Programmation Avancée

Sous-typage: types récursifs

David Baelde

ENS Paris-Saclay, L3 2020–2021

Types infinis

Considérons des types infinis,
par exemple $\mathsf{Int} \times (\mathsf{Int} \times \ldots)$ c'est à dire le type satisfaisant $\tau = \mathsf{Int} \times \tau$.
Pour pouvoir en profiter, on se donne des définitions récursives :

$$\frac{\Gamma, x : \tau \vdash u : \tau}{\Gamma \vdash \mathsf{rec}\ x.u : \tau} \qquad u \rightsquigarrow u\{x \mapsto \mathsf{rec}\ x.u\}$$

Les dérivations de typage restent des arbres finis (définition inductive).

Exemple

$\vdash \mathsf{rec}\ x.\langle 0, \langle 1, x \rangle\rangle : \tau$ où $\tau = \mathsf{Int} \times \tau$.
$\vdash (\mathsf{rec}\ f.\lambda x.\langle x, f(x+1)\rangle)0 : \tau$ aussi, et s'évalue en $\langle 0, \langle 1, \langle 2, \ldots\rangle\rangle\rangle$.
$\vdash \mathsf{rec}\ f.\lambda x.f : \tau'$ où $\tau' : \mathsf{Any} \to \tau'$.

Les deux commandements

Théorème

Si $\Gamma \vdash u : \tau$ et $u \rightsquigarrow v$ alors $\Gamma \vdash v : \tau$.

Théorème

Si u n'est pas une valeur et $\vdash u : \tau$ alors il existe v tel que $u \rightsquigarrow v$.

Les preuves ne sont pas changées par le passage aux types infinis et l'ajout de la récursion.

- Récurrence sur les termes et les dérivations, pas les types.
- Le terme $\mathrm{rec}\ x.u$ est un redex : nouveau cas facile pour la préservation.
- Le terme $\mathrm{rec}\ x.u$ n'est pas une valeur : cas immédiat pour le progrès.

On ne s'intéresse pas à la terminaison de la réduction !

Relation de sous-typage

Les contraintes sur la relation de sous-typage sont donc inchangées :

- La relation est réflexive et transitive.
- Tout sous-type d'une flèche est une flèche, tout sous-type d'un produit est un produit, etc.
- Si $\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'$ alors $\tau_2 \leq \tau_2'$ et $\tau_1' \leq \tau_1$, et autres inversions.

On peut donc toujours prendre la relation $\leq$ inductivement générée par les règles

$$\frac{\mathrm{T} \leq^B \mathrm{T'}}{\mathrm{T} \leq \mathrm{T'}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'} \qquad \frac{\tau_1 \leq \tau_1' \quad \tau_2 \leq \tau_2'}{\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'} \qquad \text{etc.}$$

et juste ajouter une règle de réflexivité !

Exemple

Soit Even^ω le type $\tau = \mathrm{Even} \times \tau$ et de même pour Int^ω.
On a $\mathrm{Even} \times \mathrm{Even}^\omega \leq \mathrm{Int} \times \mathrm{Even}^\omega$ mais $\mathrm{Even}^\omega \not\leq \mathrm{Int}^\omega$.

Points fixes

Théorème

- Une fonction monotone F sur un treillis complet admet un plus petit point fixe $\mathrm{lfp}(F)$. De plus $\mathrm{lfp}(F)$ est le plus petit des pré-points fixes de F, i.e. des X tel que $F(X) \subseteq X$.

La relation $\leq$ définie inductivement est $\mathrm{lfp}(F)$ où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \ldots$$

(Ou une version de F enrichie qui travaille sur des ensembles de dérivations.)

Points fixes

Théorème

- Une fonction monotone F sur un treillis complet admet un plus petit point fixe $\mathrm{lfp}(F)$. De plus $\mathrm{lfp}(F)$ est le plus petit des pré-points fixes de F, i.e. des X tel que $F(X) \subseteq X$.
- Une fonction monotone F sur un treillis complet admet un plus grand point fixe $\mathrm{gfp}(F)$. De plus $\mathrm{gfp}(F)$ est le plus grand des post-points fixes de F, i.e. des X tel que $X \subseteq F(X)$.

La relation $\leq$ définie inductivement est $\mathrm{lfp}(F)$ où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \ldots$$

(Ou une version de F enrichie qui travaille sur des ensembles de dérivations.)

Soit $S = \{(\mathsf{Even}^\omega, \mathsf{Int}^\omega), (\mathsf{Even}, \mathsf{Int})\}$, on constate $S \subseteq F(S)$ donc $S \subseteq \mathrm{gfp}(S)$.

Points fixes

Théorème

- Une fonction monotone F sur un treillis complet admet un plus petit point fixe $\mathrm{lfp}(F)$. De plus $\mathrm{lfp}(F)$ est le plus petit des pré-points fixes de F, i.e. des X tel que $F(X) \subseteq X$.
- Une fonction monotone F sur un treillis complet admet un plus grand point fixe $\mathrm{gfp}(F)$. De plus $\mathrm{gfp}(F)$ est le plus grand des post-points fixes de F, i.e. des X tel que $X \subseteq F(X)$.

La relation $\leq$ définie inductivement est $\mathrm{lfp}(F)$ où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \dots$$

(Ou une version de F enrichie qui travaille sur des ensembles de dérivations.)

Soit $S = \{(\mathsf{Even}^\omega, \mathsf{Int}^\omega), (\mathsf{Even}, \mathsf{Int})\}$, on constate $S \subseteq F(S)$ donc $S \subseteq \mathrm{gfp}(S)$.

Prenons pour relation de sous-typage le plus grand point fixe de F ?

Plus grande relation de sous-typage

Soit $\leq$ la relation gfp(F) où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \ldots$$

- Comme $\leq \subseteq F(\leq)$ les sous-types de produits sont des produits, etc.
- De même, si $\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'$, on aura $\tau_1 \leq \tau_1'$ et $\tau_2 \leq \tau_2'$, etc.

Plus grande relation de sous-typage

Soit $\leq$ la relation gfp(F) où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \ldots$$

- Comme $\leq \subseteq F(\leq)$ les sous-types de produits sont des produits, etc.
- De même, si $\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'$, on aura $\tau_1 \leq \tau_1'$ et $\tau_2 \leq \tau_2'$, etc.
- La relation est réflexive car $S \subseteq F(S)$ pour $S = \{(\tau, \tau') \mid \tau = \tau'\}$.
  Pas besoin de règle spécifique dans la définition de F !

Plus grande relation de sous-typage

Soit $\leq$ la relation gfp(F) où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \ldots$$

- Comme $\leq \subseteq F(\leq)$ les sous-types de produits sont des produits, etc.
- De même, si $\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'$, on aura $\tau_1 \leq \tau_1'$ et $\tau_2 \leq \tau_2'$, etc.
- La relation est réflexive car $S \subseteq F(S)$ pour $S = \{(\tau, \tau') \mid \tau = \tau'\}$.
  Pas besoin de règle spécifique dans la définition de F !
- Elle est transitive car $T \subseteq F(T)$ pour $T = \{(\tau, \tau'') \mid \tau \leq \tau' \leq \tau'' \text{ pour un } \tau'\}$.
  Si $\tau'' = \tau_1'' \times \tau_2''$ on a aussi $\tau = \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2' \leq \tau_1'' \times \tau_2'' = \tau''$.
  De plus $\tau_i \leq \tau_i' \leq \tau_i''$ donc $(\tau_i, \tau_i'') \in T$ et $(\tau, \tau'') \in F(T)$.

Plus grande relation de sous-typage

Soit $\leq$ la relation gfp(F) où

$$F = \mathcal{R} \mapsto \{(\mathsf{T}, \mathsf{T}') \mid \mathsf{T} \leq^B \mathsf{T}'\} \cup \{(\tau_1 \times \tau_2, \tau_1' \times \tau_2') \mid (\tau_1, \tau_1') \in \mathcal{R} \text{ et } (\tau_2, \tau_2') \in \mathcal{R}\} \cup \dots$$

- Comme $\leq \subseteq F(\leq)$ les sous-types de produits sont des produits, etc.
- De même, si $\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'$, on aura $\tau_1 \leq \tau_1'$ et $\tau_2 \leq \tau_2'$, etc.
- La relation est réflexive car $S \subseteq F(S)$ pour $S = \{(\tau, \tau') \mid \tau = \tau'\}$.
  Pas besoin de règle spécifique dans la définition de F !
- Elle est transitive car $T \subseteq F(T)$ pour $T = \{(\tau, \tau'') \mid \tau \leq \tau' \leq \tau'' \text{ pour un } \tau'\}$.
  Si $\tau'' = \tau_1'' \times \tau_2''$ on a aussi $\tau = \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2' \leq \tau_1'' \times \tau_2'' = \tau''$.
  De plus $\tau_i \leq \tau_i' \leq \tau_i''$ donc $(\tau_i, \tau_i'') \in T$ et $(\tau, \tau'') \in F(T)$.

Pourquoi ne pas avoir pris gfp(F) quand les types étaient finis ? Cela n'aurait rien changé.

Avec des arbres

On dira simplement que $\leq$ est définie coinductivement par les règles

$$\frac{\mathsf{T} \ \leq^B \mathsf{T'}}{\mathsf{T} \leq \mathsf{T'}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'} \qquad \frac{\tau_1 \leq \tau_1' \quad \tau_2 \leq \tau_2'}{\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'} \qquad \text{etc.}$$

pour dire qu'on considère des arbres de dérivation infinis construits avec ces règles.

(Pour faire le lien avec le slide précédent, enrichir F avec des dérivations finies ou infinies.)

On pourra faire des preuves par coinduction sans revenir aux post-points fixes : il suffit d'exhiber un procédé récursif productif pour construire les dérivations souhaitées.

- On construit pour tout τ une dérivation de $\tau \leq \tau$.
  Si $\tau = \tau_1 \times \tau_2$ on applique la règle du produit et on continue...
- On prouve $\mathsf{Even}^\omega \leq \mathsf{Int}^\omega$ par coinduction, en appliquant la règle du produit, en utilisant $\mathsf{Even} \leq \mathsf{Int}$ à gauche, et l'hypothèse de coinduction à droite.

Types pas trop infinis

Definition

Un arbre infini est régulier quand il n'a qu'un nombre fini de sous-arbres différents.

Exemple de deux types réguliers : $\tau = \mathsf{Even} \times \tau'$ et $\tau' = \mathsf{Int} \times \tau$.

Definition

On étend les types finis par la construction $\mu\alpha.\tau$ pour représenter les types récursifs, en imposant qu'il y aie toujours un constructeur de type entre une variable et son lieur.
On peut alors définir, par coinduction, l'expansion d'un type fini en un type infini :

- $[\![\tau]\!] = \tau$ quand τ est une variable ou un type de base ;
- $[\![\mu\alpha.\tau]\!] = [\![\tau\{\alpha \mapsto \mu\alpha.\tau\}]\!]$, $[\![\tau \times \tau']\!] = [\![\tau]\!] \times [\![\tau']\!]$, etc.

Théorème

Tout type infini régulier est égal à l'expansion $[\![\tau]\!]$ d'un type fini τ (avec constructions μ).

Sous-typage sur des types finis récursifs

On ne considère plus que des types finis, avec la construction μ.
On reprend la définition coinductive du sous-typage, avec les règles

$$\frac{\mathsf{T} \;\leq^B \mathsf{T'}}{\mathsf{T} \leq \mathsf{T'}} \qquad \frac{\tau_1' \leq \tau_1 \quad \tau_2 \leq \tau_2'}{\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'} \quad \dots \quad \text{et} \quad \frac{\tau\{\alpha \mapsto \mu\alpha.\tau\} \leq \tau'}{\mu\alpha.\tau \leq \tau'} \qquad \frac{\tau \leq \tau'\{\alpha \mapsto \mu\alpha.\tau'\}}{\tau \leq \mu\alpha.\tau'}$$

Exemple: $\tau = \mu\alpha.\mathsf{Even} \times (\mathsf{Nat} \times \alpha)$ et $\tau' = \mu\alpha.\mathsf{Nat} \times \alpha$.

$$\dfrac{\dfrac{}{\mathsf{Even} \leq \mathsf{Nat}} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{Nat} \leq \mathsf{Nat}} \quad \dfrac{\vdots}{\tau \leq \tau'}}{\mathsf{Nat} \times \tau \leq \mathsf{Nat} \times \tau'}}{\mathsf{Nat} \times \tau \leq \tau'}}{\dfrac{\mathsf{Even} \times (\mathsf{Nat} \times \tau) \leq \mathsf{Nat} \times \tau'}{\dfrac{\mathsf{Even} \times (\mathsf{Nat} \times \tau) \leq \tau'}{\tau \leq \tau'}}}$$

Dérivations pas trop infinies

Seul un nombre fini de types peut apparaître dans nos dérivations.
Les dérivations ne sont pas forcément régulières, mais peuvent être régularisées :
si un $\tau \leq \tau'$ apparaît deux fois sur une branche, dériver la deuxième comme la première.

Il suffit de chercher des dérivations de sous-typage régulières.

- L'existence d'une dérivation régulière est décidable.
- Une stratégie est de construire une dérivation (il n'y a pas de choix important) en s'arrêtant (succès) dès qu'on rencontre un jugement $\tau \leq \tau'$ déja apparu plus bas.
- On peut faire exponentiellement mieux... je vous laisse chercher, en travaillant sur des arbres infinis ou sur le calcul d'un post-point fixe contenant la relation à vérifier.

Types iso-récursifs

Les types récursifs qu'on a vu précédemment sont dits equi-récursifs :
un type est équivalent à son déroulage dans le système de type.

On peut aussi considérer des types iso-récursifs :
il y aura seulement isomorphisme entre un type et son déroulage.

$$\frac{\Gamma \vdash u : \tau\{\alpha \mapsto \mu\alpha.\tau\}}{\Gamma \vdash \text{fold } u : \mu\alpha.\tau} \qquad \frac{\Gamma \vdash u : \mu\alpha.\tau}{\Gamma \vdash \text{unfold } u : \tau\{\alpha \mapsto \mu\alpha.\tau\}} \qquad \text{unfold } (\text{fold } u) \rightsquigarrow u$$

Types iso-récursifs

Les types récursifs qu'on a vu précédemment sont dits equi-récursifs :
un type est équivalent à son déroulage dans le système de type.

On peut aussi considérer des types iso-récursifs :
il y aura seulement isomorphisme entre un type et son déroulage.

$$\frac{\Gamma \vdash u : \tau\{\alpha \mapsto \mu\alpha.\tau\}}{\Gamma \vdash \text{fold } u : \mu\alpha.\tau} \qquad \frac{\Gamma \vdash u : \mu\alpha.\tau}{\Gamma \vdash \text{unfold } u : \tau\{\alpha \mapsto \mu\alpha.\tau\}} \qquad \text{unfold } (\text{fold } u) \rightsquigarrow u$$

Avec des types iso-récursifs, on a moins de sous-typage : $\mu\alpha.\text{Even} \times (\text{Int} \times \alpha) \not\leq \mu\alpha.\text{Int} \times \alpha$.

Types iso-récursifs

Les types récursifs qu'on a vu précédemment sont dits equi-récursifs :
un type est équivalent à son déroulage dans le système de type.

On peut aussi considérer des types iso-récursifs :
il y aura seulement isomorphisme entre un type et son déroulage.

$$\frac{\Gamma \vdash u : \tau\{\alpha \mapsto \mu\alpha.\tau\}}{\Gamma \vdash \mathsf{fold}\ u : \mu\alpha.\tau} \qquad \frac{\Gamma \vdash u : \mu\alpha.\tau}{\Gamma \vdash \mathsf{unfold}\ u : \tau\{\alpha \mapsto \mu\alpha.\tau\}} \qquad \mathsf{unfold}\ (\mathsf{fold}\ u) \rightsquigarrow u$$

Avec des types iso-récursifs, on a moins de sous-typage : $\mu\alpha.\mathsf{Even} \times (\mathsf{Int} \times \alpha) \not\leq \mu\alpha.\mathsf{Int} \times \alpha$.

La préservation du typage exige que $\mu\alpha.\tau \leq \mu\alpha.\tau'$ entraîne $\tau\{\alpha \mapsto \mu\alpha.\tau\} \leq \tau'\{\alpha \mapsto \mu\alpha.\tau'\}$.
On peut prendre une relation de sous-typage coinductive où les déroulages sont synchronisés :

$$\frac{\tau\{\alpha \mapsto \mu\alpha.\tau\} \leq \tau'\{\alpha \mapsto \mu\alpha.\tau'\}}{\mu\alpha.\tau \leq \mu\alpha.\tau'}$$

Types iso-récursifs, dérivations de sous-typage finies

On revient à des arbres de dérivation finis. Première tentative :

$$\frac{\tau \leq \tau'}{\mu\alpha.\tau \leq \mu\alpha.\tau'} \qquad \frac{}{\alpha \leq \alpha}$$

Exemple

On dériverait $\tau = \mu\alpha.\big((\alpha \to \mathbb{N}^{+}) \times (\alpha \to \mathbb{Z})\big) \leq \mu\alpha.\big((\alpha \to \mathbb{Z}) \times (\alpha \to \mathbb{Z})\big) = \tau'$.
C'est incorrect : considérer fold $\langle \lambda x.4, \lambda x.\sqrt{(\pi_1(\mathsf{unfold} x))x}\rangle$: τ et fold $\langle \lambda x.-4, \lambda x.0\rangle$: τ'.

Types iso-récursifs, dérivations de sous-typage finies

On revient à des arbres de dérivation finis. Première tentative :

$$\frac{\tau \leq \tau'}{\mu\alpha.\tau \leq \mu\alpha.\tau'} \qquad \frac{}{\alpha \leq \alpha}$$

Exemple

On dériverait $\tau = \mu\alpha.((\alpha \to \mathbb{N}^+) \times (\alpha \to \mathbb{Z})) \leq \mu\alpha.((\alpha \to \mathbb{Z}) \times (\alpha \to \mathbb{Z})) = \tau'$.
C'est incorrect : considérer fold $\langle \lambda x.4, \lambda x.\sqrt{(\pi_1(\mathsf{unfold}\, x))x} \rangle : \tau$ et fold $\langle \lambda x.-4, \lambda x.0 \rangle : \tau'$.

Une solution qui exprime la bonne hypothèse sur les variables liées :

$$\frac{}{\Sigma, \alpha \leq \beta \vdash \alpha \leq \beta} \qquad \frac{\Sigma, \alpha \leq \beta \vdash \tau \leq \tau'}{\Sigma \vdash \mu\alpha.\tau \leq \mu\beta.\tau'}$$