Types, Polymorphism, and Type Reconstruction

Sources

This material is based on the following sources:

- Pierce, B.C., Types and Programming Languages. MIT Press, 2002.
- Kanellakis, P.C., Mairson, H.G. and Mitchell, J.C., Unification and ML type reconstruction. In Computational Logic: Essays in Honor of Alan Robinson, ed. J.-L. Lassez and G.D. Plotkin, MIT Press, 1991, pages 444–478.
- Damas L., Milner, R., Principal type-schemes for functional programs. Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM, 1982, pages 207–212.

Recommended further reading:

- Pierce, B.C., Turner D.N., Local type inference. ACM Transactions on Programming Languages and Systems, Volume 22 Issue 1, Jan. 2000, pages 1–44.

The Simply Typed Lambda-Calculus

The simply typed lambda-calculus

The system we are about to define is commonly called the *simply typed lambda-calculus*, or $\lambda_{\to}$ for short.

Unlike the untyped lambda-calculus, the "pure" form of $\lambda_{\to}$ (with no primitive values or operations) is not very interesting; to talk about $\lambda_{\to}$, we always begin with some set of "base types."

- So, strictly speaking, there are *many* variants of $\lambda_{\to}$, depending on the choice of base types.
- For now, we'll work with a variant constructed over the booleans.

Untyped lambda-calculus with booleans

t ::=		*terms*
	x	*variable*
	λx.t	*abstraction*
	t t	*application*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*
v ::=		*values*
	λx.t	*abstraction value*
	true	*true value*
	false	*false value*

"Simple Types"

T ::=		*types*
	Bool	*type of booleans*
	T→T	*types of functions*

What are some examples?

Type Annotations

We now have a choice to make. Do we...

- annotate lambda-abstractions with the expected type of the argument

  $$\lambda x{:}T_1.\ t_2$$

  (as in most mainstream programming languages), or
- continue to write lambda-abstractions as before

  $$\lambda x.\ t_2$$

  and ask the typing rules to "guess" an appropriate annotation (as in OCaml)?

Both are reasonable choices, but the first makes the job of defining the typing rules simpler. Let's take this choice for now.

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

$$\frac{???}{\lambda x{:}T_1 . t_2 : T_1 \to T_2} \qquad \text{(T-Abs)}$$

Typing rules

$$\texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{t_1 : \texttt{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

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

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x : T} \qquad \text{(T-Var)}$$

Typing rules

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad \text{(T-If)}$$

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

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x : T} \qquad \text{(T-Var)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \qquad \text{(T-App)}$$

Typing Derivations

What derivations justify the following typing statements?

- $\vdash$ `(λx:Bool.x) true : Bool`
- `f:Bool→Bool` $\vdash$
  `f (if false then true else false) : Bool`
- `f:Bool→Bool` $\vdash$
  `λx:Bool. f (if x then false else x) : Bool→Bool`

Properties of $\lambda_{\to}$

The fundamental property of the type system we have just defined is *soundness* with respect to the operational semantics.

1. *Progress:* A closed, well-typed term is not stuck

   If $\vdash t : T$, *then either* t *is a value or else* $t \longrightarrow t'$ *for some* t'.

2. *Preservation:* Types are preserved by one-step evaluation

   If $\Gamma \vdash t : T$ *and* $t \longrightarrow t'$, *then* $\Gamma \vdash t' : T$.

Proving progress

- inversion lemma for typing relation
- canonical forms lemma
- progress theorem

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x:T}_1 \texttt{.t}_2 : \texttt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x:T}_1 . \texttt{t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \to \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x:T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.
5. If $\Gamma \vdash \lambda\texttt{x:T}_1\texttt{.t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \to \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x:T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.
6. If $\Gamma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{R}$, then

Inversion

Lemma:

1. If $\Gamma \vdash \texttt{true} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
2. If $\Gamma \vdash \texttt{false} : \texttt{R}$, then $\texttt{R} = \texttt{Bool}$.
3. If $\Gamma \vdash \texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3 : \texttt{R}$, then $\Gamma \vdash \texttt{t}_1 : \texttt{Bool}$ and $\Gamma \vdash \texttt{t}_2, \texttt{t}_3 : \texttt{R}$.
4. If $\Gamma \vdash \texttt{x} : \texttt{R}$, then $\texttt{x:R} \in \Gamma$.
5. If $\Gamma \vdash \lambda \texttt{x:T}_1 \texttt{.t}_2 : \texttt{R}$, then $\texttt{R} = \texttt{T}_1 \to \texttt{R}_2$ for some $\texttt{R}_2$ with $\Gamma, \texttt{x:T}_1 \vdash \texttt{t}_2 : \texttt{R}_2$.
6. If $\Gamma \vdash \texttt{t}_1 \ \texttt{t}_2 : \texttt{R}$, then there is some type $\texttt{T}_{11}$ such that $\Gamma \vdash \texttt{t}_1 : \texttt{T}_{11} \to \texttt{R}$ and $\Gamma \vdash \texttt{t}_2 : \texttt{T}_{11}$.

Canonical Forms

Lemma:

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then

Canonical Forms

Lemma:

1. If `v` is a value of type `Bool`, then `v` is either `true` or `false`.

Canonical Forms

Lemma:

1. If $\mathtt{v}$ is a value of type $\mathtt{Bool}$, then $\mathtt{v}$ is either $\mathtt{true}$ or $\mathtt{false}$.
2. If $\mathtt{v}$ is a value of type $\mathtt{T_1 \to T_2}$, then

Canonical Forms

Lemma:

1. If $\mathtt{v}$ is a value of type $\mathtt{Bool}$, then $\mathtt{v}$ is either $\mathtt{true}$ or $\mathtt{false}$.
2. If $\mathtt{v}$ is a value of type $\mathtt{T}_1 \to \mathtt{T}_2$, then $\mathtt{v}$ has the form $\lambda \mathtt{x}\!:\!\mathtt{T}_1 \,.\, \mathtt{t}_2$.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.
Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.
Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$. By the induction hypothesis, either $\mathtt{t}_1$ is a value or else it can make a step of evaluation, and likewise $\mathtt{t}_2$.

Progress

Theorem: Suppose $\mathtt{t}$ is a closed, well-typed term (that is, $\vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$). Then either $\mathtt{t}$ is a value or else there is some $\mathtt{t}'$ with $\mathtt{t} \longrightarrow \mathtt{t}'$.

Proof: By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because $\mathtt{t}$ is closed). The abstraction case is immediate, since abstractions are values.
Consider the case for application, where $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$ with $\vdash \mathtt{t}_1 : \mathtt{T}_{11}{\to}\mathtt{T}_{12}$ and $\vdash \mathtt{t}_2 : \mathtt{T}_{11}$. By the induction hypothesis, either $\mathtt{t}_1$ is a value or else it can make a step of evaluation, and likewise $\mathtt{t}_2$. If $\mathtt{t}_1$ can take a step, then rule E-App1 applies to $\mathtt{t}$. If $\mathtt{t}_1$ is a value and $\mathtt{t}_2$ can take a step, then rule E-App2 applies. Finally, if both $\mathtt{t}_1$ and $\mathtt{t}_2$ are values, then the canonical forms lemma tells us that $\mathtt{t}_1$ has the form $\lambda \mathtt{x}{:}\mathtt{T}_{11}.\mathtt{t}_{12}$, and so rule E-AppAbs applies to $\mathtt{t}$.

Preservation for STLC

Preservation for STLC

Theorem: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\Gamma \vdash \mathtt{t}' : \mathtt{T}$.

Proof: By induction

Preservation for STLC

Theorem: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\Gamma \vdash \mathtt{t}' : \mathtt{T}$.

Proof: By induction on typing derivations.

Which case is the hard one??

Preservation for STLC

Theorem: If $\Gamma \vdash t : T$ and $t \longrightarrow t'$, then $\Gamma \vdash t' : T$.

Proof: By induction on typing derivations.

Case T-App: Given $t = t_1\ t_2$

$\Gamma \vdash t_1 : T_{11} \to T_{12}$

$\Gamma \vdash t_2 : T_{11}$

$T = T_{12}$

Show $\Gamma \vdash t' : T_{12}$

Preservation for STLC

Theorem: If $\Gamma \vdash t : T$ and $t \longrightarrow t'$, then $\Gamma \vdash t' : T$.

Proof: By induction on typing derivations.

Case T-APP: Given $t = t_1\ t_2$

$\Gamma \vdash t_1 : T_{11} \to T_{12}$

$\Gamma \vdash t_2 : T_{11}$

$T = T_{12}$

Show $\Gamma \vdash t' : T_{12}$

By the inversion lemma for evaluation, there are three subcases...

The "Substitution Lemma"

Lemma: Types are preserved under substitition.

That is, if $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

The "Substitution Lemma"

Lemma: Types are preserved under substitition.

That is, if $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: ...

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathit{dom}(\Gamma)$, then $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathit{dom}(\Gamma)$, then $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.

Permutation tells us that the order of assumptions in (the list) Γ does not matter.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and Δ is a permutation of Γ, then $\Delta \vdash \mathtt{t} : \mathtt{T}$.

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathit{dom}(\Gamma)$, then $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$.

Moreover, the latter derivation has the same depth as the former.

Permutation tells us that the order of assumptions in (the list) Γ does not matter.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and Δ is a permutation of Γ, then $\Delta \vdash \mathtt{t} : \mathtt{T}$.

Moreover, the latter derivation has the same depth as the former.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

I.e., "Types are preserved under substitition."

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$. Proceed by cases on the final typing rule used in the derivation.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x{:}S} \vdash \mathtt{t} : \mathtt{T}$. Proceed by cases on the final typing rule used in the derivation.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$. Proceed by cases on the final typing rule used in the derivation.

Case T-App:
$$\begin{aligned} &\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2 \\ &\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1 \\ &\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t}_2 : \mathtt{T}_2 \\ &\mathtt{T} = \mathtt{T}_1 \end{aligned}$$

By the induction hypothesis, $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$ and $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2 : \mathtt{T}_2$. By T-App, $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2 : \mathtt{T}$, i.e., $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}](\mathtt{t}_1\ \mathtt{t}_2) : \mathtt{T}$.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$. Proceed by cases on the final typing rule used in the derivation.

Case T-VAR: $\mathtt{t} = \mathtt{z}$
with $\mathtt{z}:\mathtt{T} \in (\Gamma, \mathtt{x}:\mathtt{S})$

There are two sub-cases to consider, depending on whether $\mathtt{z}$ is $\mathtt{x}$ or another variable. If $\mathtt{z} = \mathtt{x}$, then $[\mathtt{x} \mapsto \mathtt{s}]\mathtt{z} = \mathtt{s}$. The required result is then $\Gamma \vdash \mathtt{s} : \mathtt{S}$, which is among the assumptions of the lemma. Otherwise, $[\mathtt{x} \mapsto \mathtt{s}]\mathtt{z} = \mathtt{z}$, and the desired result is immediate.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}:\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$. Proceed by cases on the final typing rule used in the derivation.

Case T-Abs: $\mathtt{t} = \lambda \mathtt{y}:\mathtt{T}_2.\mathtt{t}_1 \qquad \mathtt{T} = \mathtt{T}_2 \to \mathtt{T}_1$
$\Gamma, \mathtt{x}:\mathtt{S}, \mathtt{y}:\mathtt{T}_2 \vdash \mathtt{t}_1 : \mathtt{T}_1$

By our conventions on choice of bound variable names, we may assume $\mathtt{x} \neq \mathtt{y}$ and $\mathtt{y} \notin \mathit{FV}(\mathtt{s})$. Using *permutation* on the given subderivation, we obtain $\Gamma, \mathtt{y}:\mathtt{T}_2, \mathtt{x}:\mathtt{S} \vdash \mathtt{t}_1 : \mathtt{T}_1$. Using *weakening* on the other given derivation ($\Gamma \vdash \mathtt{s} : \mathtt{S}$), we obtain $\Gamma, \mathtt{y}:\mathtt{T}_2 \vdash \mathtt{s} : \mathtt{S}$. Now, by the induction hypothesis, $\Gamma, \mathtt{y}:\mathtt{T}_2 \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_1$. By T-Abs, $\Gamma \vdash \lambda \mathtt{y}:\mathtt{T}_2.\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$, i.e. (by the definition of substitution), $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\lambda \mathtt{y}:\mathtt{T}_2.\ \mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$.

Summary: Preservation

Theorem: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$, then $\Gamma \vdash \mathtt{t}' : \mathtt{T}$.

Lemmas to prove:

- Weakening
- Permutation
- Substitution preserves types
- Reduction preserves types (i.e., preservation)

Review: Type Systems

To define and verify a type system, you must:

1. Define types
2. Specify typing rules
3. Prove soundness: *progress* and *preservation*