Strong Inductions

- The nominal datatype package generates the α-equivalence classes as a **type** in Isabelle/HOL.

```
atom_decl name

nominal_datatype lam =
        Var  "name"
      | App  "lam" "lam"
      | Lam  "«name»lam"  ("Lam [_]._" [100,100] 100)
```

The type is defined so that we have **equations** such as:

$$\text{Lam}\ [a].(\text{Var}\ a) = \text{Lam}\ [b].(\text{Var}\ b)$$

Struct. Induction for Free

$$\frac{}{\mathsf{Var}(a) \in \mathsf{lam}} \qquad \frac{t \in \mathsf{lam} \quad s \in \mathsf{lam}}{\mathsf{App}(t, s) \in \mathsf{lam}}$$

$$\frac{t \in \mathsf{lam}}{\mathsf{Lam}\,[a].t \in \mathsf{lam}}$$

Structural Induction Principle:

$$\frac{\begin{array}{l}\forall a.\ P\,(\mathsf{Var}(a)) \\ \forall t\ s.\ P\ t\ \wedge\ P\ s\ \Rightarrow\ P\,(\mathsf{App}(t, s)) \\ \forall a\ t.\ P\ t\ \Rightarrow\ P\,(\mathsf{Lam}\,[a].t)\end{array}}{\forall t.\ P\ t}$$

Substitution Lemma: If $x \not\equiv y$ and $x \notin FV(L)$, then

$$M[x := N][y := L] \equiv M[y := L][x := N[y := L]].$$

Proof: By induction on the structure of M.

- **Case 1:** M is a variable.

  Case 1.1. $M \equiv x$. Then both sides equal $N[y := L]$ since $x \not\equiv y$.

  Case 1.2. $M \equiv y$. Then both sides equal L, for $x \notin FV(L)$ implies $L[x := \ldots] \equiv L$.

  Case 1.3. $M \equiv z \not\equiv x, y$. Then both sides equal z.

- **Case 2:** $M \equiv \lambda z.M_1$. By the variable convention we may assume that $z \not\equiv x, y$ and z is not free in N, L. Then by induction hypothesis

$$\begin{aligned}
&(\lambda z.M_1)[x := N][y := L] \\
\equiv\ &\lambda z.(M_1[x := N][y := L]) \\
\equiv\ &\lambda z.(M_1[y := L][x := N[y := L]]) \\
\equiv\ &(\lambda z.M_1)[y := L][x := N[y := L]].
\end{aligned}$$

- **Case 3:** $M \equiv M_1 M_2$. The statement follows again from the induction hypothesis.

" □ "

Induction over $\vdash$ for Free

$$\frac{\mathsf{valid}\,\Gamma \quad (a,\tau) \in \mathsf{set}\,\Gamma}{\Gamma \vdash \mathsf{Var}(a) : \tau} \qquad \frac{\Gamma \vdash t : \tau \to \sigma \quad \Gamma \vdash s : \tau}{\Gamma \vdash \mathsf{App}(t,s) : \sigma}$$

$$\frac{a \# \Gamma \quad (a,\tau) :: \Gamma \vdash t : \sigma}{\Gamma \vdash \mathsf{Lam}\,[a].t : \tau \to \sigma}$$

Induction Principle:

$$\forall \Gamma\, \tau\, a.\ \mathsf{valid}\,\Gamma \wedge (a,\tau) \in \mathsf{set}\,\Gamma \Rightarrow P\,\Gamma\,(\mathsf{Var}(a))\,\tau$$

$$\forall \Gamma\, t\, s\, \tau\, \sigma.\ \Gamma \vdash t : \tau \to \sigma \wedge P\,\Gamma\,t\,(\tau \to \sigma) \wedge \Gamma \vdash s : \tau \wedge P\,\Gamma\,s\,\tau \Rightarrow P\,\Gamma\,(\mathsf{App}(t,s))\,\sigma$$

$$\forall \Gamma\, a\, t\, \sigma\, \tau.\ a \# \Gamma \wedge ((a,\tau) :: \Gamma) \vdash t : \sigma \wedge P\,((a,\tau) :: \Gamma)\,t\,\sigma \Rightarrow P\,\Gamma\,(\mathsf{Lam}\,[a].t)\,(\tau \to \sigma)$$

$$\Gamma \vdash t : \tau \Rightarrow P\,\Gamma\,t\,\tau$$

Weakening Lemma

- Attempt to prove the Weakening-Lemma:

$$\Gamma_1 \vdash t : \tau \Rightarrow (\forall \Gamma_2.\, \text{valid}\, \Gamma_2 \Rightarrow\ \Gamma_1 \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \tau)$$

- **Lam-Case:**

$$\frac{\begin{array}{l}\forall \Gamma_3.\ \text{valid}\ \Gamma_3 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_3 \Rightarrow \Gamma_3 \vdash t : \sigma \\ \Gamma_1 \ll \Gamma_2 \\ \text{valid}\ \Gamma_2 \\ a \# \Gamma_1\end{array}}{\Gamma_2 \vdash \text{Lam}\ [a].t : \tau \to \sigma}$$

$$\frac{(a,\tau)::\Gamma \vdash t : \sigma \quad a \# \Gamma}{\Gamma \vdash \text{Lam}\ [a].t : \tau \to \sigma}$$

Can We Do Better? (I)

- structural induction for free:

$$\frac{\begin{array}{l}\forall a.\ P\,(\mathsf{Var}(a)) \\ \forall t\,s.\ P\,t\ \wedge\ P\,s\ \Rightarrow\ P\,(\mathsf{App}(t,s)) \\ \forall a\,t.\ P\,t\ \Rightarrow\ P\,(\mathsf{Lam}\,[a].t)\end{array}}{P\,t}$$

- stronger structural induction principle (where x is finitely supported):

$$\frac{\begin{array}{l}\forall a\,x.\ P\,x\,(\mathsf{Var}(a)) \\ \forall t\,s\,x.\ (\forall z.\ P\,z\,t) \wedge (\forall z.\ P\,z\,s) \Rightarrow P\,x\,(\mathsf{App}(t,s)) \\ \forall a\,t\,x.\ a \# x \wedge (\forall z.\ P\,z\,t) \Rightarrow P\,x\,(\mathsf{Lam}\,[a].t)\end{array}}{P\,x\,t}$$

Can We Do Better? (II)

- free induction principle for $\vdash$:

$$\frac{\begin{array}{l}\forall \Gamma\, \tau\, a.\ \mathsf{valid}\,\Gamma \wedge (a,\tau) \in \mathsf{set}\,\Gamma \Rightarrow P\,\Gamma\,(\mathsf{Var}(a))\,\tau \\ \forall \Gamma\, t\, s\, \tau\, \sigma.\ \Gamma \vdash t : \tau \to \sigma \wedge P\,\Gamma\, t\,(\tau \to \sigma) \wedge \\ \qquad \Gamma \vdash s : \tau \wedge P\,\Gamma\, s\,\tau \Rightarrow P\,\Gamma\,(\mathsf{App}(t,s))\,\sigma \\ \forall \Gamma\, a\, t\, \sigma\, \tau.\ a \# \Gamma \wedge ((a,\tau)::\Gamma) \vdash t : \sigma \wedge \\ \qquad P\,((a,\tau)::\Gamma)\, t\,\sigma \Rightarrow P\,\Gamma\,(\mathsf{Lam}\,[a].t)\,(\tau \to \sigma)\end{array}}{\Gamma \vdash t : \tau \Rightarrow P\,\Gamma\, t\,\tau}$$

Can We Do Better? (II)

- free induction principle for $\vdash$:

$$\frac{\begin{array}{l}\forall \Gamma\, \tau\, a\, x.\ \mathsf{valid}\, \Gamma \wedge (a, \tau) \in \mathsf{set}\, \Gamma \Rightarrow P\, x\, \Gamma\, (\mathsf{Var}(a))\, \tau \\ \forall \Gamma\, t\, s\, \tau\, \sigma\, x.\ \Gamma \vdash t : \tau \to \sigma \wedge (\forall z. P\, z\, \Gamma\, t\, (\tau \to \sigma)) \wedge \\ \qquad \Gamma \vdash s : \tau \wedge (\forall z. P\, z\, \Gamma\, s\, \tau) \Rightarrow P\, x\, \Gamma\, (\mathsf{App}(t, s))\, \sigma \\ \forall \Gamma\, a\, t\, \sigma\, \tau\, x.\ a \# x \wedge a \# \Gamma \wedge ((a, \tau) :: \Gamma) \vdash t : \sigma \wedge \\ (\forall z. P\, z\, ((a, \tau) :: \Gamma)\, t\, \sigma) \Rightarrow P\, x\, \Gamma\, (\mathsf{Lam}\, [a].t)\, (\tau \to \sigma)\end{array}}{\Gamma \vdash t : \tau \Rightarrow P\, x\, \Gamma\, t\, \tau}$$

where x is finitely supported and $\vdash$ needs to be equivariant, that is:

$$\Gamma \vdash t : \tau \Rightarrow (\pi \bullet \Gamma) \vdash (\pi \bullet t) : (\pi \bullet \tau)$$

How does this help?

- Instantiating the induction priciple:

$$\Gamma^? \vdash t^? : \tau^? \Rightarrow P^?\ x^?\ \Gamma^?\ t^?\ \tau^?$$

$$\Gamma_1 \vdash t : \tau \Rightarrow \text{valid}\ \Gamma_2 \Rightarrow\ \Gamma_1 \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \tau$$

$$P^? \mapsto \lambda \Gamma_2\, \Gamma_1\, t\, \tau.\ \text{valid}\ \Gamma_2 \Rightarrow \Gamma_1 \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \tau$$
$$x^? \mapsto \Gamma_2$$
$$\Gamma^? \mapsto \Gamma_1$$
$$t^? \mapsto t$$
$$\tau^? \mapsto \tau$$

$$\forall \Gamma_1\, a\, t\, \sigma\, \tau\, \Gamma_2.\ a \# \Gamma_2 \wedge a \# \Gamma_1 \wedge ((a, \tau) :: \Gamma_1) \vdash t : \sigma \wedge$$
$$(\forall \Gamma_3 . P\ \Gamma_3\ ((a, \tau) :: \Gamma_1)\ t\ \sigma) \Rightarrow P\ \Gamma_2\ \Gamma_1\ (\text{Lam}\ [a].t)\ (\tau \rightarrow \sigma)$$

How does this help?

- Instantiating the induction priciple:

$$\frac{\begin{array}{l} a \# \Gamma_2 \\ a \# \Gamma_1 \\ \forall \Gamma_3.\ \text{valid}\ \Gamma_3 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_3 \Rightarrow \Gamma_3 \vdash t : \sigma \\ \text{valid}\ \Gamma_2 \\ \Gamma_1 \ll \Gamma_2 \end{array}}{\Gamma_2 \vdash \text{Lam}\ [a].t : \tau \to \sigma}$$

$$\forall \Gamma_1\, a\, t\, \sigma\, \tau\, \Gamma_2.\ a \# \Gamma_2 \wedge a \# \Gamma_1 \wedge ((a,\tau)::\Gamma_1) \vdash t : \sigma \wedge (\forall \Gamma_3. P\, \Gamma_3\, ((a,\tau)::\Gamma_1)\, t\, \sigma) \Rightarrow P\, \Gamma_2\, \Gamma_1\, (\text{Lam}\ [a].t)\, (\tau \to \sigma)$$

How does this help?

- Instantiating the induction priciple:

$$\frac{\begin{array}{l} a \# \Gamma_2 \\ a \# \Gamma_1 \\ \forall \Gamma_3.\ \text{valid}\ \Gamma_3 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_3 \Rightarrow \Gamma_3 \vdash t : \sigma \\ \text{valid}\ \Gamma_2 \quad \text{gives} \quad \text{valid}\ ((a,\tau)::\Gamma_2) \\ \Gamma_1 \ll \Gamma_2 \end{array}}{\Gamma_2 \vdash \text{Lam}\ [a].t : \tau \to \sigma}$$

$$\forall \Gamma_1\, a\, t\, \sigma\, \tau\, \Gamma_2.\ a \# \Gamma_2 \wedge a \# \Gamma_1 \wedge ((a,\tau)::\Gamma_1) \vdash t : \sigma \wedge (\forall \Gamma_3.P\ \Gamma_3\ ((a,\tau)::\Gamma_1)\ t\ \sigma) \Rightarrow P\ \Gamma_2\ \Gamma_1\ (\text{Lam}\ [a].t)\ (\tau \to \sigma)$$

How does this help?

- Instantiating the induction priciple:

$$\frac{\begin{array}{l} a \# \Gamma_2 \\ a \# \Gamma_1 \\ \forall \Gamma_3.\ \text{valid}\ \Gamma_3 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_3 \Rightarrow \Gamma_3 \vdash t : \sigma \\ \text{valid}\ \Gamma_2 \quad \text{gives} \quad \text{valid}\ ((a,\tau)::\Gamma_2) \\ \Gamma_1 \ll \Gamma_2 \quad \text{gives} \quad ((a,\tau)::\Gamma_1) \ll ((a,\tau)::\Gamma_2) \end{array}}{\Gamma_2 \vdash \text{Lam}\ [a].t : \tau \to \sigma}$$

$$\forall \Gamma_1\, a\, t\, \sigma\, \tau\, \Gamma_2.\ a \# \Gamma_2 \wedge a \# \Gamma_1 \wedge ((a,\tau)::\Gamma_1) \vdash t : \sigma \wedge (\forall \Gamma_3. P\, \Gamma_3\, ((a,\tau)::\Gamma_1)\, t\, \sigma) \Rightarrow P\, \Gamma_2\, \Gamma_1\, (\text{Lam}\ [a].t)\, (\tau \to \sigma)$$

How does this help?

- Instantiating the induction priciple:

$$\frac{a \# \Gamma_2 \quad a \# \Gamma_1 \quad \forall \Gamma_3.\ \text{valid}\ \Gamma_3 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_3 \Rightarrow \Gamma_3 \vdash t : \sigma \quad \text{valid}\ \Gamma_2 \ \text{gives}\ \text{valid}\ ((a,\tau)::\Gamma_2) \quad \Gamma_1 \ll \Gamma_2 \ \text{gives}\ ((a,\tau)::\Gamma_1) \ll ((a,\tau)::\Gamma_2) \quad ((a,\tau)::\Gamma_2) \vdash t : \sigma}{\Gamma_2 \vdash \text{Lam}\,[a].t : \tau \to \sigma}$$

$$\forall \Gamma_1\, a\, t\, \sigma\, \tau\, \Gamma_2.\ a \# \Gamma_2 \wedge a \# \Gamma_1 \wedge ((a,\tau)::\Gamma_1) \vdash t : \sigma \wedge (\forall \Gamma_3. P\, \Gamma_3\, ((a,\tau)::\Gamma_1)\, t\, \sigma) \Rightarrow P\, \Gamma_2\, \Gamma_1\, (\text{Lam}\,[a].t)\, (\tau \to \sigma)$$

Variant that does not work

- This variant is not working (in general):

$$\forall \Gamma\, \tau\, a\, x.\ \mathrm{valid}\, \Gamma \wedge (a, \tau) \in \mathrm{set}\, \Gamma \Rightarrow P\, x\, \Gamma\, (\mathrm{Var}(a))\, \tau$$

$$\forall \Gamma\, t\, s\, \tau\, \sigma\, x.\ \Gamma \vdash t : \tau \to \sigma \wedge P\, x\, \Gamma\, t\, (\tau \to \sigma) \wedge \Gamma \vdash s : \tau \wedge P\, x\, \Gamma\, s\, \tau \Rightarrow P\, x\, \Gamma\, (\mathrm{App}(t, s))\, \sigma$$

$$\forall \Gamma\, a\, t\, \sigma\, \tau\, x.\ a \# x \wedge a \# \Gamma \wedge ((a, \tau) :: \Gamma) \vdash t : \sigma \wedge P\, x\, ((a, \tau) :: \Gamma)\, t\, \sigma \Rightarrow P\, x\, \Gamma\, (\mathrm{Lam}\, [a].t)\, (\tau \to \sigma)$$

$$\overline{\Gamma \vdash t : \tau \Rightarrow P\, x\, \Gamma\, t\, \tau}$$

- The induction principle for "free"

$$\forall \Gamma\, \tau\, a.\ \mathrm{valid}\, \Gamma \wedge (a, \tau) \in \mathrm{set}\, \Gamma \Rightarrow P\, \Gamma\, (\mathrm{Var}(a))\, \tau$$

$$\forall \Gamma\, t\, s\, \tau\, \sigma.\ \Gamma \vdash t : \tau \to \sigma \wedge P\, \Gamma\, t\, (\tau \to \sigma) \wedge \Gamma \vdash s : \tau \wedge P\, \Gamma\, s\, \tau \Rightarrow P\, \Gamma\, (\mathrm{App}(t, s))\, \sigma$$

$$\forall \Gamma\, a\, t\, \sigma\, \tau.\ a \# \Gamma \wedge ((a, \tau) :: \Gamma) \vdash t : \sigma \wedge P\, ((a, \tau) :: \Gamma)\, t\, \sigma \Rightarrow P\, \Gamma\, (\mathrm{Lam}\, [a].t)\, (\tau \to \sigma)$$

$$\overline{\Gamma \vdash t : \tau \Rightarrow P\, \Gamma\, t\, \tau}$$

E.g. Weakening Fails

- Lambda Case:

$$\frac{\begin{array}{l} a \# \Gamma_2 \\ a \# \Gamma_1 \\ \text{valid } \Gamma_2 \Rightarrow ((a,\tau)::\Gamma_1) \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \sigma \\ \text{valid } \Gamma_2 \\ \Gamma_1 \ll \Gamma_2 \end{array}}{\Gamma_2 \vdash \text{Lam } [a].t : \tau \to \sigma}$$

- attempt to fix it:

$$\Gamma_1 \vdash t : \tau \Rightarrow (\forall \Gamma_2.\, \text{valid}\, \Gamma_2 \Rightarrow\ \Gamma_1 \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \tau)$$

E.g. Weakening Fails

- Lambda Case:

$$a \# \Gamma_2$$

$$\frac{\begin{array}{l}\forall \Gamma\, \tau\, a\, x.\ \mathsf{valid}\,\Gamma \wedge (a,\tau) \in \mathsf{set}\,\Gamma \Rightarrow P\, x\, \Gamma\, (\mathsf{Var}(a))\, \tau \\ \forall \Gamma\, t\, s\, \tau\, \sigma\, x.\ \Gamma \vdash t : \tau \to \sigma \wedge P\, x\, \Gamma\, t\, (\tau \to \sigma) \wedge \\ \qquad \Gamma \vdash s : \tau \wedge P\, x\, \Gamma\, s\, \tau \Rightarrow P\, x\, \Gamma\, (\mathsf{App}(t,s))\, \sigma \\ \forall \Gamma\, a\, t\, \sigma\, \tau\, x.\ a \# x \wedge a \# \Gamma \wedge ((a,\tau) :: \Gamma) \vdash t : \sigma \wedge \\ \qquad P\, x\, ((a,\tau) :: \Gamma)\, t\, \sigma \Rightarrow P\, x\, \Gamma\, (\mathsf{Lam}\,[a].t)\, (\tau \to \sigma)\end{array}}{\Gamma \vdash t : \tau \Rightarrow P\, x\, \Gamma\, t\, \tau}$$

- attempt to fix it:

$$\Gamma_1 \vdash t : \tau \Rightarrow (\forall \Gamma_2.\ \mathsf{valid}\,\Gamma_2 \Rightarrow\ \Gamma_1 \ll \Gamma_2 \Rightarrow \Gamma_2 \vdash t : \tau)$$

Empirical Proven

- This version has been proved **in numerous instances** to be the one that is useful:

$$\forall \Gamma\, \tau\, a\, x.\ \mathsf{valid}\, \Gamma \wedge (a, \tau) \in \mathsf{set}\, \Gamma \Rightarrow P\, x\, \Gamma\, (\mathsf{Var}(a))\, \tau$$

$$\forall \Gamma\, t\, s\, \tau\, \sigma\, x.\ \Gamma \vdash t : \tau \to \sigma \wedge (\forall z. P\, z\, \Gamma\, t\, (\tau \to \sigma)) \wedge \Gamma \vdash s : \tau \wedge (\forall z. P\, z\, \Gamma\, s\, \tau) \Rightarrow P\, x\, \Gamma\, (\mathsf{App}(t, s))\, \sigma$$

$$\frac{\forall \Gamma\, a\, t\, \sigma\, \tau\, x.\ a \# x \wedge a \# \Gamma \wedge ((a, \tau) :: \Gamma) \vdash t : \sigma \wedge (\forall z. P\, z\, ((a, \tau) :: \Gamma)\, t\, \sigma) \Rightarrow P\, x\, \Gamma\, (\mathsf{Lam}\, [a].t)\, (\tau \to \sigma)}{\Gamma \vdash t : \tau \Rightarrow P\, x\, \Gamma\, t\, \tau}$$

where x is finitely supported.

Similarly

- Also this one seems to be the structural induction principle one wants:

$$\frac{\begin{array}{l}\forall a\, x.\ P\, x\, (\mathsf{Var}(a)) \\ \forall t\, s\, x.\ (\forall z.\ P\, z\, t) \wedge (\forall z.\ P\, z\, s) \Rightarrow P\, x\, (\mathsf{App}(t, s)) \\ \forall a\, t\, x.\ a \# x \wedge (\forall z.\ P\, z\, t) \Rightarrow P\, x\, (\mathsf{Lam}\, [a].t)\end{array}}{P\, x\, t}$$

where x is finitely supported.

Proof

By structural induction (weak version) over t.
We need to strengthen the i.h. to

$$\forall \pi\, x.\ P\, x\, (\pi \bullet t)$$

In the lambda-case we need to show:

$$P\, x\, (\pi \bullet \mathsf{Lam}\, [a].t)$$

We have:

$$\forall \pi\, x.\ P\, x\, (\pi \bullet t)$$

$$\forall a\, t\, x.\ a \# x \Rightarrow (\forall z.\ P\, z\, t) \Rightarrow P\, x\, (\mathsf{Lam}\, [a].t)$$

Proof (II)

We have:

$$\forall \pi\, x.\ P\, x\, (\pi \bullet t)$$

$$\forall a\, t\, x.\ a \# x \Rightarrow (\forall z.\ P\, z\, t) \Rightarrow P\, x\, (\mathsf{Lam}\, [a].t)$$

We choose a c so that $c \# (x, \pi \bullet a, \pi \bullet t)$.
Hence $\forall t.\ (\forall z.\ P\, z\, t) \Rightarrow P\, x\, (\mathsf{Lam}\, [c].t)$.
Moreover $\forall x.\ P\, x\, ((c\ \ \pi \bullet a) :: \pi \bullet t)$.
Therefore $P\, x\, (\mathsf{Lam}\, [c].((c\ \ \pi \bullet a) :: \pi \bullet t))$.
Now

$$\mathsf{Lam}\, [c].((c\, \pi \bullet a) :: \pi \bullet t) = \mathsf{Lam}\, [\pi \bullet a].(\pi \bullet t).$$

So we have $P\, x\, (\pi \bullet \mathsf{Lam}\, [a].t)$. Done.

```
lemma lam_induct:
fixes P:: "('a :: fs_name) ⇒ lam ⇒ bool"
assumes h1: "⋀ a x.P x (Var(a))"
      and h2: "⋀ t s x. (⋀ z.P z t) ⟹ (⋀ z.P z s) ⟹ P x (App t s)"
      and h3: "⋀ a t x. a # x ⟹ (⋀ z.P z t) ⟹ P x (Lam [a].t)"
shows "P x t"
proof -
  have "∀(π::name prm) x. P x (π • t)"
  proof (induct t rule: lam.induct_weak)
    case (Lam a t)
    have ih: "∀(π::name prm) x. P x (π • t)" by fact
    show "∀(π::name prm) x. P x (π • (Lam [a].t))"
    proof (intro strip)
      fix π::"name prm" and x::"'a :: fs_name"
      have "∃c :: name. c # (π • a, x, π • t)"
        by (rule exists_fresh, simp add: supp_prod fs_name)
      then obtain c::"name"
        where f1: "c ≠ π • a" and f2: "c # x" and f3: "c # π • t"
        by (force simp add: fresh_prod fresh_atm)
```

```
    from ih have "∀x. P x (([(c, π • a)]@π) • t)" by force
    hence "∀x. P x ([(c, π • a)] • (π • t))"
      by (simp add: pt_name2[symmetric])
    hence "P x (Lam [c].([(c, π • a)] • (π • t)))" using h3 f2by simp
    moreover
    have "Lam [c].([(c, π • a)] • (π • t)) = Lam [π • a].(π • t)"
      using f1 f3 by (simp add: lam.inject alpha)
    ultimately show "P x (π • Lam [a].t)" by simp
  qed
 qed (auto intro: h1 h2)
 hence "P x (([] :: name prm) • t)" by blast
 thus "P x t" by simp
qed
```

Conclusion

- These stronger induction principles work jolly good in practise.
- Tomorrow we will have a look at the "bigger picture".