X_1',\ldots, X_n = X_n']}{\Gamma \succ x : \sigma\, T \mid \sigma\, C} \ (\text{CT-Var2})$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example

Now, the program is well typed.

```
let
  id = λx.x
in
  id 1;
  id true
end
```

$\mathtt{id} : \forall X_1 : X_1 \rightarrow X_1$

$\mathtt{id} : X_{11} \rightarrow X_{11}$ $\quad X_{11}=\mathtt{Nat}$

$\mathtt{id} : X_{12} \rightarrow X_{12}$ $\quad X_{12}=\mathtt{Bool}$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Problem: side effects

`let`		
`  r = ref(λx.x)`	$r : \forall X_1:(X_1 \to X_1)\mathrm{Ref}$	
`in`		
`  r:= λx:Nat.succ x;`	$r : (X_{11} \to X_{11})\mathrm{Ref}$	$X_{11}=\mathrm{Nat}$
`  (!r) true`	$r : (X_{12} \to X_{12})\mathrm{Ref}$	$X_{12}=\mathrm{Bool}$
`end`		

no type clash

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Solution

$$\frac{\Gamma \succ x = t_1 : T \mid C \qquad \Gamma, x : (\forall X_1, \ldots, X_n : T, C) \succ t_2 : T' \mid C'}{\Gamma \succ \text{let } x = t_1 \text{ in } t_2 \text{ end} : T' \mid C'} \; (\text{CT - Let})$$

Only if t_1 is a value !!!!!

Note: The type scheme is introduced **after** the typechecking of the term $x=t_1$. This means, that you can not use x polymorphically in the term t_1 itself (no polymorphic recursion).

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Example

$$
\begin{array}{lll}
\texttt{let} & & \\
\quad \texttt{r = }\overbrace{\texttt{ref(}\lambda\texttt{x.x)}}^{\text{no value}} & \texttt{r : }\overbrace{(X_1 \to X_1)\texttt{Ref}}^{\text{no type scheme}} & \\
\texttt{in} & & \\
\quad \texttt{r:= }\lambda\texttt{x:Nat.succ x;} & \texttt{r : }(X_1 \to X_1)\texttt{Ref} & X_1\texttt{=Nat} \\
\quad \texttt{(!r) true} & \texttt{r : }(X_1 \to X_1)\texttt{Ref} & \underbrace{X_1\texttt{=Bool}}_{\text{type clash}} \\
\texttt{end} & &
\end{array}
$$

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Restriction

You can not compute a polymorphic function.

E.g:

```
val f = let val i = ref true
        in
           fn x => fn y =>
              (if !i then x else y) before i := not(!i)
        end
```

f is no polymorphic function.

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Runtime is Exponential

The following program is well typed but takes a long time to typecheck.

```
let val f0 = fun x => (x,x) in
  let val f1 = fun y => f0 (f0 y) in
    let val f2 = fun y => f1 (f1 y) in
      let val f3 = fun y => f2 (f2 y) in
        let val f4 = fun y => f3 (f3 y) in
          f4 (fun z => z)
end end end end end
```

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Runtime Analysis

Program	Derived Type	Type Size	Constraints
`let val f0 =`	∀X0:X0→X0*X0	2^0	0
`fun x => (x,x) in`			
`let val f1 = fun y =>`	∀X1:X1 →(X1*X1)*(X1*X1)	2^2	2
`f0 (f0 y) in`			
`let val f2 = fun y =>`	∀X2:X2→((((X2*X2)*(X2*X2))*	2^4	4
`f1 (f1 y) in`	((X2*X2)*(X2*X2)))*		
`let val f3 = fun y =>`	(((X2*X2)*(X2*X2))*	2^8	8
`f2 (f2 y) in`	((X2*X2)*(X2*X2))))		
`let val f4 = fun y =>`		2^{16}	16
`f3 (f3 y) in`	(...)		
`f4 (fun z => z)`			
`end end end end end`			

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Overview

- Unification, Nonstandard Unification
- Constraint typing rules for λ-calculus (similar to standard typing rules)
- It exists a principal type annotation as the solution of a set of constraints (Unification Theorem)
- Constraint typing rules and recursive types
- Let-Polymorphism

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Historical Context

- Unification, [Robinson, 1965]
- Unification in linear space complexity [Martelli, Montanary, 1984]
- Nonstandard Unification ???
- Principal Types, Curry and Feys [1958]
- Algorithm to compute principal types, Hindley [1969]
- Type reconstruction, Algorithm W, Damas and Milner [1982]
- Type Reconstruction with Recursive Types [Huet, 1975, 1976]
- Let-polymorphism, Milner [1978]

Visage eXPert PDF Copyright © 1998,2003 Visage Software
This document was created with free TRIAL version of Visage eXPert PDF.This watermark will be removed
after purchasing the licensed full version of Visage eXPert PDF. Please visit http://www.visagesoft.com for more details

Questions?