POPLmark, nominal everywhere, in Isabelle

Christian Urban, Benjamin Pierce, Dimitrios Vytiniotis,
Stephanie Weirich and Steve Zdancewic

January 11, 2006

Abstract

We give a solution to Part IA of the POPLmark-challenge using the *nominal datatype package*. This package has still a number of shortcomings, which manifest themselves for example in the lack of a recursion combinator for defining functions. Nevertheless the point of this solution is that by using the nominal datatype package some proofs of the POPLmark-challenge can be formalised following quite closely the informal reasoning from the POPLmark-paper.

Contents

Authors: Christian Urban, Benjamin Pierce, Dimitrios Vytiniotis Stephanie Weirich and Steve Zdancewic

with great help from Stefan Berghofer and Markus Wenzel.

1 Types for Names, Nominal Datatype Declaration for Types and Terms

The main point of this solution is to use names everywhere (be they bound, binding or free). In System $F_{<:}$ there are two kinds of names corresponding to type-variables and to term-variables. These two kinds of names are represented in the nominal datatype package as atom-types *tyvrs* and *vrs*:

atom-decl *tyvrs vrs*

There are numerous facts that come with this declaration: for example that there are infinitely many elements in *tyvrs* and *vrs*.

The constructors for types and terms in System $F_{<:}$ contain abstractions over type-variables and term-variables. The nominal datatype-package uses $\ll\ldots\gg\ldots$ to indicate where abstractions occur.

nominal-datatype *ty* =
  Tvar tyvrs
| *Top*
| *Arrow ty ty* (- → - [*100*,*100*] *100*)
| *Forall* ≪*tyvrs*≫*ty ty*

nominal-datatype *trm* =
  Var vrs
| *Lam* ≪*vrs*≫*trm ty*
| *Tabs* ≪*tyvrs*≫*trm ty*
| *App* *trm trm*
| *Tapp* *trm ty*

To be polite to the eye, some more familiar notation is introduced. Because of the change in the order of arguments, one needs to use translation rules, instead of syntax annotations at the term-constructors as given above for *Arrow*.

syntax
Forall-syn :: *tyvrs* ⇒ *ty* ⇒ *ty* ⇒ *ty* (∀ [-<:-].- [*100*,*100*,*100*] *100*)
Lam-syn :: *vrs* ⇒ *ty* ⇒ *trm* ⇒ *trm* (*Lam* [-:-].- [*100*,*100*,*100*] *100*)
Tabs-syn :: *tyvrs* ⇒ *ty* ⇒ *trm* ⇒ *trm* (*Tabs* [-<:-].- [*100*,*100*,*100*] *100*)

translations
$\forall [X{<:}T_1].T_2 \rightleftharpoons ty.Forall\ X\ T_2\ T_1$
$Lam\ [x{:}T].t \rightleftharpoons trm.Lam\ x\ t\ T$
$Tabs\ [X{<:}T].t \rightleftharpoons trm.Tabs\ X\ t\ T$

Again there are numerous facts that are proved automatically for *ty* and *trm*: for example that the set of free variables, i.e. the *support*, is finite. However note that nominal-datatype declarations do *not* define "classical" constructor-based datatypes, but rather define α-equivalence classes—we can for example show that α-equivalent *ty*s and *trm*s are equal:

lemma *alpha-illustration*:

shows $\forall[X{<:}T].(Tvar\ X) = \forall[Y{<:}T].(Tvar\ Y)$
and $Lam\ [x{:}T].(Var\ x) = Lam\ [y{:}T].(Var\ y)$
by (*simp-all add*: *ty.inject trm.inject alpha calc-atm fresh-atm*)

2 SubTyping Contexts

types $ty\text{-}context = (tyvrs \times ty)\ list$

Typing contexts are represented as lists that "grow" on the left; we thereby deviating from the convention in the POPLmark-paper. The lists contain pairs of type-variables and types (this is sufficient for Part 1A).

In order to state validity-conditions for typing-contexts, the notion of a *domain* of a typing-context is handy.

consts
$domain :: ty\text{-}context \Rightarrow tyvrs\ set$
primrec
$domain\ [] = \{\}$
$domain\ (X{::}\Gamma) = \{fst\ X\} \cup (domain\ \Gamma)$

lemma *domain-eqvt*:
fixes $pi{::}tyvrs\ prm$
shows $pi{\cdot}(domain\ \Gamma) = domain\ (pi{\cdot}\Gamma)$
by (*induct* Γ, *simp-all add*: *perm-empty perm-insert perm-fst*)

lemma *finite-domain*:
shows $finite\ (domain\ \Gamma)$
by (*induct* Γ, *auto*)

lemma *domain-supp*:
shows $(supp\ (domain\ \Gamma)) = (domain\ \Gamma)$
by (*simp only*: *at-fin-set-supp at-tyvrs-inst finite-domain*)

lemma *domain-inclusion*:
assumes a: $(X,T) \in set\ \Gamma$
shows $X \in (domain\ \Gamma)$
using a **by** (*induct* Γ, *auto*)

lemma *domain-existence*:
assumes a: $X \in (domain\ \Gamma)$
shows $\exists\ T.(X,T) \in set\ \Gamma$
using a **by** (*induct* Γ, *auto*)

lemma *domain-append*:
shows $domain\ (\Gamma @ \Delta) = ((domain\ \Gamma) \cup (domain\ \Delta))$
by (*induct* Γ, *auto*)

lemma *fresh-domain-cons*:
fixes $X{::}tyvrs$

shows $X\#(domain\ (Y{::}\Gamma)) = (X\#(fst\ Y) \wedge X\#(domain\ \Gamma))$
by (*simp add*: *fresh-fin-insert pt-tyvrs-inst at-tyvrs-inst fs-tyvrs-inst finite-domain*)

lemma *fresh-domain*:
fixes $X{::}tyvrs$
assumes a: $X\#\Gamma$
shows $X\#(domain\ \Gamma)$
using a
apply(*induct* Γ)
apply(*simp add*: *fresh-set-empty*)
apply(*simp only*: *fresh-domain-cons*)
apply(*auto simp add*: *fresh-prod fresh-list-cons*)
done

Not all lists of type *ty-context* are well-formed. One condition requires that in $(X, S) :: \Gamma$ all free variables of S must be in the *domain* of Γ, that is S must be *closed* in Γ. The set of free variables of S is the *support* of S.

constdefs
closed-in :: $ty \Rightarrow ty\text{-}context \Rightarrow bool$ (- *closed'-in* - [100,100] 100)
S *closed-in* $\Gamma \equiv (supp\ S) \subseteq (domain\ \Gamma)$

lemma *closed-in-eqvt*:
fixes $pi{::}tyvrs\ prm$
assumes a: S *closed-in* Γ
shows $(pi{\cdot}S)$ *closed-in* $(pi{\cdot}\Gamma)$
using a
proof (*unfold closed-in-def*)
assume $supp\ S \subseteq (domain\ \Gamma)$
hence $pi{\cdot}(supp\ S) \subseteq pi{\cdot}(domain\ \Gamma)$
by (*simp add*: *pt-subseteq-eqvt*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
thus $(supp\ (pi{\cdot}S)) \subseteq (domain\ (pi{\cdot}\Gamma))$
by (*simp add*: *domain-eqvt pt-perm-supp*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
qed

Now validity of a context is a straightforward inductive definition.

consts
valid-rel :: *ty-context set*
valid-sym :: $ty\text{-}context \Rightarrow bool$ ($\vdash$ - *ok* [100] 100)
translations
$\vdash \Gamma$ *ok* $\rightleftharpoons \Gamma \in$ *valid-rel*
inductive *valid-rel*
intros
valid-nil[*simp*]: $\vdash []$ *ok*
valid-cons[*simp*]: $[\![\vdash \Gamma\ ok;\ X\#(domain\ \Gamma);\ T\ closed\text{-}in\ \Gamma]\!] \Longrightarrow\ \vdash ((X,T){::}\Gamma)\ ok$

lemma *valid-eqvt*:
fixes $pi{::}tyvrs\ prm$
assumes a: $\vdash \Gamma$ *ok*
shows $\vdash (pi{\cdot}\Gamma)$ *ok*

using *a*
proof (*induct*)
case *valid-nil*
show $\vdash (pi\cdot[])$ *ok* **by** *simp*
next
case (*valid-cons* T X Γ)
have $\vdash (pi\cdot\Gamma)$ *ok* **by** *fact*
moreover
have $X\#(domain\ \Gamma)$ **by** *fact*
hence $(pi\cdot X)\#(domain\ (pi\cdot\Gamma))$ **by** (*simp add*: *fresh-eqvt domain-eqvt*[*symmetric*])
moreover
have T *closed-in* Γ **by** *fact*
hence $(pi\cdot T)$ *closed-in* $(pi\cdot\Gamma)$ **by** (*simp add*: *closed-in-eqvt*)
ultimately show $\vdash (pi\cdot((X,T)::\Gamma))$ *ok* **by** *simp*
qed

lemma *validE*:
assumes *a*: $\vdash ((X,T)::\Gamma)$ *ok*
shows $\vdash \Gamma$ *ok* $\wedge$ $X\#(domain\ \Gamma)$ $\wedge$ T *closed-in* Γ
using *a* **by** (*cases*, *auto*)

lemma *validE-append*:
assumes *a*: $\vdash (\Delta@\Gamma)$ *ok*
shows $\vdash \Gamma$ *ok*
using *a* **by** (*induct* Δ, *auto dest*: *validE*)

lemma *replace-type*:
assumes *a*: $\vdash (\Delta@(X,T)::\Gamma)$ *ok*
and *b*: S *closed-in* Γ
shows $\vdash (\Delta@(X,S)::\Gamma)$ *ok*
using *a b*
apply(*induct* Δ)
apply(*auto dest*!: *validE intro*!: *valid-cons simp add*: *domain-append closed-in-def*)
done

Well-formed contexts have a unique type-binding for a type-variable.

lemma *uniqueness-of-ctxt*:
fixes Γ::*ty-context*
assumes *a*: $\vdash \Gamma$ *ok*
and *b*: $(X,T)\in set\ \Gamma$
and *c*: $(X,S)\in set\ \Gamma$
shows $T=S$
using *a b c*
proof (*induct*)
case *valid-nil* **thus** $T=S$ **by** *simp*
next
case (*valid-cons* U Y Γ)
moreover
{ **fix** Γ::*ty-context*
assume *a*: $X\#(domain\ \Gamma)$

have $\neg(\exists\, T.(X,T) \in (set\ \Gamma))$ **using** a
proof $(induct\ \Gamma)$
case $(Cons\ Y\ \Gamma)$
thus $\neg\ (\exists\, T.(X,T) \in set(Y{::}\Gamma))$
by $(simp\ only\colon fresh\text{-}domain\text{-}cons,\ auto\ simp\ add\colon fresh\text{-}atm)$
qed $(simp)$
}
ultimately show $T{=}S$ **by** *auto*
qed

3 Size and Capture-Avoiding Substitution for Types

In the current version of the nominal datatype package even simple functions (such as size of types and capture-avoiding substitution) can only be defined manually via some laborious proof constructions. Therefore we just assume them here. Cheat!

consts $size\text{-}ty :: ty \Rightarrow nat$

lemma $size\text{-}ty[simp]$:
shows $size\text{-}ty\ (Tvar\ X) = 1$
and $size\text{-}ty\ (Top) = 1$
and $[\![size\text{-}ty\ T_1 = m;\ size\text{-}ty\ T_2 = n]\!] \Longrightarrow size\text{-}ty\ (T_1 \rightarrow T_2) = m + n + 1$
and $[\![size\text{-}ty\ T_1 = m;\ size\text{-}ty\ T_2 = n]\!] \Longrightarrow size\text{-}ty\ (\forall\,[X{<:}T_1].T_2) = m + n + 1$
sorry

consts $subst\text{-}ty :: ty \Rightarrow tyvrs \Rightarrow ty \Rightarrow ty\ (\text{-}[\text{-}{:=}\text{-}]_{ty}\ [100,100,100]\ 100)$

lemma $subst\text{-}ty[simp]$:
shows $(Tvar\ X)[Y{:=}T]_{ty} = (if\ X{=}Y\ then\ T\ else\ (Tvar\ X))$
and $(Top)[Y{:=}T]_{ty} = Top$
and $(T_1 \rightarrow T_2)[Y{:=}T]_{ty} = (T_1[Y{:=}T]_{ty}) \rightarrow (T_2[Y{:=}T]_{ty})$
and $X \sharp (Y,T) \Longrightarrow (\forall\,[X{<:}T_1].T_2)[Y{:=}T]_{ty} = (\forall\,[X{<:}(T_1[Y{:=}T]_{ty})].(T_2[Y{:=}T]_{ty}))$
and $[\![X \sharp Y;\ X \sharp T]\!] \Longrightarrow (\forall\,[X{<:}T_1].T_2)[Y{:=}T]_{ty} = (\forall\,[X{<:}(T_1[Y{:=}T]_{ty})].(T_2[Y{:=}T]_{ty}))$
sorry

consts $subst\text{-}tyc :: ty\text{-}context \Rightarrow tyvrs \Rightarrow ty \Rightarrow ty\text{-}context\ (\text{-}[\text{-}{:=}\text{-}]_{tyc}\ [100,100,100]\ 100)$
primrec
$([])[Y{:=}T]_{tyc} = []$
$(XT{::}\Gamma)[Y{:=}T]_{tyc} = (fst\ XT,(snd\ XT)[Y{:=}T]_{ty}){::}(\Gamma[Y{:=}T]_{tyc})$

4 Subtyping-Relation

The definition for the subtyping-relation follows quite closely what is written in the POPLmark-paper, except for the premises dealing with well-formed contexts and the freshness constraint $X \sharp \Gamma$ in the *S-Forall*-rule. (The freshness constraint is specific to the *nominal approach*. Note, however, that the constraint does *not* make the subtyping-relation "partial"... because we work over α-equivalence classes.)

consts

subtype-of :: $(\textit{ty-context} \times \textit{ty} \times \textit{ty})$ *set*
syntax
subtype-of-syn :: $\textit{ty-context} \Rightarrow \textit{ty} \Rightarrow \textit{ty} \Rightarrow \textit{bool}$ $(\text{-}\vdash\text{-<:-}\ [100,100,100]\ 100)$

translations
$\Gamma \vdash S <: T \rightleftharpoons (\Gamma,S,T) \in \textit{subtype-of}$
inductive *subtype-of*
intros
S-Top[*intro*]: $[\![\vdash \Gamma\ \textit{ok};\ S\ \textit{closed-in}\ \Gamma]\!] \Longrightarrow \Gamma \vdash S <: \textit{Top}$
S-Var[*intro*]: $[\![(X,S) \in \textit{set}\ \Gamma;\ \Gamma \vdash S <: T]\!] \Longrightarrow \Gamma \vdash (\textit{Tvar}\ X) <: T$
S-Refl[*intro*]: $[\![\vdash \Gamma\ \textit{ok};\ X \in \textit{domain}\ \Gamma]\!] \Longrightarrow \Gamma \vdash \textit{Tvar}\ X <: \textit{Tvar}\ X$
S-Arrow[*intro*]: $[\![\Gamma \vdash T_1 <: S_1;\ \Gamma \vdash S_2 <: T_2]\!] \Longrightarrow \Gamma \vdash (S_1 \to S_2) <: (T_1 \to T_2)$
S-Forall[*intro*]: $[\![\Gamma \vdash T_1 <: S_1;\ X \sharp \Gamma;\ ((X,T_1)::\Gamma) \vdash S_2 <: T_2]\!]$
$\Longrightarrow \Gamma \vdash \forall[X<:S_1].S_2 <: \forall[X<:T_1].T_2$

lemma *subtype-implies-closed*:
assumes *a*: $\Gamma \vdash S <: T$
shows S *closed-in* $\Gamma \wedge T$ *closed-in* Γ
using *a*
proof (*induct*)
case (*S-Top S* Γ)
have *Top closed-in* Γ **by** (*simp add*: *closed-in-def ty.supp*)
moreover
have S *closed-in* Γ **by** *fact*
ultimately show S *closed-in* $\Gamma \wedge$ *Top closed-in* Γ **by** *simp*
next
case (*S-Var S T X* Γ)
have $(X,S) \in \textit{set}\ \Gamma$ **by** *fact*
hence $X \in \textit{domain}\ \Gamma$ **by** (*rule domain-inclusion*)
hence (*Tvar X*) *closed-in* Γ **by** (*simp add*: *closed-in-def ty.supp supp-atm*)
moreover
have S *closed-in* $\Gamma \wedge T$ *closed-in* Γ **by** *fact*
hence T *closed-in* Γ **by** *force*
ultimately show (*Tvar X*) *closed-in* $\Gamma \wedge T$ *closed-in* Γ **by** *simp*
qed (*auto simp add*: *closed-in-def ty.supp supp-atm abs-supp*)

lemma *subtype-implies-ok*:
fixes X::*tyvrs*
assumes *a1*: $\Gamma \vdash S <: T$
shows $\vdash \Gamma$ *ok*
using *a1* **by** (*induct*, *auto*)

lemma *subtype-implies-fresh*:
fixes X::*tyvrs*
assumes *a1*: $\Gamma \vdash S <: T$
and *a2*: $X \sharp \Gamma$
shows $X \sharp S \wedge X \sharp T$
proof −
from *a1* **have** $\vdash \Gamma$ *ok* **by** (*rule subtype-implies-ok*)
with *a2* **have** $X \sharp \textit{domain}(\Gamma)$ **by** (*simp add*: *fresh-domain*)

moreover
from *a1* **have** S *closed-in* $\Gamma \wedge T$ *closed-in* Γ **by** (*rule subtype-implies-closed*)
hence *supp* $S \subseteq$ ((*supp* (*domain* Γ))::*tyvrs set*)
and *supp* $T \subseteq$ ((*supp* (*domain* Γ))::*tyvrs set*) **by** (*simp-all add*: *domain-supp closed-in-def*)
ultimately show $X \# S \wedge X \# T$ **by** (*force simp add*: *supp-prod fresh-def*)
qed

lemma *subtype-eqvt*:
fixes *pi*::*tyvrs prm*
shows $\Gamma \vdash S <: T \Longrightarrow (pi{\cdot}\Gamma) \vdash (pi{\cdot}S) <: (pi{\cdot}T)$
apply(*erule subtype-of.induct*)
apply(*force intro*: *valid-eqvt closed-in-eqvt*)
apply(*force intro*!: *S-Var*
intro: *closed-in-eqvt valid-eqvt*
dest: *pt-set-bij2*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*]
simp add: *pt-list-set-pi*[*OF pt-tyvrs-inst*, *symmetric*])
apply(*force intro*: *valid-eqvt*
dest: *pt-set-bij2*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*]
simp add: *domain-eqvt*)
apply(*force*)
apply(*force intro*!: *S-Forall simp add*: *fresh-prod fresh-eqvt*)
done

The most painful part of the subtyping definition is the strengthening of the induction principle. The induction principle that comes for free with the definition of *subtype-of* requires to prove the *S-Forall*-case for *all* binders *X*. This will often cause some renaming-manoeuvres in the reasoning. To avoid this, we strengthening the induction-principle so that we only have to establish the *S-Forall*-case for *fresh* binders *X*. The property that allows us to strengthen the induction-principle is that the equivariance of *subtype-of*.

lemma *subtype-of-induct*[*consumes 1*, *case-names S-Top S-Var S-Refl S-Arrow S-Forall*]:
fixes P :: '*a*::*fs-tyvrs*$\Rightarrow$*ty-context* $\Rightarrow$ *ty* $\Rightarrow$ *ty* $\Rightarrow$*bool*
assumes *a*: $\Gamma \vdash S <: T$
and *a1*: $\bigwedge\Gamma\ S\ x.\ [\![\vdash \Gamma\ ok;\ S\ closed\text{-}in\ \Gamma]\!] \Longrightarrow P\ x\ \Gamma\ S\ Top$
and *a2*: $\bigwedge\Gamma\ X\ S\ T\ x.\ [\![(X,S){\in}set\ \Gamma;\ \Gamma \vdash S <: T;\ \bigwedge z.\ P\ z\ \Gamma\ S\ T]\!] \Longrightarrow P\ x\ \Gamma\ (Tvar\ X)\ T$
and *a3*: $\bigwedge\Gamma\ X\ x.\ [\![\vdash \Gamma\ ok;\ X{\in}domain\ \Gamma]\!] \Longrightarrow P\ x\ \Gamma\ (Tvar\ X)\ (Tvar\ X)$
and *a4*: $\bigwedge\Gamma\ S_1\ S_2\ T_1\ T_2\ x.$
$[\![\Gamma \vdash T_1 <: S_1;\ \bigwedge z.\ P\ z\ \Gamma\ T_1\ S_1;\ \Gamma \vdash S_2 <: T_2;\ \bigwedge z.\ P\ z\ \Gamma\ S_2\ T_2]\!]$
$\Longrightarrow P\ x\ \Gamma\ (S_1 \rightarrow S_2)\ (T_1 \rightarrow T_2)$
and *a5*: $\bigwedge\Gamma\ X\ S_1\ S_2\ T_1\ T_2\ x.$
$[\![X \# x;\ X \# (\Gamma,S_1,T_1);\ \Gamma \vdash T_1 <: S_1;\ \bigwedge z.\ P\ z\ \Gamma\ T_1\ S_1;$
$((X,T_1)::\Gamma) \vdash S_2 <: T_2;\ \bigwedge z.\ P\ z\ ((X,T_1)::\Gamma)\ S_2\ T_2]\!]$
$\Longrightarrow P\ x\ \Gamma\ (\forall [X{<:}S_1].S_2)\ (\forall [X{<:}T_1].T_2)$
shows $P\ x\ \Gamma\ S\ T$
proof −
from *a* **have** $\bigwedge$(*pi*::*tyvrs prm*) (*x*::'*a*::*fs-tyvrs*). $P\ x\ (pi{\cdot}\Gamma)\ (pi{\cdot}S)\ (pi{\cdot}T)$
proof (*induct*)
case (*S-Top S* Γ)
thus $P\ x\ (pi{\cdot}\Gamma)\ (pi{\cdot}S)\ (pi{\cdot}Top)$ **by** (*force intro*: *valid-eqvt closed-in-eqvt a1*)

```
next
  case (S-Var S T X Γ)
  have (X,S) ∈ set Γ by fact
  hence pi•(X,S)∈pi•(set Γ) by (rule pt-set-bij2[OF pt-tyvrs-inst, OF at-tyvrs-inst])
  hence (pi•X,pi•S)∈set (pi•Γ) by (simp add: pt-list-set-pi[OF pt-tyvrs-inst])
  moreover
  have Γ ⊢ S <: T by fact
  hence (pi•Γ) ⊢ (pi•S) <: (pi•T) by (rule subtype-eqvt)
  moreover
  have ⋀(pi::tyvrs prm) x. P x (pi•Γ) (pi•S) (pi•T) by fact
  hence ⋀x. P x (pi•Γ) (pi•S) (pi•T) by simp
  ultimately
  show P x (pi•Γ) (pi•(Tvar X)) (pi•T) by (simp add: a2)
next
  case (S-Refl X Γ)
  have ⊢ Γ ok by fact
  hence ⊢ (pi•Γ) ok by (rule valid-eqvt)
  moreover
  have X ∈ domain Γ by fact
  hence (pi•X)∈pi•domain Γ by (rule pt-set-bij2[OF pt-tyvrs-inst, OF at-tyvrs-inst])
  hence (pi•X)∈domain (pi•Γ) by (simp add: domain-eqvt pt-list-set-pi[OF pt-tyvrs-inst])
  ultimately show P x (pi•Γ) (pi•(Tvar X)) (pi•(Tvar X)) by (simp add: a3)
next
  case S-Arrow thus ?case by (auto intro: a4 subtype-eqvt)
next
  case (S-Forall S1 S2 T1 T2 X Γ)
  have b1: Γ ⊢ T1 <: S1 by fact
  have b2: ⋀(pi::tyvrs prm) x. P x (pi•Γ) (pi•T1) (pi•S1) by fact
  have b4: ((X,T1)::Γ) ⊢ S2 <: T2 by fact
  have b5: ⋀(pi::tyvrs prm) x. P x (pi•((X,T1)::Γ)) (pi•S2) (pi•T2) by fact
  have b3: X♯Γ by fact
  have b3': X♯T1 X♯S1 using b1 b3 by (simp-all add: subtype-implies-fresh)
  have ∃C::tyvrs. C♯(pi•X,pi•S2,pi•T2,pi•S1,pi•T1,pi•Γ,x)
    by (rule at-exists-fresh[OF at-tyvrs-inst], simp add: fs-tyvrs1)
  then obtain C::tyvrs where  f1: C≠(pi•X) and f2: C♯(pi•S1) and f3: C♯(pi•T1)
    and f4: C♯(pi•S2) and f5: C♯(pi•T2) and f6: C♯(pi•Γ) and f7: C♯x
    by (auto simp add: fresh-prod fresh-atm)
  let ?pi' = [(C,pi•X)]@pi
  from b2 have c1: ⋀x. P x (?pi'•Γ) (?pi'•T1) (?pi'•S1) by simp
  from b5 have ⋀x. P x (?pi'•((X,T1)::Γ)) (?pi'•S2) (?pi'•T2) by simp
  hence ⋀x. P x ((?pi'•X,?pi'•T1)::(?pi'•Γ)) (?pi'•S2) (?pi'•T2) by simp
  hence c2: ⋀x. P x ((C,?pi'•T1)::(?pi'•Γ)) (?pi'•S2) (?pi'•T2) using f1
    by (simp only: pt-tyvrs2 calc-atm, simp)
  from b3 have (pi•X)♯(pi•Γ)
    by (simp add: fresh-prod pt-fresh-bij[OF pt-tyvrs-inst, OF at-tyvrs-inst])
  with f6 have f6a: ?pi'•Γ=pi•Γ
    by (simp only: pt-tyvrs2 pt-fresh-fresh[OF pt-tyvrs-inst, OF at-tyvrs-inst])
  hence f6': C♯(?pi'•Γ) using f6 by simp
  from b3' have (pi•X)♯(pi•S1)
    by (simp add: fresh-prod pt-fresh-bij[OF pt-tyvrs-inst, OF at-tyvrs-inst])
```

with *f2* **have** *f2a*: *?pi'*·*S1*=*pi*·*S1*
  by (*simp only*: *pt-tyvrs2 pt-fresh-fresh*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
hence *f2'*: *C*♯(*?pi'*·*S1*) **using** *f2* **by** *simp*
from *b3'* **have** (*pi*·*X*)♯(*pi*·*T1*)
  by (*simp add*: *fresh-prod pt-fresh-bij*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
with *f3* **have** *f3a*: *?pi'*·*T1*=*pi*·*T1*
  by (*simp only*: *pt-tyvrs2 pt-fresh-fresh*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
hence *f3'*: *C*♯(*?pi'*·*T1*) **using** *f3* **by** *simp*
from *b1* **have** *e1*: (*?pi'*·Γ) ⊢ (*?pi'*·*T1*) <: (*?pi'*·*S1*) **by** (*rule subtype-eqvt*)
from *b4* **have** (*?pi'*·((*X*,*T1*)::Γ)) ⊢ (*?pi'*·*S2*) <: (*?pi'*·*T2*) **by** (*rule subtype-eqvt*)
hence ((*?pi'*·*X*,*?pi'*·*T1*)::(*?pi'*·Γ)) ⊢ (*?pi'*·*S2*) <: (*?pi'*·*T2*) **by** *simp*
hence *e2*: ((*C*,*?pi'*·*T1*)::(*?pi'*·Γ)) ⊢ (*?pi'*·*S2*) <: (*?pi'*·*T2*) **using** *f1*
  by (*simp only*: *pt-tyvrs2 calc-atm*, *simp*)
have *fnew*: *C*♯(*?pi'*·Γ,*?pi'*·*S1*,*?pi'*·*T1*) **using** *f6' f2' f3'* **by** (*simp add*: *fresh-prod*)
have *main*: *P x* (*?pi'*·Γ) (∀[*C*<:(*?pi'*·*S1*)].(*?pi'*·*S2*)) (∀[*C*<:(*?pi'*·*T1*)].(*?pi'*·*T2*))
  using *f7 fnew e1 c1 e2 c2* **by** (*rule a5*)
have *alpha1*: (∀[*C*<:(*?pi'*·*S1*)].(*?pi'*·*S2*)) = (*pi*·(∀[*X*<:*S1*].*S2*))
  using *f1 f4 f2a*[*symmetric*] **by** (*simp add*: *ty.inject alpha pt-tyvrs2*[*symmetric*])
have *alpha2*: (∀[*C*<:(*?pi'*·*T1*)].(*?pi'*·*T2*)) = (*pi*·(∀[*X*<:*T1*].*T2*))
  using *f1 f5 f3a*[*symmetric*] **by** (*simp add*: *ty.inject alpha pt-tyvrs2*[*symmetric*])
show *P x* (*pi*·Γ) (*pi*·(∀[*X*<:*S1*].*S2*)) (*pi*·(∀[*X*<:*T1*].*T2*))
  using *alpha1 alpha2 f6a main* **by** *simp*
qed
hence *P x* (([]::*tyvrs prm*)·Γ) (([]::*tyvrs prm*)·*S*) (([]::*tyvrs prm*)·*T*) **by** *blast*
thus *?thesis* **by** *simp*
qed

5 Reflexivity of Subtyping

lemma *subtype-reflexivity*:
  assumes *a*: ⊢ Γ *ok*
  and *b*: *T closed-in* Γ
  shows Γ ⊢ *T* <: *T*
using *a b*
proof(*nominal-induct T avoiding*: Γ *rule*: *ty.induct*)
  case (*Forall X* T_1 T_2)
  have *ih*-T_1: ⋀Γ. ⊢ Γ *ok* ⟹ T_1 *closed-in* Γ ⟹ Γ ⊢ T_1 <: T_1 **by** *fact*
  have *ih*-T_2: ⋀Γ. ⊢ Γ *ok* ⟹ T_2 *closed-in* Γ ⟹ Γ ⊢ T_2 <: T_2 **by** *fact*
  have *fresh-cond*: *X*♯Γ **by** *fact*
  hence *fresh-domain*: *X*♯(*domain* Γ) **by** (*simp add*: *fresh-domain*)
  have (∀[*X*<:T_2].T_1) *closed-in* Γ **by** *fact*
  hence *closed*$_{T2}$: T_2 *closed-in* Γ **and** *closed*$_{T1}$: T_1 *closed-in* ((*X*,T_2)::Γ)
    by (*auto simp add*: *closed-in-def ty.supp abs-supp*)
  have *ok*: ⊢ Γ *ok* **by** *fact*
  hence *ok'*: ⊢ ((*X*,T_2)::Γ) *ok* **using** *closed*$_{T2}$ *fresh-domain* **by** *simp*
  have Γ ⊢ T_2 <: T_2 **using** *ih*-T_2 *closed*$_{T2}$ *ok* **by** *simp*
  moreover
  have ((*X*,T_2)::Γ) ⊢ T_1 <: T_1 **using** *ih*-T_1 *closed*$_{T1}$ *ok'* **by** *simp*
  ultimately show Γ ⊢ ∀[*X*<:T_2].T_1 <: ∀[*X*<:T_2].T_1 **using** *fresh-cond*

by (*simp add*: *subtype-of.S-Forall*)
qed (*auto simp add*: *closed-in-def ty.supp supp-atm*)

lemma *subtype-reflexivity-semiautomated*:
assumes *a*: $\vdash \Gamma$ *ok*
and *b*: *T closed-in* Γ
shows $\Gamma \vdash T <: T$
using *a b*
apply(*nominal-induct T avoiding*: Γ *rule*: *ty.induct*)
apply(*auto simp add*: *ty.supp abs-supp closed-in-def supp-atm*)
— Too bad that this instantiation cannot be found automatically by **auto**; **blast** would find it if we had not used an explicit definition for *closed-in-def*.
apply(*drule-tac x*=(*tyvrs*, *ty2*)::Γ **in** *meta-spec*)
apply(*force simp add*: *closed-in-def fresh-domain*)
done

6 Weakening

In order to prove weakening we introduce the notion of a type-context extending another. This generalization seems to make the proof for weakening to be smoother than if we had strictly adhered to the version in the POPLmark-paper.

constdefs
extends :: *ty-context* $\Rightarrow$ *ty-context* $\Rightarrow$ *bool* (- *extends* - [*100*,*100*] *100*)
Δ *extends* $\Gamma \equiv \forall X\ Q.\ (X,Q) \in set\ \Gamma \longrightarrow (X,Q) \in set\ \Delta$

lemma *extends-domain*:
assumes *a*: Δ *extends* Γ
shows *domain* $\Gamma \subseteq$ *domain* Δ
using *a*
apply (*auto simp add*: *extends-def*)
apply (*drule domain-existence*)
apply (*force simp add*: *domain-inclusion*)
done

lemma *extends-closed*:
assumes *a1*: *T closed-in* Γ
and *a2*: Δ *extends* Γ
shows *T closed-in* Δ
using *a1 a2*
by (*auto dest*: *extends-domain simp add*: *closed-in-def*)

lemma *extends-memb*:
assumes *a*: Δ *extends* Γ
and *b*: $(X,T) \in set\ \Gamma$
shows $(X,T) \in set\ \Delta$
using *a b* **by** (*simp add*: *extends-def*)

lemma *weakening*:

assumes a: $\Gamma \vdash S <: T$
and b: $\vdash \Delta$ *ok*
and c: Δ *extends* Γ
shows $\Delta \vdash S <: T$
using $a\ b\ c$
proof (*nominal-induct* Γ S T *avoiding*: Δ *rule*: *subtype-of-induct*)
case (*S-Top* Γ S)
have *lh-drv-prem*: S *closed-in* Γ **by** *fact*
have $\vdash \Delta$ *ok* **by** *fact*
moreover
have Δ *extends* Γ **by** *fact*
hence S *closed-in* Δ **using** *lh-drv-prem* **by** (*simp only*: *extends-closed*)
ultimately show $\Delta \vdash S <:$ *Top* **by** *force*
next
case (*S-Var* Γ X S T)
have *lh-drv-prem*: $(X,S) \in$ *set* Γ **by** *fact*
have *ih*: $\bigwedge\Delta.\ \vdash \Delta$ *ok* $\Longrightarrow \Delta$ *extends* $\Gamma \Longrightarrow \Delta \vdash S <: T$ **by** *fact*
have *ok*: $\vdash \Delta$ *ok* **by** *fact*
have *extends*: Δ *extends* Γ **by** *fact*
have $(X,S) \in$ *set* Δ **using** *lh-drv-prem extends* **by** (*simp only*: *extends-memb*)
moreover
have $\Delta \vdash S <: T$ **using** *ok extends ih* **by** *simp*
ultimately show $\Delta \vdash$ *Tvar* $X <: T$ **using** *ok* **by** *force*
next
case (*S-Refl* Γ X)
have *lh-drv-prem*: $X \in$ *domain* Γ **by** *fact*
have $\vdash \Delta$ *ok* **by** *fact*
moreover
have Δ *extends* Γ **by** *fact*
hence $X \in$ *domain* Δ **using** *lh-drv-prem* **by** (*force dest*: *extends-domain*)
ultimately show $\Delta \vdash$ *Tvar* $X <:$ *Tvar* X **by** *force*
next
case (*S-Arrow* Γ S_1 S_2 T_1 T_2) **thus** $\Delta \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$ **by** *blast*
next
case (*S-Forall* Γ X S_1 S_2 T_1 T_2)
have *fresh-cond*: $X \# \Delta$ **by** *fact*
hence *fresh-domain*: $X \#($*domain* $\Delta)$ **by** (*simp add*: *fresh-domain*)
have ih_1: $\bigwedge\Delta.\ \vdash \Delta$ *ok* $\Longrightarrow \Delta$ *extends* $\Gamma \Longrightarrow \Delta \vdash T_1 <: S_1$ **by** *fact*
have ih_2: $\bigwedge\Delta.\ \vdash \Delta$ *ok* $\Longrightarrow \Delta$ *extends* $((X,T_1)::\Gamma) \Longrightarrow \Delta \vdash S_2 <: T_2$ **by** *fact*
have *lh-drv-prem*: $\Gamma \vdash T_1 <: S_1$ **by** *fact*
hence $closed_{T1}$: T_1 *closed-in* Γ **by** (*simp add*: *subtype-implies-closed*)
have *ok*: $\vdash \Delta$ *ok* **by** *fact*
have *ext*: Δ *extends* Γ **by** *fact*
have T_1 *closed-in* Δ **using** *ext* $closed_{T1}$ **by** (*simp only*: *extends-closed*)
hence $\vdash ((X,T_1)::\Delta)$ *ok* **using** *fresh-domain ok* **by** *force*
moreover
have $((X,T_1)::\Delta)$ *extends* $((X,T_1)::\Gamma)$ **using** *ext* **by** (*force simp add*: *extends-def*)
ultimately have $((X,T_1)::\Delta) \vdash S_2 <: T_2$ **using** ih_2 **by** *simp*
moreover
have $\Delta \vdash T_1 <: S_1$ **using** *ok ext* ih_1 **by** *simp*

ultimately show $\Delta \vdash \forall[X{<:}S_1].S_2 <: \forall[X{<:}T_1].T_2$ **using** *ok* **by** (*force intro*: *S-Forall*)
qed

In fact all "non-binding" cases can be solved automatically:

lemma *weakening-more-automated*:
assumes *a*: $\Gamma \vdash S <: T$
and *b*: $\vdash \Delta$ *ok*
and *c*: Δ *extends* Γ
shows $\Delta \vdash S <: T$
using *a b c*
proof (*nominal-induct* Γ S T *avoiding*: Δ *rule*: *subtype-of-induct*)
case (*S-Forall* Γ X S_1 S_2 T_1 T_2)
have *fresh-cond*: $X \# \Delta$ **by** *fact*
hence *fresh-domain*: $X \#(domain\ \Delta)$ **by** (*simp add*: *fresh-domain*)
have ih_1: $\bigwedge\Delta.\ \vdash \Delta\ ok \Longrightarrow \Delta\ extends\ \Gamma \Longrightarrow \Delta \vdash T_1 <: S_1$ **by** *fact*
have ih_2: $\bigwedge\Delta.\ \vdash \Delta\ ok \Longrightarrow \Delta\ extends\ ((X,T_1)::\Gamma) \Longrightarrow \Delta \vdash S_2 <: T_2$ **by** *fact*
have *lh-drv-prem*: $\Gamma \vdash T_1 <: S_1$ **by** *fact*
hence $closed_{T1}$: T_1 *closed-in* Γ **by** (*simp add*: *subtype-implies-closed*)
have *ok*: $\vdash \Delta$ *ok* **by** *fact*
have *ext*: Δ *extends* Γ **by** *fact*
have T_1 *closed-in* Δ **using** *ext* $closed_{T1}$ **by** (*simp only*: *extends-closed*)
hence $\vdash ((X,T_1)::\Delta)$ *ok* **using** *fresh-domain ok* **by** *force*
moreover
have $((X,T_1)::\Delta)$ *extends* $((X,T_1)::\Gamma)$ **using** *ext* **by** (*force simp add*: *extends-def*)
ultimately have $((X,T_1)::\Delta) \vdash S_2 <: T_2$ **using** ih_2 **by** *simp*
moreover
have $\Delta \vdash T_1 <: S_1$ **using** *ok ext* ih_1 **by** *simp*
ultimately show $\Delta \vdash \forall[X{<:}S_1].S_2 <: \forall[X{<:}T_1].T_2$ **using** *ok* **by** (*force intro*: *S-Forall*)
qed (*blast intro*: *extends-closed extends-memb dest*: *extends-domain*)+

7 Transitivity and Narrowing

Some inversion lemmas that are needed in the transitivity and narrowing proof.

lemma *S-TopE*:
assumes *a*: $\Gamma \vdash Top <: T$
shows $T = Top$
using *a* **by** (*cases*, *auto*)

lemma *S-ArrowE-left*:
assumes *a*: $\Gamma \vdash S_1 \rightarrow S_2 <: T$
shows $T = Top \vee (\exists\, T_1\ T_2.\ T = T_1 \rightarrow T_2 \wedge \Gamma \vdash T_1 <: S_1 \wedge \Gamma \vdash S_2 <: T_2)$
using *a* **by** (*cases*, *auto simp add*: *ty.inject*)

lemma *S-ForallE-left*:
shows $[\![\Gamma \vdash \forall[X{<:}S_1].S_2 <: T;\ X\#\Gamma;\ X\#S_1]\!]$
$\Longrightarrow T = Top \vee (\exists\, T_1\ T_2.\ T = \forall[X{<:}T_1].T_2 \wedge \Gamma \vdash T_1 <: S_1 \wedge ((X,T_1)::\Gamma) \vdash S_2$
$<: T_2)$
apply(*frule subtype-implies-ok*)

apply(*ind-cases* $\Gamma \vdash \forall[X<:S_1].S_2 <: T$)
apply(*auto simp add*: *ty.inject alpha*)
apply(*rule-tac* $x=[(X,Xa)]\cdot T_2$ **in** *exI*)
apply(*rule conjI*)
apply(*rule sym*)
apply(*rule pt-bij2*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
apply(*rule pt-tyvrs3*)
apply(*simp*)
apply(*rule at-ds5*[*OF at-tyvrs-inst*])
apply(*rule conjI*)
apply(*simp add*: *pt-fresh-left*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*] *calc-atm*)
apply(*drule-tac* $\Gamma=((Xa,T_1)::\Gamma)$ **in** *subtype-implies-closed*)+
apply(*simp add*: *closed-in-def*)
apply(*drule fresh-domain*)+
apply(*simp add*: *fresh-def*)
apply(*subgoal-tac* $X \notin$ (*insert Xa* (*domain* Γ)))
apply(*force*)
apply(*simp add*: *at-fin-set-supp*[*OF at-tyvrs-inst*, *OF finite-domain*])
apply(*frule-tac* $X=X$ **in** *subtype-implies-fresh*)
apply(*assumption*)
apply(*drule-tac* $X=Xa$ **in** *subtype-implies-fresh*)
apply(*assumption*)
apply(*simp add*: *fresh-prod*)
apply(*drule-tac* $pi=[(X,Xa)]$ **in** *subtype-eqvt*)
apply(*simp add*: *calc-atm*)
apply(*simp add*: *pt-fresh-fresh*[*OF pt-tyvrs-inst*, *OF at-tyvrs-inst*])
done

Next we prove the transitivity and narrowing for the subtyping-relation. The POPLmark-paper says the following:

> **Lemma 1 (Transitivity and Narrowing)**
>
> 1. *If* $\Gamma\vdash S<:Q$ *and* $\Gamma\vdash Q<:T$, *then* $\Gamma\vdash S<:T$.
> 2. *If* $\Gamma,X<:Q,\Delta \vdash M<:N$ *and* $\Gamma\vdash P<:Q$ *then* $\Gamma,X<:P,\Delta \vdash M<:N$.
>
> The two parts are proved simultaneously, by induction on the size of Q. The argument for part (2) assumes that part (1) has been established already for the Q in question; part (1) uses part (2) only for strictly smaller Q.

For the induction on the size of Q, we use the induction-rule *measure-induct-rule*:

$$(\bigwedge x.\ (\bigwedge y.\ \textit{size-ty}\ y < \textit{size-ty}\ x \Longrightarrow P\ y) \Longrightarrow P\ x) \Longrightarrow P\ a$$

That means in order to show a property $P\ a$ for all a, the induct-rule requires to prove that for all x $P\ x$ holds using the assumption that for all y whose size is strictly smaller than that of x the property $P\ y$ holds.

lemma
  shows *trans*: $\Gamma \vdash S <: Q \Longrightarrow \Gamma \vdash Q <: T \Longrightarrow \Gamma \vdash S <: T$
  and *narrow*: $(\Delta @ [(X, Q)] @ \Gamma) \vdash M <: N \Longrightarrow \Gamma \vdash P <: Q \Longrightarrow (\Delta @ [(X, P)] @ \Gamma) \vdash M <: N$
proof (*induct Q fixing*: Γ S T Δ X P M N *taking*: *size-ty rule*: *measure-induct-rule*)
  case (*less* Q)
  — First we mention the induction hypotheses of the outer induction for later reference:
  have *IH-trans*:
  $\bigwedge Q'\ \Gamma\ S\ T.\ [\![\textit{size-ty}\ Q' < \textit{size-ty}\ Q;\ \Gamma \vdash S <: Q';\ \Gamma \vdash Q' <: T]\!] \Longrightarrow \Gamma \vdash S <: T$ **by** *fact*
  have *IH-narrow*:
  $\bigwedge Q'\ \Delta\ \Gamma\ X\ M\ N\ P.\ [\![\textit{size-ty}\ Q' < \textit{size-ty}\ Q;\ (\Delta @ [(X, Q')] @ \Gamma) \vdash M <: N;\ \Gamma \vdash P <: Q]\!]$
  $\Longrightarrow (\Delta @ [(X, P)] @ \Gamma) \vdash M <: N$ **by** *fact*
  — We proceed with the transitivity proof as an auxiliary lemma, because it needs to be referenced in the narrowing proof.
  have *transitivity-aux*:
  $\bigwedge \Gamma\ S\ T.\ [\![\Gamma \vdash S <: Q;\ \Gamma \vdash Q <: T]\!] \Longrightarrow \Gamma \vdash S <: T$
  proof −
    fix Γ' S' T
    assume $\Gamma' \vdash S' <: Q$ — left-hand derivation
      and $\Gamma' \vdash Q <: T$ — right-hand derivation
    thus $\Gamma' \vdash S' <: T$
    proof (*nominal-induct* Γ' S' $Q \equiv Q$ *avoiding*: Γ' S' *rule*: *subtype-of-induct*)
      case (*S-Top* Γ S)
      — In this case the left-hand derivation is $\Gamma \vdash S <: \textit{Top}$, giving us $\vdash \Gamma$ *ok* and S *closed-in* Γ. This case is straightforward, because the right-hand derivation must be of the form $\Gamma \vdash \textit{Top} <: \textit{Top}$ giving us the equation $T = \textit{Top}$.

      hence *rh-drv*: $\Gamma \vdash \textit{Top} <: T$ **by** *simp*
      hence *T-inst*: $T = \textit{Top}$ **by** (*simp add*: *S-TopE*)
      have $\vdash \Gamma$ *ok*
        and S *closed-in* Γ **by** *fact*
      hence $\Gamma \vdash S <: \textit{Top}$ **by** (*simp add*: *subtype-of.S-Top*)
      thus $\Gamma \vdash S <: T$ **using** *T-inst* **by** *simp*
    next
      case (*S-Var* Γ Y U)
      — In this case the left-hand derivation is $\Gamma \vdash \textit{Tvar}\ Y <: Q$ with $S = \textit{Tvar}\ Y$. We have therefore (Y, U) is in Γ and by inner induction hypothesis that $\Gamma \vdash U <: T$. By *S-Var* follows $\Gamma \vdash \textit{Tvar}\ Y <: T$.

      hence *IH-inner*: $\Gamma \vdash U <: T$ **by** *simp*
      have $(Y, U) \in \textit{set}\ \Gamma$ **by** *fact*
      with *IH-inner* **show** $\Gamma \vdash \textit{Tvar}\ Y <: T$ **by** (*simp add*: *subtype-of.S-Var*)
    next
      case (*S-Refl* Γ X)
      — In this case the left-hand derivation is $\Gamma \vdash \textit{Tvar}\ X <: \textit{Tvar}\ X$ with $Q = \textit{Tvar}\ X$. The goal then follows immediately from the right-hand derivation.

      thus $\Gamma \vdash \textit{Tvar}\ X <: T$ **by** *simp*
    next
      case (*S-Arrow* Γ S_1 S_2 Q_1 Q_2)

— In this case the left-hand derivation is $\Gamma \vdash S_1 \rightarrow S_2 <: Q_1 \rightarrow Q_2$ with $S_1 \rightarrow S_2 = S$ and $Q_1 \rightarrow Q_2 = Q$. We know that the *size-ty* of Q_1 and Q_2 is smaller than that of Q; so we can apply the outer induction hypotheses for Q_1 and Q_2. We also have the sub-derivations $\Gamma \vdash Q_1 <: S_1$ and $\Gamma \vdash S_2 <: Q_2$. The right-hand derivation is $\Gamma \vdash Q_1 \rightarrow Q_2 <: T$. There exist types T_1, T_2 such that $T = Top \vee T = T_1 \rightarrow T_2$. The *Top*-case is straightforward once we know $S_1 \rightarrow S_2$ *closed-in* Γ and $\vdash \Gamma$ *ok*. In the other case we have the sub-derivations $\Gamma \vdash T_1 <: Q_1$ and $\Gamma \vdash Q_2 <: T_2$. Using the outer induction hypothesis for transitivity we can derive $\Gamma \vdash T_1 <: S_1$ and $\Gamma \vdash S_2 <: T_2$. By rule *S-Arrow* follows $\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$, which is $\Gamma \vdash S_1 \rightarrow S_2 <: T$.

hence *rh-drv*: $\Gamma \vdash Q_1 \rightarrow Q_2 <: T$ **by** *simp*
from $\langle Q_1 \rightarrow Q_2 = Q \rangle$
have Q_{12}*-less*: *size-ty* Q_1 < *size-ty* Q *size-ty* Q_2 < *size-ty* Q **by** *auto*
have *lh-drv-prm*$_1$: $\Gamma \vdash Q_1 <: S_1$ **by** *fact*
have *lh-drv-prm*$_2$: $\Gamma \vdash S_2 <: Q_2$ **by** *fact*
from *rh-drv* **have** $T = Top \vee (\exists\, T_1\ T_2.\ T = T_1 \rightarrow T_2 \wedge \Gamma \vdash T_1 <: Q_1 \wedge \Gamma \vdash Q_2 <: T_2)$
  by (*simp add*: *S-ArrowE-left*)
moreover
have S_1 *closed-in* Γ **and** S_2 *closed-in* Γ
  using *lh-drv-prm*$_1$ *lh-drv-prm*$_2$ **by** (*simp-all add*: *subtype-implies-closed*)
hence $(S_1 \rightarrow S_2)$ *closed-in* Γ **by** (*simp add*: *closed-in-def ty.supp*)
moreover
have $\vdash \Gamma$ *ok* **using** *rh-drv* **by** (*rule subtype-implies-ok*)
moreover
{ assume $\exists\, T_1\ T_2.\ T = T_1 \rightarrow T_2 \wedge \Gamma \vdash T_1 <: Q_1 \wedge \Gamma \vdash Q_2 <: T_2$
  then obtain $T_1\ T_2$
    where *T-inst*: $T = T_1 \rightarrow T_2$
    and *rh-drv-prm*$_1$: $\Gamma \vdash T_1 <: Q_1$
    and *rh-drv-prm*$_2$: $\Gamma \vdash Q_2 <: T_2$ **by** *force*
  from *IH-trans*[*of* Q_1]
  have $\Gamma \vdash T_1 <: S_1$ **using** Q_{12}*-less rh-drv-prm*$_1$ *lh-drv-prm*$_1$ **by** *simp*
  moreover
  from *IH-trans*[*of* Q_2]
  have $\Gamma \vdash S_2 <: T_2$ **using** Q_{12}*-less rh-drv-prm*$_2$ *lh-drv-prm*$_2$ **by** *simp*
  ultimately have $\Gamma \vdash S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$ **by** (*simp add*: *subtype-of.S-Arrow*)
  hence $\Gamma \vdash S_1 \rightarrow S_2 <: T$ **using** *T-inst* **by** *simp*
}
ultimately show $\Gamma \vdash S_1 \rightarrow S_2 <: T$ **by** *blast*
next
case (*S-Forall* Γ X S_1 S_2 Q_1 Q_2)

— In this case the left-hand derivation is $\Gamma\vdash\forall[X<:S_1].S_2 <: \forall[X<:Q_1].Q_2$ with $\forall[X<:S_1].S_2=S$ and $\forall[X<:Q_1].Q_2=Q$. We therefore have the sub-derivations $\Gamma\vdash Q_1<:S_1$ and $((X,\ Q_1) :: \Gamma)\vdash S_2<:Q_2$. Since X is a binder, we assume that it is sufficiently fresh; in particular we have the freshness conditions $X \# \Gamma$ and $X \# Q_1$ (these assumptions are provided by the strong induction-rule *subtype-of-induct*). We know that the *size-ty* of Q_1 and Q_2 is smaller than that of Q; so we can apply the outer induction hypotheses for Q_1 and Q_2. The right-hand derivation is $\Gamma \vdash \forall[X<:Q_1].Q_2 <: T$. Since $X \# \Gamma$ and $X \# Q_1$ there exists types T_1,T_2 such that $T=\mathit{Top} \vee T=\forall[X<:T_1].T_2$. The *Top*-case is straightforward once we know $(\forall[X<:S_1].S_2)$ *closed-in* Γ and $\vdash \Gamma$ *ok*. In the other case we have the sub-derivations $\Gamma\vdash T_1<:Q_1$ and $((X,\ T_1) :: \Gamma)\vdash Q_2<:T_2$. Using the outer induction hypothesis for transitivity we can derive $\Gamma\vdash T_1<:S_1$. From the outer induction for narrowing we get $((X,\ T_1) :: \Gamma)\vdash S_2<:Q_2$ and then using again induction for transitivity we obtain $((X,\ T_1) :: \Gamma)\vdash S_2<:T_2$. By rule *S-Forall* and the freshness condition $X \# \Gamma$ follows $\Gamma \vdash \forall[X<:S_1].S_2 <: \forall[X<:T_1].T_2$, which is $\Gamma \vdash \forall[X<:S_1].S_2 <: T$.

hence *rh-drv*: $\Gamma \vdash \forall[X<:Q_1].Q_2 <: T$ **by** *simp*
have *lh-drv-prm*$_1$: $\Gamma \vdash Q_1 <: S_1$ **by** *fact*
have *lh-drv-prm*$_2$: $((X,Q_1)::\Gamma) \vdash S_2 <: Q_2$ **by** *fact*
have *fresh-cond*: $X\#\Gamma$ $X\#Q_1$ **by** *fact*
from $\langle\forall[X<:Q_1].Q_2 = Q\rangle$
have Q_{12}*-less*: *size-ty* Q_1 < *size-ty* Q *size-ty* Q_2 < *size-ty* Q **by** *auto*
from *rh-drv*
have $T=\mathit{Top} \vee (\exists\, T_1\ T_2.\ T=\forall[X<:T_1].T_2 \wedge \Gamma\vdash T_1<:Q_1 \wedge ((X,T_1)::\Gamma)\vdash Q_2<:T_2)$
  using *fresh-cond* **by** (*simp add*: *S-ForallE-left*)
moreover
have S_1 *closed-in* Γ **and** S_2 *closed-in* $((X,Q_1)::\Gamma)$
  using *lh-drv-prm*$_1$ *lh-drv-prm*$_2$ **by** (*simp-all add*: *subtype-implies-closed*)
hence $(\forall[X<:S_1].S_2)$ *closed-in* Γ **by** (*force simp add*: *closed-in-def ty.supp abs-supp*)
moreover
have $\vdash \Gamma$ *ok* **using** *rh-drv* **by** (*rule subtype-implies-ok*)
moreover
{ **assume** $\exists\, T_1\ T_2.\ T=\forall[X<:T_1].T_2 \wedge \Gamma\vdash T_1<:Q_1 \wedge ((X,T_1)::\Gamma)\vdash Q_2<:T_2$
  then obtain T_1 T_2
    where *T-inst*: $T = \forall[X<:T_1].T_2$
    and *rh-drv-prm*$_1$: $\Gamma \vdash T_1 <: Q_1$
    and *rh-drv-prm*$_2$:$((X,T_1)::\Gamma) \vdash Q_2 <: T_2$ **by** *force*
  from *IH-trans*[*of* Q_1]
  have $\Gamma \vdash T_1 <: S_1$ **using** *lh-drv-prm*$_1$ *rh-drv-prm*$_1$ Q_{12}*-less* **by** *blast*
  moreover
  from *IH-narrow*[*of* Q_1 []]
  have $((X,T_1)::\Gamma) \vdash S_2 <: Q_2$ **using** Q_{12}*-less lh-drv-prm*$_2$ *rh-drv-prm*$_1$ **by** *simp*
  with *IH-trans*[*of* Q_2]
  have $((X,T_1)::\Gamma) \vdash S_2 <: T_2$ **using** Q_{12}*-less rh-drv-prm*$_2$ **by** *simp*
  ultimately have $\Gamma \vdash \forall[X<:S_1].S_2 <: \forall[X<:T_1].T_2$
    using *fresh-cond* **by** (*simp add*: *subtype-of.S-Forall*)
  hence $\Gamma \vdash \forall[X<:S_1].S_2 <: T$ **using** *T-inst* **by** *simp*
}
ultimately show $\Gamma \vdash \forall[X<:S_1].S_2 <: T$ **by** *blast*
qed

qed

{ — The transitivity proof is now by the auxiliary lemma.
case *1*
have $\Gamma \vdash S <: Q$
and $\Gamma \vdash Q <: T$ **by** *fact*
thus $\Gamma \vdash S <: T$ **by** (*rule transitivity-aux*)
next
— The narrowing proof proceeds by an induction over $(\Delta\ @\ [(X,\ Q)]\ @\ \Gamma)\vdash M{<:}N$.
case *2*
have $(\Delta @[(X,Q)]@\Gamma) \vdash M <: N$ — left-hand derivation
and $\Gamma \vdash P{<:}Q$ **by** *fact* — right-hand derivation
thus $(\Delta @[(X,P)]@\Gamma) \vdash M <: N$
proof (*nominal-induct* $\Gamma{\equiv}\Delta@[(X,Q)]@\Gamma$ M N *avoiding*: Δ Γ X *rule*: *subtype-of-induct*)
case (*S-Top* - S Δ Γ X)
— In this case the left-hand derivation is $(\Delta\ @\ [(X,\ Q)]\ @\ \Gamma)\vdash S{<:}\mathit{Top}$. We show that the context $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$ is ok and that S is closed in $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$. Then we can apply the *S-Top*-rule.

hence *lh-drv-prm*$_1$: $\vdash (\Delta@[(X,Q)]@\Gamma)$ *ok*
and *lh-drv-prm*$_2$: S *closed-in* $(\Delta@[(X,Q)]@\Gamma)$ **by** *simp-all*
have *rh-drv*: $\Gamma \vdash P <: Q$ **by** *fact*
hence P *closed-in* Γ **by** (*simp add*: *subtype-implies-closed*)
with *lh-drv-prm*$_1$ **have** $\vdash (\Delta@[(X,P)]@\Gamma)$ *ok* **by** (*simp add*: *replace-type*)
moreover
from *lh-drv-prm*$_2$ **have** S *closed-in* $(\Delta@[(X,P)]@\Gamma)$
by (*simp add*: *closed-in-def domain-append*)
ultimately show $(\Delta@[(X,P)]@\Gamma) \vdash S <: \mathit{Top}$ **by** (*simp add*: *subtype-of.S-Top*)
next
case (*S-Var* - Y S N Δ Γ X)
— In this case the left-hand derivation is $(\Delta\ @\ [(X,\ Q)]\ @\ \Gamma)\vdash \mathit{Tvar}\ Y{<:}N$ and by inner induction hypothesis we have $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma)\vdash S{<:}N$. We therefore know that the contexts $\Delta\ @\ [(X,\ Q)]\ @\ \Gamma$ and $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$ are ok, and that $(Y,\ S)$ is in $\Delta\ @\ [(X,\ Q)]\ @\ \Gamma$. We need to show that $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma)\vdash \mathit{Tvar}\ Y{<:}N$ holds. In case $X \neq Y$ we know that $(Y,\ S)$ is in $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$ and can use the inner induction hypothesis and rule *S-Var* to conclude. In case $X = Y$ we can infer that $S = Q$; moreover we have that $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma)$ *extends* Γ and therefore by *weakening* that $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma)\vdash P{<:}Q$ holds. By transitivity we obtain then $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma)\vdash P{<:}N$ and can conclude by applying rule *S-Var*.

hence *IH-inner*: $(\Delta@[(X,P)]@\Gamma) \vdash S <: N$
and *lh-drv-prm*: $(Y,S) \in$ *set* $(\Delta@[(X,Q)]@\Gamma)$
and *rh-drv*: $\Gamma \vdash P{<:}Q$
and ok_Q: $\vdash (\Delta@[(X,Q)]@\Gamma)$ *ok* **by** (*simp-all add*: *subtype-implies-ok*)
hence ok_P: $\vdash (\Delta@[(X,P)]@\Gamma)$ *ok* **by** (*simp add*: *subtype-implies-ok*)
show $(\Delta@[(X,P)]@\Gamma) \vdash \mathit{Tvar}\ Y <: N$
proof (*cases* $X{=}Y$)
case *False*
have $X{\neq}Y$ **by** *fact*
hence $(Y,S){\in}$*set* $(\Delta@[(X,P)]@\Gamma)$ **using** *lh-drv-prm* **by** *simp*
with *IH-inner* **show** $(\Delta@[(X,P)]@\Gamma) \vdash \mathit{Tvar}\ Y <: N$ **by** (*simp add*: *subtype-of.S-Var*)

next
case *True*
have $memb_{XQ}$: $(X,Q) \in set\ (\Delta @ [(X,Q)] @ \Gamma)$ **by** *simp*
have $memb_{XP}$: $(X,P) \in set\ (\Delta @ [(X,P)] @ \Gamma)$ **by** *simp*
have *eq*: $X = Y$ **by** *fact*
hence $S = Q$ **using** ok_Q *lh-drv-prm* $memb_{XQ}$ **by** (*simp only*: *uniqueness-of-ctxt*)
hence $(\Delta @ [(X,P)] @ \Gamma) \vdash Q <: N$ **using** *IH-inner* **by** *simp*
moreover
have $(\Delta @ [(X,P)] @ \Gamma)$ *extends* Γ **by** (*simp add*: *extends-def*)
hence $(\Delta @ [(X,P)] @ \Gamma) \vdash P <: Q$ **using** *rh-drv* ok_P **by** (*simp only*: *weakening*)
ultimately have $(\Delta @ [(X,P)] @ \Gamma) \vdash P <: N$ **by** (*simp add*: *transitivity-aux*)
thus $(\Delta @ [(X,P)] @ \Gamma) \vdash Tvar\ Y <: N$ **using** $memb_{XP}$ *eq* **by** (*simp only*: *subtype-of.S-Var*)
qed
next
case (*S-Refl* - $Y\ \Delta\ \Gamma\ X$)
— In this case the left-hand derivation is $(\Delta\ @\ [(X,\ Q)]\ @\ \Gamma) \vdash Tvar\ Y <: Tvar\ Y$ and we therefore know that $\Delta\ @\ [(X,\ Q)]\ @\ \Gamma$ is ok and that Y is in the domain of $\Delta\ @\ [(X,\ Q)]\ @\ \Gamma$. We therefore know that $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$ is ok and that Y is in the domain of $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$. We can conclude by applying rule *S-Refl*.

hence $lh\text{-}drv\text{-}prm_1$: $\vdash (\Delta @ [(X,Q)] @ \Gamma)$ *ok*
and $lh\text{-}drv\text{-}prm_2$: $Y \in domain\ (\Delta @ [(X,Q)] @ \Gamma)$ **by** *simp-all*
have $\Gamma \vdash P <: Q$ **by** *fact*
hence P *closed-in* Γ **by** (*simp add*: *subtype-implies-closed*)
with $lh\text{-}drv\text{-}prm_1$ **have** $\vdash (\Delta @ [(X,P)] @ \Gamma)$ *ok* **by** (*simp add*: *replace-type*)
moreover
from $lh\text{-}drv\text{-}prm_2$ **have** $Y \in domain\ (\Delta @ [(X,P)] @ \Gamma)$ **by** (*simp add*: *domain-append*)
ultimately show $(\Delta @ [(X,P)] @ \Gamma) \vdash Tvar\ Y <: Tvar\ Y$ **by** (*simp add*: *subtype-of.S-Refl*)
next
case (*S-Arrow* - $Q_1\ Q_2\ S_1\ S_2\ \Delta\ \Gamma\ X$)
— In this case the left-hand derivation is $(\Delta\ @\ [(X,\ Q)]\ @\ \Gamma) \vdash Q_1 \rightarrow Q_2 <: S_1 \rightarrow S_2$ and the proof is trivial.

thus $(\Delta @ [(X,P)] @ \Gamma) \vdash Q_1 \rightarrow Q_2 <: S_1 \rightarrow S_2$ **by** *blast*
next
case (*S-Forall* - $Y\ S_1\ S_2\ T_1\ T_2\ \Delta\ \Gamma\ X$)
— In this case the left-hand derivation is $(\Delta @ [(X,Q)] @ \Gamma) \vdash \forall [Y <: S_1].S_2 <: \forall [Y <: T_1].T_2$ and therfore we know that the binder Y is fresh for $\Delta\ @\ [(X,\ Q)]\ @\ \Gamma$. By the inner induction hypothesis we have that $(\Delta\ @\ [(X,\ P)]\ @\ \Gamma) \vdash T_1 <: S_1$ and $((Y,\ T_1) :: \Delta\ @\ [(X,\ P)]\ @\ \Gamma) \vdash S_2 <: T_2$. Since P is a subtype of Q we can infer that Y is fresh for P and thus also fresh for $\Delta\ @\ [(X,\ P)]\ @\ \Gamma$. We can then conclude by applying rule *S-Forall*.

hence $IH\text{-}inner_1$: $(\Delta @ [(X,P)] @ \Gamma) \vdash T_1 <: S_1$
and $IH\text{-}inner_2$: $((Y,T_1) :: \Delta @ [(X,P)] @ \Gamma) \vdash S_2 <: T_2$
and *lh-drv-prm*: $Y \sharp (\Delta @ [(X,Q)] @ \Gamma)$ **by** *force+*
have *rh-drv*: $\Gamma \vdash P <: Q$ **by** *fact*
hence $Y \sharp P$ **using** *lh-drv-prm* **by** (*simp only*: *fresh-list-append subtype-implies-fresh*)
hence $Y \sharp (\Delta @ [(X,P)] @ \Gamma)$ **using** *lh-drv-prm*
by (*simp add*: *fresh-list-append fresh-list-cons fresh-prod*)
with $IH\text{-}inner_1$ $IH\text{-}inner_2$
show $(\Delta @ [(X,P)] @ \Gamma) \vdash \forall [Y <: S_1].S_2 <: \forall [Y <: T_1].T_2$ **by** (*simp add*: *subtype-of.S-Forall*)

```
    qed
  }
qed

end
```