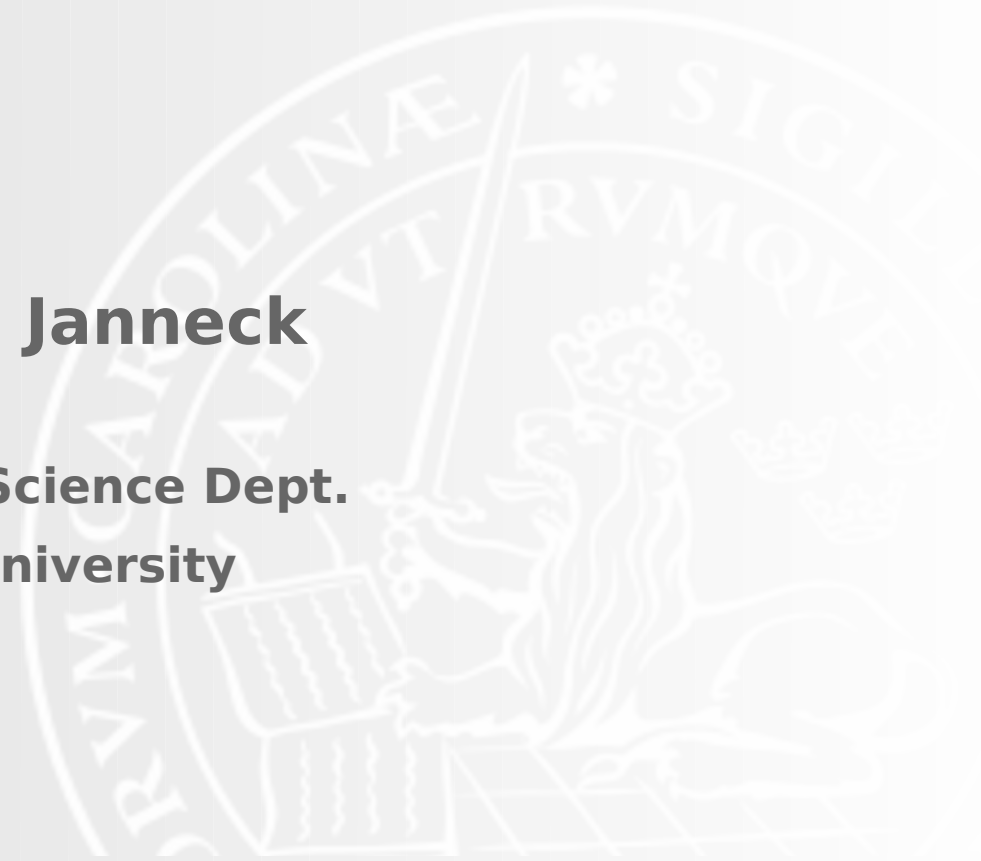


Type Systems

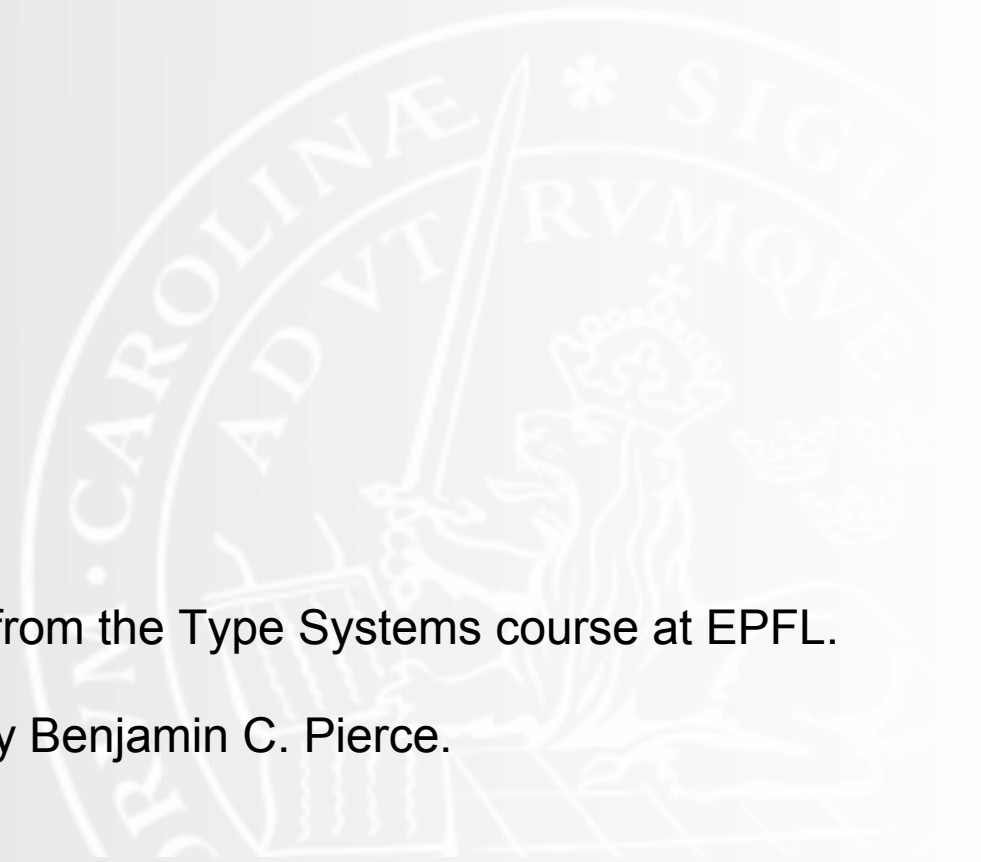
07 - Type Reconstruction

Jörn W. Janneck

**Computer Science Dept.
Lund University**



Slides begged, borrowed, and copied from the Type Systems course at EPFL.
Courtesy of Martin Odersky.
Some material also lifted from TAPL by Benjamin C. Pierce.



Type Reconstruction



Type Checking and Type Reconstruction

We now come to the question of type checking and type reconstruction.

Type checking: Given Γ , t and T , check whether $\Gamma \vdash t : T$

Type reconstruction: Given Γ and t , find a type T such that $\Gamma \vdash t : T$

Type checking and reconstruction seem difficult since parameters in lambda calculus do not carry their types with them.

Type reconstruction also suffers from the problem that a term can have many types.

Idea: : We construct all type derivations in parallel, reducing type reconstruction to a unification problem.

substitutions

Substitution: mapping σ from type variables to types

eg. $\{T : \text{Nat}, U : X \rightarrow \text{Nat}, V : \text{Bool} \rightarrow \text{Bool}\}$

extends naturally to mappings on

types:

$$\begin{aligned}\sigma(X) &= \begin{cases} T & \text{if } (X \mapsto T) \in \sigma \\ X & \text{if } X \text{ is not in the domain of } \sigma \end{cases} \\ \sigma(\text{Nat}) &= \text{Nat} \\ \sigma(\text{Bool}) &= \text{Bool} \\ \sigma(T_1 \rightarrow T_2) &= \sigma T_1 \rightarrow \sigma T_2\end{aligned}$$

contexts:

$$\sigma(x_1 : T_1, \dots, x_n : T_n) = (x_1 : \sigma T_1, \dots, x_n : \sigma T_n).$$

type equations:

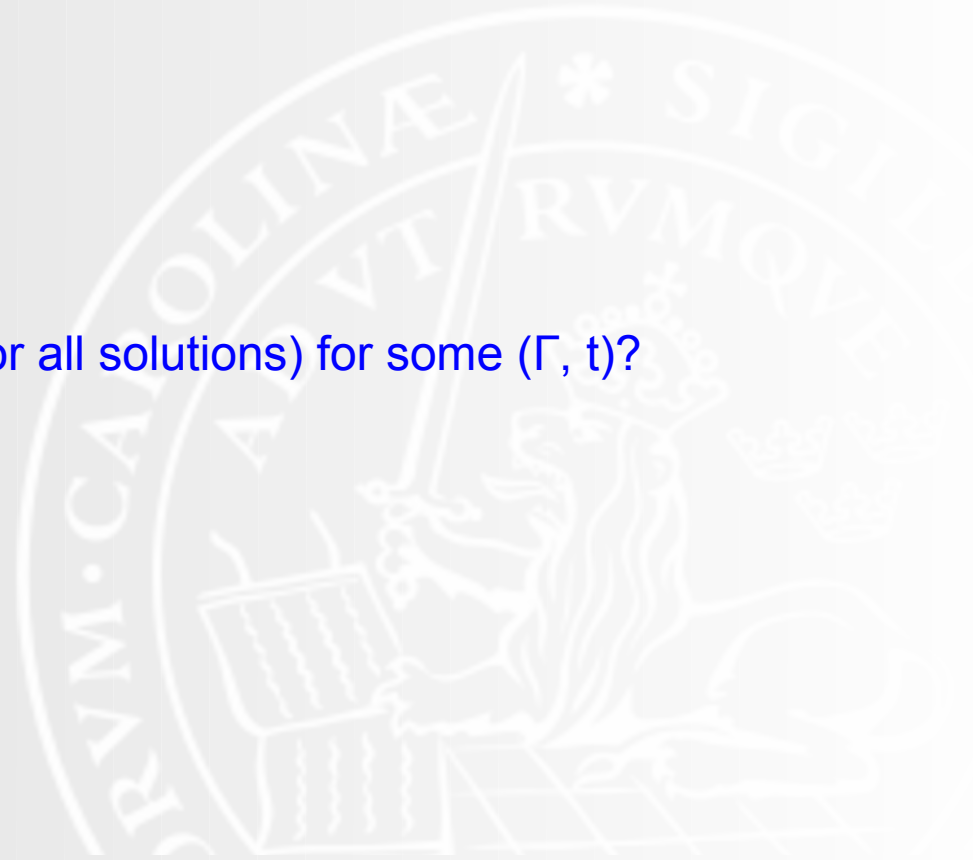
$$\sigma(S == T) = \sigma(S) == \sigma(T)$$

22.1.2 THEOREM [PRESERVATION OF TYPING UNDER TYPE SUBSTITUTION]: If σ is any type substitution and $\Gamma \vdash t : T$, then $\sigma\Gamma \vdash \sigma t : \sigma T$. \square

solutions

22.2.1 DEFINITION: Let Γ be a context and t a term. A *solution* for (Γ, t) is a pair (σ, T) such that $\sigma\Gamma \vdash \sigma t : T$. \square

How do we find a solution (or all solutions) for some (Γ, t) ?



constraint-based typing

Problem:

Given (Γ, t) find solution (σ, T) .

Basic approach:

1. transform problem $\Gamma \vdash t : T$ into set of **equations** (aka constraints)

$$\Gamma \vdash t : T \mid x \ C$$

2. compute the solutions for the equations

constraint-based typing: equations

$$\Gamma \vdash t : T \mid_X C$$

$\frac{x:T \in \Gamma}{\Gamma \vdash x : T \mid_{\emptyset} \{ \}} \quad \text{(CT-VAR)}$	
$\frac{\Gamma, x:T_1 \vdash t_2 : T_2 \mid_X C}{\Gamma \vdash \lambda x:T_1. t_2 : T_1 \rightarrow T_2 \mid_X C} \quad \text{(CT-ABS)}$	
$\frac{\Gamma \vdash t_1 : T_1 \mid_{X_1} C_1 \quad \Gamma \vdash t_2 : T_2 \mid_{X_2} C_2 \quad X_1 \cap X_2 = X_1 \cap FV(T_2) = X_2 \cap FV(T_1) = \emptyset \quad X \notin X_1, X_2, T_1, T_2, C_1, C_2, \Gamma, t_1, \text{ or } t_2 \quad C' = C_1 \cup C_2 \cup \{T_1 = T_2 \rightarrow X\}}{\Gamma \vdash t_1 t_2 : X \mid_{X_1 \cup X_2 \cup \{X\}} C'} \quad \text{(CT-APP)}$	
$\Gamma \vdash 0 : \text{Nat} \mid_{\emptyset} \{ \} \quad \text{(CT-ZERO)}$	
$\frac{\Gamma \vdash t_1 : T \mid_X C \quad C' = C \cup \{T = \text{Nat}\}}{\Gamma \vdash \text{succ } t_1 : \text{Nat} \mid_X C'} \quad \text{(CT-SUCC)}$	
	$\frac{\Gamma \vdash t_1 : T \mid_X C \quad C' = C \cup \{T = \text{Nat}\}}{\Gamma \vdash \text{pred } t_1 : \text{Nat} \mid_X C'} \quad \text{(CT-PRED)}$
	$\frac{\Gamma \vdash t_1 : T \mid_X C \quad C' = C \cup \{T = \text{Nat}\}}{\Gamma \vdash \text{iszero } t_1 : \text{Bool} \mid_X C'} \quad \text{(CT-ISZERO)}$
	$\Gamma \vdash \text{true} : \text{Bool} \mid_{\emptyset} \{ \} \quad \text{(CT-TRUE)}$
	$\Gamma \vdash \text{false} : \text{Bool} \mid_{\emptyset} \{ \} \quad \text{(CT-FALSE)}$
	$\frac{\Gamma \vdash t_1 : T_1 \mid_{X_1} C_1 \quad \Gamma \vdash t_2 : T_2 \mid_{X_2} C_2 \quad \Gamma \vdash t_3 : T_3 \mid_{X_3} C_3 \quad X_1, X_2, X_3 \text{ nonoverlapping} \quad C' = C_1 \cup C_2 \cup C_3 \cup \{T_1 = \text{Bool}, T_2 = T_3\}}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T_2 \mid_{X_1 \cup X_2 \cup X_3} C'} \quad \text{(CT-IF)}$

Figure 22-1: Constraint typing rules

constraint-based typing: equations

$$\Gamma \vdash t : T \mid_{\mathcal{X}} C$$

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

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

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

constraint-based typing: equations

$$\Gamma \vdash t : T \mid_{\mathcal{X}} C$$

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

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

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

constraint-based typing: equations

$$\Gamma \vdash \mathbf{t} : \mathsf{T} \mid_{\mathcal{X}} \mathcal{C}$$

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

$$\frac{\Gamma, \mathbf{x} : \mathsf{T}_1 \vdash \mathbf{t}_2 : \mathsf{T}_2 \mid_{\mathcal{X}} \mathcal{C}}{\Gamma \vdash \lambda \mathbf{x} : \mathsf{T}_1 . \mathbf{t}_2 : \mathsf{T}_1 \rightarrow \mathsf{T}_2 \mid_{\mathcal{X}} \mathcal{C}} \quad \text{(CT-ABS)}$$

constraint-based typing: equations $\Gamma \vdash t : T \mid_{\mathcal{X}} C$

$$\Gamma \vdash t_1 : T_1 \mid_{\mathcal{X}_1} C_1$$

$$\Gamma \vdash t_2 : T_2 \mid_{\mathcal{X}_2} C_2 \quad \Gamma \vdash t_3 : T_3 \mid_{\mathcal{X}_3} C_3$$

$\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ nonoverlapping

$$C' = C_1 \cup C_2 \cup C_3 \cup \{T_1 = \text{Bool}, T_2 = T_3\}$$

$$\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T_2 \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} C'$$

(CT-IF)

constraint-based typing: equations $\Gamma \vdash t : T \mid_{\mathcal{X}} C$

$$\frac{\begin{array}{l} \Gamma \vdash t_1 : T_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash t_2 : T_2 \mid_{\mathcal{X}_2} C_2 \\ \mathcal{X}_1 \cap \mathcal{X}_2 = \mathcal{X}_1 \cap FV(T_2) = \mathcal{X}_2 \cap FV(T_1) = \emptyset \\ \underline{X} \notin \mathcal{X}_1, \mathcal{X}_2, T_1, T_2, C_1, C_2, \Gamma, t_1, \text{ or } t_2 \\ C' = C_1 \cup C_2 \cup \{T_1 = T_2 \rightarrow X\} \end{array}}{\Gamma \vdash t_1 t_2 : X \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \{X\}} C'} \quad (\text{CT-APP})$$

constraint-based typing

Problem:

Given (Γ, t) find solution (σ, T) .

Basic approach:

1. transform problem $\Gamma \vdash t : T$ into set of **equations** (aka constraints)

$$\Gamma \vdash t : T \mid x \ C$$

2. compute the solutions for the equations

$$\sigma = \text{unify}(C)$$

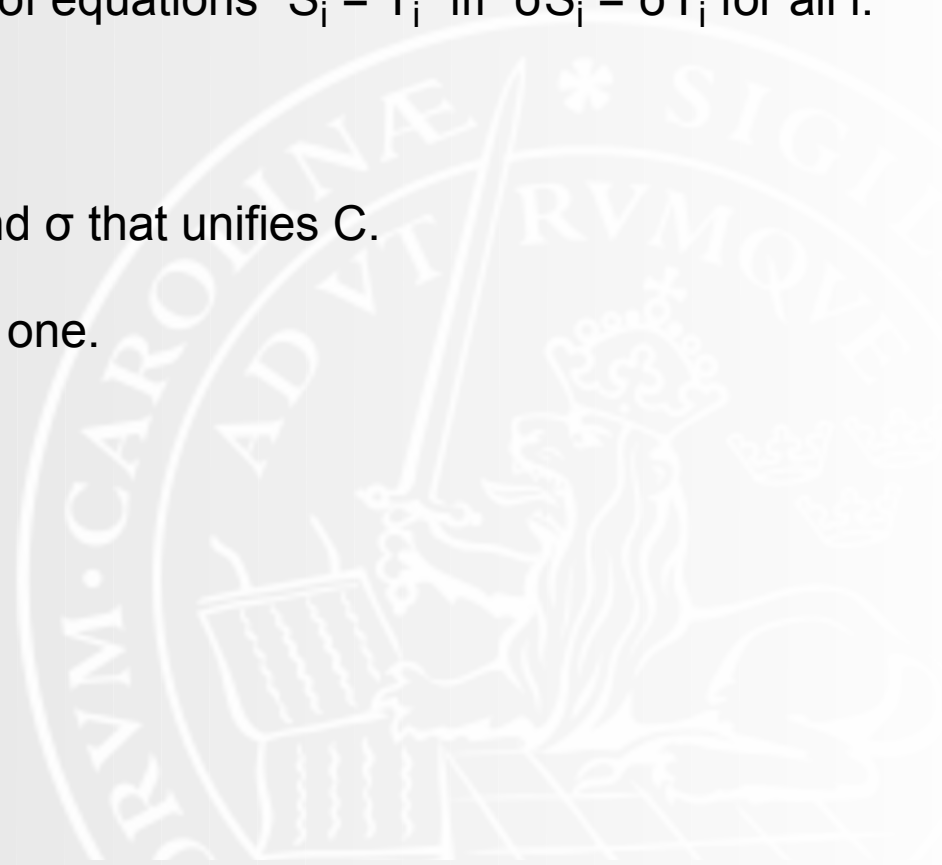
Constraint-based typing: unification

Recall: a substitution is a mapping from type variables to types.

Definition: A substitution σ *unifies* a set of equations " $S_i = T_i$ " iff $\sigma S_i = \sigma T_i$ for all i .

Problem: Given a set of equations C , find σ that unifies C .

Subproblem: There might be more than one.



Constraint-based typing: unification

```
unify(C)  =  if C = ∅, then [ ]  
             else let {S = T} ∪ C' = C in  
               if S = T  
                 then unify(C')  
               else if S = X and X ∉ FV(T)  
                 then unify([X ↦ T]C') ∘ [X ↦ T]  
               else if T = X and X ∉ FV(S)  
                 then unify([X ↦ S]C') ∘ [X ↦ S]  
               else if S = S1 → S2 and T = T1 → T2  
                 then unify(C' ∪ {S1 = T1, S2 = T2})  
               else  
                 fail
```

Figure 22-2: Unification algorithm

recap

type checking

Given Γ , t , and T , check whether

$$\Gamma \vdash t : T$$

type reconstruction

Given Γ , and t , find T such that

$$\Gamma \vdash t : T \quad \xrightarrow{\quad} \quad \Gamma \vdash t : T \mid_x C \quad \xrightarrow{\quad} \quad \sigma = \text{unify}(C)$$

$(\sigma, \sigma T)$

two problems, two solutions

22.2.1 DEFINITION: Let Γ be a context and t a term. A *solution* for (Γ, t) is a pair (σ, T) such that $\sigma\Gamma \vdash \sigma t : T$. \square

$$\Gamma \vdash t : T \quad \Rightarrow \quad \Gamma \vdash t : T \mid_{\mathcal{X}} C \quad \Rightarrow \quad \sigma = \text{unify}(C)$$

22.3.4 DEFINITION: Suppose that $\Gamma \vdash t : S \mid C$. A *solution* for (Γ, t, S, C) is a pair (σ, T) such that σ satisfies C and $\sigma S = T$. \square

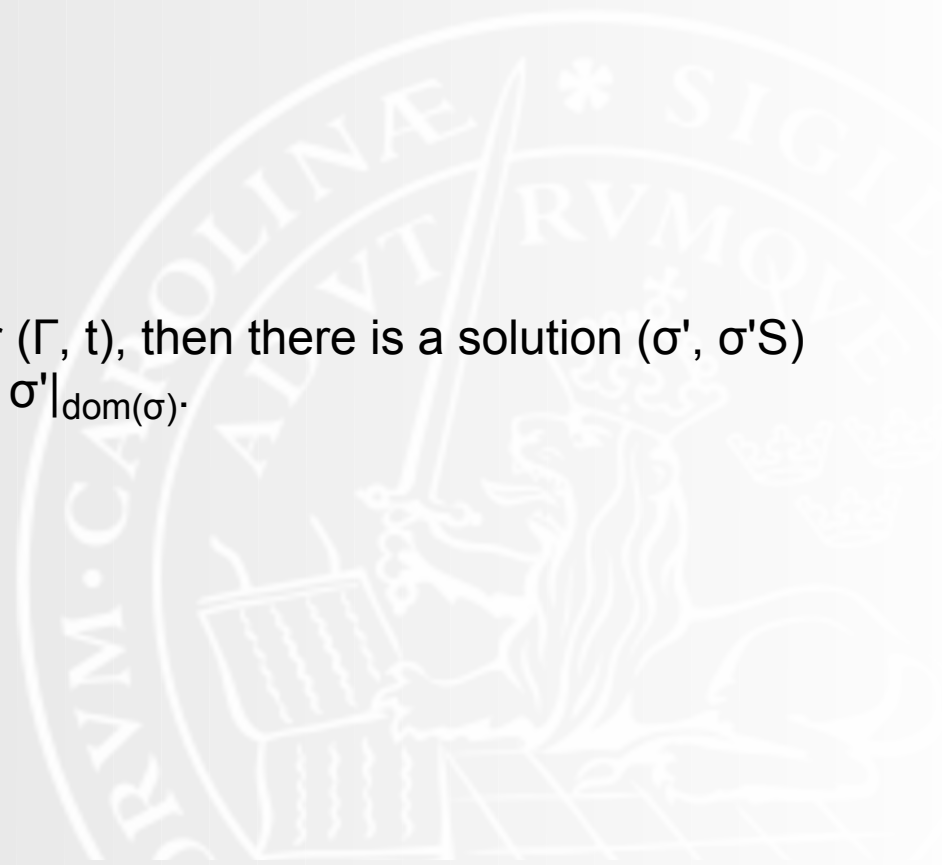
soundness and completeness

Soundness: If $(\sigma, \sigma T)$ is a solution for (Γ, t, T, C) , then it is also a solution for (Γ, t) .

(Theorem 22.3.5, p.323)

Completeness: If (σ, T) is a solution for (Γ, t) , then there is a solution $(\sigma', \sigma'S)$ for (Γ, t, S, C) such that $T = \sigma'S$ and $\sigma = \sigma'|_{\text{dom}(\sigma)}$.

(Theorem 22.3.7, p. 324)



principal unifier

preorder substitutions based on their 'specificity':

$\sigma \leq \sigma'$ iff exists γ such that $\gamma \circ \sigma = \sigma'$

principal unifier for C:

some σ such that

1. σ satisfies C and
2. for all σ' satisfying C, $\sigma \leq \sigma'$

unification theorem: (22.4.5, p.328)

1. $\text{unify}(C)$ halts for all C
2. it only fails if there is no unifier for C
3. $\sigma = \text{unify}(C)$ implies σ is a unifier for C
4. $\text{unify}(C)$ is a principal unifier for C

principal types

principal solution, principal type:

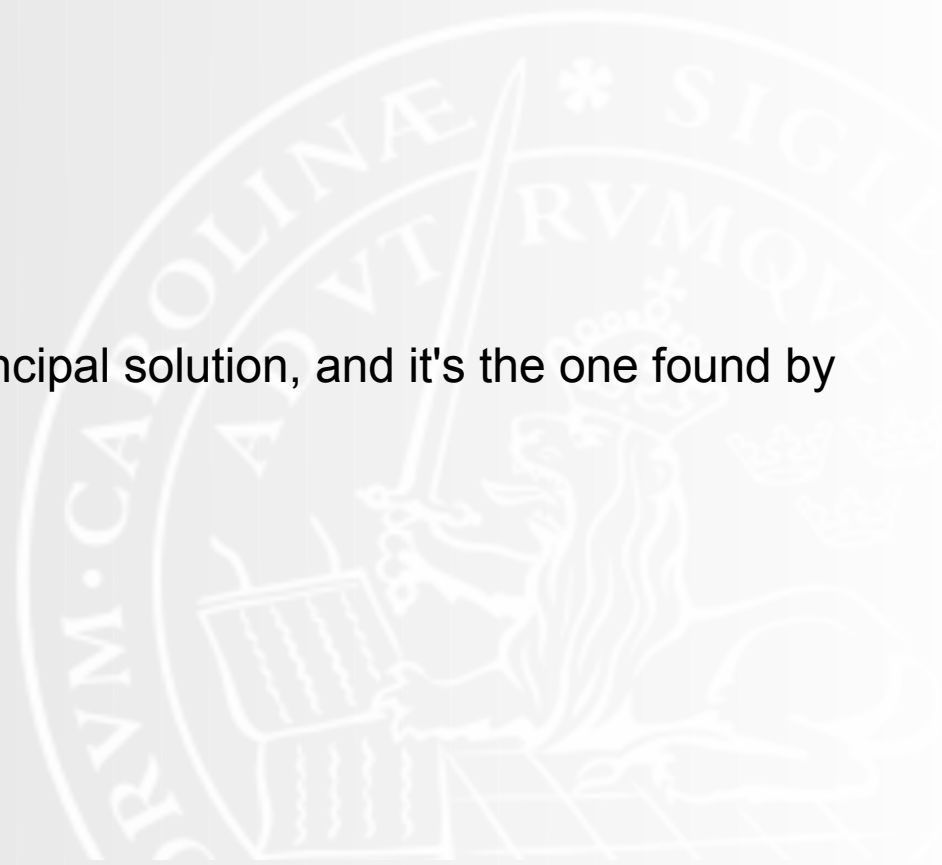
A solution (σ, T) of (Γ, t, S, C) is a *principal solution* iff for all other solutions (σ', T) it is the case that $\sigma \leq \sigma'$. T is then the *principal type* of t under Γ .

(Def 22.5.1)

principal type theorem:

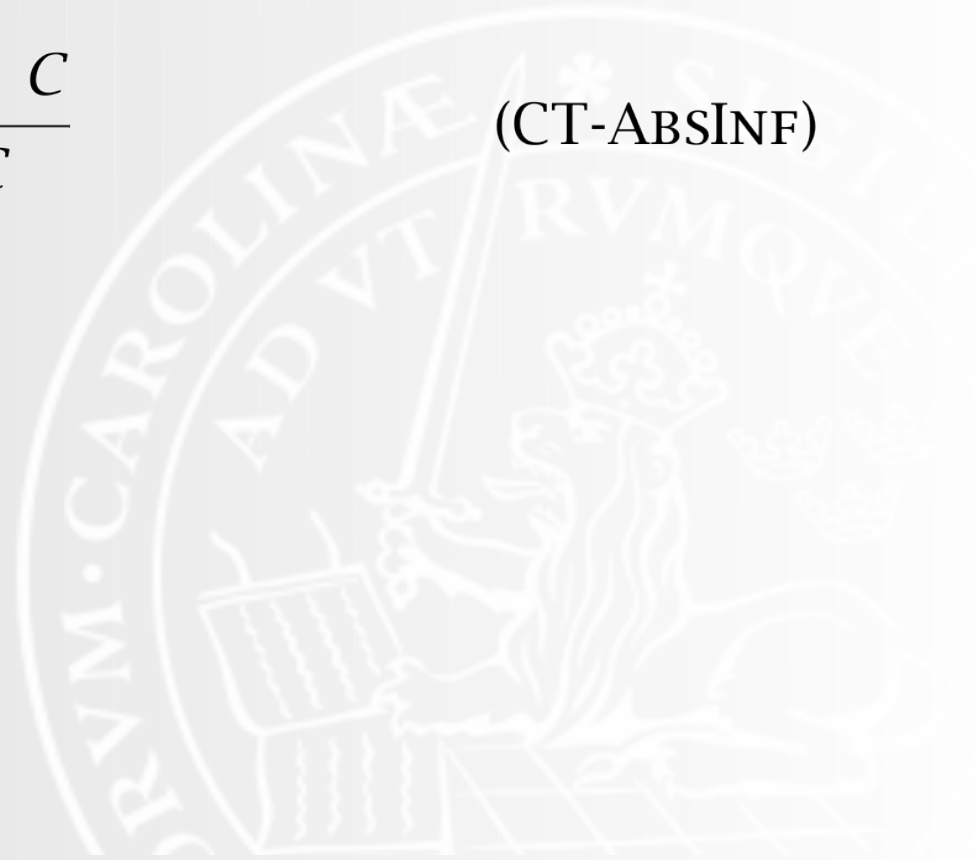
If (Γ, t, S, C) has a solution, it has a principal solution, and it's the one found by $\text{unify}(C)$.

(Theorem 22.5.3)



implicit type annotations

$$\frac{X \notin \mathcal{X} \quad \Gamma, x:X \vdash t_1 : T \mid_{\mathcal{X}} C}{\Gamma \vdash \lambda x. t_1 : X \rightarrow T \mid_{\mathcal{X} \cup \{X\}} C} \text{ (CT-ABSINF)}$$



an odd case

$(\lambda x:A. xx)(\lambda x:B. xx)$

