Types intersection — Intersection types

Examen final du cours MPRI 2-4 / Final exam for MPRI 2-4 course

2012/03/06 — Durée / Duration: 2h45

Le but de l'examen est d'étudier l'extension du lambda-calcul simplement typé par des types intersections.
La syntaxe des termes est standard, tandis que les types sont étendus par un connecteur d'intersection :

$$\tau ::= \alpha \mid \tau \to \tau \mid \tau \wedge \tau$$

La sémantique opérationnelle du langage s'appuie sur la notion de réduction β qui se réduit sous tout contexte :

$$(\beta) \qquad (\lambda x.a)b \longrightarrow a\{b/x\} \qquad\qquad \mathcal{C} ::= [] \mid a\,\mathcal{C} \mid \mathcal{C}\,a \mid \lambda x.\mathcal{C} \qquad\qquad \frac{a \longrightarrow b}{\mathcal{C}[a] \longrightarrow \mathcal{C}[b]}$$

Dans la suite, on pourra utiliser, sans les prouver de nouveau, toutes les propriétés vues en cours sur ce langage.
Le comportement des intersections est caractérisé par la relation de sous-typage suivante :

$$\frac{}{\tau \le \tau}\ \text{REFL} \qquad \frac{}{\tau_1 \wedge \tau_2 \le \tau_1}\ \text{INCL}_L \qquad \frac{}{\tau_1 \wedge \tau_2 \le \tau_2}\ \text{INCL}_R$$

$$\frac{\tau \le \tau_1 \qquad \tau \le \tau_2}{\tau \le \tau_1 \wedge \tau_2}\ \text{GLB} \qquad\qquad \frac{\tau_1 \le \tau_2 \qquad \tau_2 \le \tau_3}{\tau_1 \le \tau_3}\ \text{TRANS}$$

Comme d'habitude, cette relation de sous-typage est injectée dans le système de type à l'aide d'une règle de subsomption. Pour que le système soit complet, il est nécessaire d'y ajouter les règles d'introduction des intersections :

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{Ax} \qquad \frac{\Gamma, x : \tau_0 \vdash a : \tau_1}{\Gamma \vdash \lambda x.\, a : \tau_0 \to \tau_1}\ \to\text{Intro} \qquad \frac{\Gamma \vdash a_1 : \tau_2 \to \tau_1 \qquad \Gamma \vdash a_2 : \tau_2}{\Gamma \vdash a_1\, a_2 : \tau_1}\ \to\text{Elim}$$

$$\frac{\Gamma \vdash a : \tau_1 \qquad \Gamma \vdash a : \tau_2}{\Gamma \vdash a : \tau_1 \wedge \tau_2}\ \wedge\text{Intro} \qquad\qquad \frac{\Gamma \vdash a : \tau_1 \qquad \tau_1 \le \tau_2}{\Gamma \vdash a : \tau_2}\ \text{Subsumption}$$

On supposera que les environnements de typage Γ contiennent des variables toutes distinctes. Ainsi, on les utilisera librement comme des fonctions à support fini dont on notera "$\Gamma\,(x)$" l'évaluation en la variable x lorsque x est dans le support de Γ.

The goal of this exam is to study an extension of simply typed lambda calculus with intersection types.
The syntax of terms is the usual one, while simple types are extended by an intersection connective:

$$a ::= x \mid a\,a \mid \lambda x.a$$

The operational semantics is the one of pure λ-calculus: we have the notion of reduction β that can be reduced in any context:

In the sequel, it is allowed to reuse directly all the results about this language that were given in the course.
The behaviour of intersections is characterized by a subtyping relation:

As customary, subtyping is embedded in the typing relation via a subsumption rule. To complete the type system we also need an introduction rule for intersections:

We will assume that typing environments Γ contain distinct variables. That way, they will be freely used as functions of finite support. We will write "$\Gamma\,(x)$" to look for the image of the variable x through Γ, if x is in the support of Γ.

1 Échauffement / Warm up

Question 1

Donnez une dérivation pour :

Give a derivation for:

$$\emptyset \vdash \lambda x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to \alpha \to \beta)$$

Answer:

$$\dfrac{\dfrac{x : \alpha \vdash x : \alpha}{\emptyset \vdash \lambda x.\, x : \alpha \to \alpha}\,(\to\text{Intro}) \qquad \dfrac{x : \alpha \to \beta \vdash x : \alpha \to \beta}{\emptyset \vdash \lambda x.\, x : (\alpha \to \beta) \to \alpha \to \beta}\,(\to\text{Intro})}{\emptyset \vdash \lambda x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to \alpha \to \beta)}\,(\wedge\text{Intro})$$

Question 2

Remarquez qu'il n'y a pas de règles d'élimination pour l'intersection. En fait, ces règles peuvent être *dérivées* (dans le sens défini dans le cours). Montrez donc que les règles suivantes sont dérivables :

Notice that there is no elimination rules for intersection. Actually, these rules can be *derived* (as defined in the course). Hence, show that the following rules are derived:

$$\dfrac{\Gamma \vdash a : \tau_1 \wedge \tau_2}{\Gamma \vdash a : \tau_1}\,\wedge\text{Elim}_L \qquad \dfrac{\Gamma \vdash a : \tau_1 \wedge \tau_2}{\Gamma \vdash a : \tau_2}\,\wedge\text{Elim}_R$$

Answer: These rules are special cases of the subsumption rule:

$$\dfrac{\Gamma \vdash a : \tau_1 \wedge \tau_2 \qquad \dfrac{}{\tau_1 \wedge \tau_2 \leq \tau_1}\,(\text{INCL}_L)}{\Gamma \vdash a : \tau_1}\,(\text{SUBSUMPTION})$$

$$\dfrac{\Gamma \vdash a : \tau_1 \wedge \tau_2 \qquad \dfrac{}{\tau_1 \wedge \tau_2 \leq \tau_2}\,(\text{INCL}_R)}{\Gamma \vdash a : \tau_2}\,(\text{SUBSUMPTION})$$

Question 3

Le système de type formé par les cinq règles de déduction précédentes n'est pas algorithmique. Comme d'habitude, la règle de (SUBSUMPTION) pose problème. Est-ce l'unique règle problématique ? Justifiez votre réponse.

The typing system formed by the five deduction rules is not algorithmic. As usual, one reason resides in the presence of the (SUBSUMPTION) rule. Is this the only problematic rule? Explain.

Answer: No, (∧INTRO) is problematic as well, since it makes the system not to be syntax-directed.

Question 4

Grâce aux types intersections, il est possible de typer l'auto-application sans avoir recours à la généralisation du *let* (comme en ML). Cette propriété peut être illustrée en utilisant la règle d'application suivante à la place de celle présentée plus haut :

Thanks to intersection types, it is possible to type auto-application without resorting to let-generalization (as in ML). This can be illustrated by using the following application rule instead of the one given above:

$$\frac{\Gamma_1 \vdash a_1 : \tau_2 \to \tau_1 \qquad \Gamma_2 \vdash a_2 : \tau_2}{\Gamma_1 \wedge \Gamma_2 \vdash a_1\, a_2 : \tau_1}\ {\to_\wedge}\textsc{Elim}$$

où

$$(\Gamma_1 \wedge \Gamma_2)(x) = \begin{cases} \tau_1 & \text{if } \Gamma_1(x) = \tau_1 \text{ and } \Gamma_2(x) \text{ is undefined.} \\ \tau_2 & \text{if } \Gamma_2(x) = \tau_2 \text{ and } \Gamma_1(x) \text{ is undefined.} \\ \tau_1 \wedge \tau_2 & \text{if } \Gamma_1(x) = \tau_1 \text{ and } \Gamma_2(x) = \tau_2. \end{cases}$$

Donnez une dérivation montrant qu'il existe τ tel que $\emptyset \vdash \lambda y.\, y\, y : \tau$

1. en utilisant la règle $(\to_\wedge \textsc{Elim})$;
2. en utilisant la règle $(\to \textsc{Elim})$.

where

Give a derivation showing that there exists τ such that $\emptyset \vdash \lambda y.\, y\, y : \tau$

1. using the rule $(\to_\wedge \textsc{Elim})$;
2. using the rule $(\to \textsc{Elim})$.

Answer:
1.

$$\dfrac{\dfrac{\dfrac{}{y : \alpha \to \beta \vdash y : \alpha \to \beta}\,(\textsc{Ax}) \qquad \dfrac{}{y : \alpha \vdash y : \alpha}\,(\textsc{Ax})}{y : (\alpha \to \beta) \wedge \alpha \vdash y\, y : \beta}\,(\to_\wedge \textsc{Elim})}{\emptyset \vdash \lambda y.\, y\, y : ((\alpha \to \beta) \wedge \alpha) \to \beta}\,(\to \textsc{Intro})$$

2. In order to derive the judgment $y : (\alpha \to \beta) \wedge \alpha \vdash y\, y : \beta$, we use two applications of the subsumption rule, followed by an application of the rule $(\to \textsc{Elim})$ and then apply the $(\to \textsc{Elim})$ as above:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \tau}\,(\textsc{Ax}) \qquad \dfrac{}{\tau \le \alpha \to \beta}\,(\textsc{incl}_L) }{\Gamma \vdash y : \alpha \to \beta}\,(\textsc{sub}) \qquad \dfrac{\dfrac{}{\Gamma \vdash y : \tau}\,(\textsc{Ax}) \qquad \dfrac{}{\tau \le \alpha}\,(\textsc{incl}_R)}{\Gamma \vdash y : \alpha}\,(\textsc{sub})}{\Gamma \vdash yy : \beta}\,(\to \textsc{Elim})$$

where $(\alpha \to \beta) \wedge \alpha$ is $(\alpha \to \beta) \wedge \alpha$ and Γ is $y : (\alpha \to \beta) \wedge \alpha$.

Question 5

On admet les deux lemmes suivants :

Lemma 1 (Affaiblissement) *Si $\Gamma \vdash a : \tau$ est dérivable, et $x \notin dom(\Gamma)$, alors $\Gamma, x : \tau' \vdash a : \tau$ est aussi dérivable.*

Lemma 2 (Échange) *Si $\Gamma, x : \tau_1, y : \tau_2 \vdash a : \tau$ est dérivable, alors $\Gamma, y : \tau_2, x : \tau_1 \vdash a : \tau$ est aussi dérivable.*

We assume the following two Lemmas:

Lemma 3 (Weakening) *If $\Gamma \vdash a : \tau$ is derivable, and $x \notin dom(\Gamma)$, then $\Gamma, x : \tau' \vdash a : \tau$ is derivable, as well.*

Lemma 4 (Exchange) *If $\Gamma, x : \tau_1, y : \tau_2 \vdash a : \tau$ is derivable, then $\Gamma, y : \tau_2, x : \tau_1 \vdash a : \tau$ is derivable, as well.*

1. Prouvez le lemme suivant

   Lemma 5 *Si $\Gamma, x : \tau_1 \vdash a : \tau$ est prouvable et $\tau_2 \le \tau_1$, alors $\Gamma, x : \tau_2 \vdash a : \tau$ est aussi prouvable.*

   Prove the following lemma

   Lemma 6 *If $\Gamma, x : \tau_1 \vdash a : \tau$ is provable and $\tau_2 \le \tau_1$, then $\Gamma, x : \tau_2 \vdash a : \tau$ is provable as well.*

2. Utilisez le lemme précédent pour prouver que la règle ($\rightarrow_\wedge$Elim) est *admissible* dans le système de type initial.

Use the previous lemma to show that the rule ($\rightarrow_\wedge$Elim) is *admissible* in the initial type system.

Answer:

We have to rewrite typing derivations by replacing typing environments of the form Γ_1 with stronger ones of the form $\Gamma_1 \wedge \Gamma_2$. We first show the following simple strengthening lemma (first part of the question):

Lemma 7 *If $\Gamma, x : \tau_1 \vdash a : \tau$ is derivable and $\tau_2 \leq \tau_1$, then $\Gamma, x : \tau_2 \vdash a : \tau$ is derivable as well.*

Proof. *This lemma is proved by a simple induction on the terms of the language by a case analysis on the last applied rule:*

▷ Case (Ax). *The hypothesis is $\Gamma, x : \tau_1 \vdash y : \tau$. By inversion, $y : \tau \in (\Gamma, x : \tau_1)$. If $x \neq y$ we have $y : \tau \in \Gamma$ and a fortiori $y : \tau \in \Gamma, x : \tau_2$. So, by (Ax), $\Gamma, x : \tau_2 \vdash y : \tau$. Otherwise, $x = y$ and τ_1 must be τ. By Rule (Ax), we have $\Gamma, x : \tau_2 \vdash x : \tau_2$, which entails $\Gamma, x : \tau_2 \vdash x : \tau_1$ by subsumption since $\tau_2 \leq \tau_1$.*

▷ Case ($\rightarrow$Intro). *The hypothesis is $\Gamma, x : \tau_1 \vdash \lambda y.\, a : \tau_0 \rightarrow \tau$. By inversion, we have $\Gamma, x : \tau_1, y : \tau_0 \vdash a : \tau$. By exchange, we have $\Gamma, y : \tau_0, x : \tau_1 \vdash a : \tau$. By induction hypothesis, $\Gamma, y : \tau_0, x : \tau_2 \vdash a : \tau$ holds. Again, we exchange bindings inside typing environment to get $\Gamma, x : \tau_2, y : \tau_0 \vdash a : \tau$. By ($\rightarrow$Intro), we obtain $\Gamma, x : \tau_2 \vdash \lambda y.\, a : \tau_0 \rightarrow \tau$.*

▷ Cases ($\rightarrow$Elim), ($\wedge$Intro), (subsumption). *By immediate induction.*

Qed.

For the second part of the question it is then easy to extend this property to arbitrary typing environment strengthening:

Lemma 8 *If $\Gamma_1 \vdash a : \tau$ is derivable then $\Gamma_1 \wedge \Gamma_2 \vdash a : \tau$ is derivable as well.*

Proof. *By induction over Γ_2, using the previous lemma and the fact that $(\Gamma_1 \wedge \Gamma_2)(x) \leq \Gamma_1(x)$ when x is the domains of both Γ_1 and Γ_1 and using weakening when x is only in the domain of Γ_2.*

Qed.

Similarly, we have:

Lemma 9 *If $\Gamma_1 \vdash a : \tau$ is derivable then $\Gamma_2 \wedge \Gamma_1 \vdash a : \tau$ is derivable as well.*

We now show that rule ($\rightarrow_\wedge$Elim) is admissible. Assume $\Gamma_1 \vdash a_1 : \tau_2 \rightarrow \tau_1$ (H1) and $\Gamma_2 \vdash a_2 : \tau_2$ (H2). Let us show that $\Gamma_1 \wedge \Gamma_2 \vdash a_1\, a_2 : \tau_1$ (C). Lemma 8 applied to (H1) gives $\Gamma_1 \wedge \Gamma_2 \vdash a_1 : \tau_2 \rightarrow \tau_1$, while Lemma 9 applied to (H2) gives $\Gamma_1 \wedge \Gamma_2 \vdash a_2 : \tau_2$. Thus, (C) follows by an application of rule ($\rightarrow$Elim).

2 Properties

Une propriété fondamentale de ce système de type avec type intersection, *que l'on admettra*, est qu'un terme du λ-calcul est typable si, et seulement si, il est fortement normalisant.

A fundamental property of this intersection type system, *which we admit*, is that a term of the λ-calculus is typable if and only if it is strongly normalizing.

Question 6

La règle ($\rightarrow_\wedge$Elim) donne une bonne intuition des raisons pour lesquelles tout terme fortement normalisant est typable avec des types intersections. Pouvez-vous expliquer cette intuition ?

The rule [$\rightarrow_\wedge$Elim] gives a good intuition of why strongly normalizing terms are typable by intersection types. Can you explain this intuition?

Answer:

Consider a strongly normalizing term and its subterms. Consider all possible reductions. Since every reduction is finite, then every subterm is applied only to finitely many different contexts. In each context it may need to be typed by a different arrow type. But since these contexts are finite then we can take the intersection of all these function types. Then it suffices to type the subterm with the intersection of all these types to deduce a typing for the whole term.

Question 7

Est-ce que le typage est décidable dans ce système de type avec intersection ?

Is typability decidable in this intersection type system?

Answer: Since only strongly normalizing terms are typable then deciding typability is equivalent to deciding termination in λ-calculus, which we know is undecidable.

Les types intersections sont similaires aux schémas de type de ML. Les programmes ont des types principaux et la règle d'instantiation, pierre angulaire du polymorphisme de ML, est admissible :

Intersection types are quite akin to ML-types: λ-terms have principal types and the rule of instantiation, the cornerstone of ML-polymorphism, is admissible in the intersection type system:

$$\frac{\Gamma \vdash a : \tau \qquad \alpha_i \# \Gamma \qquad (i \in I)}{\Gamma \vdash a : \tau\{\tau_i/\alpha_i\}_{i\in I}} \text{ INSTANCE}$$

où $\tau\{\tau_i/\alpha_i\}_{i\in I}$ représente la substitution simultanée des types τ_i pour les variables de type α_i, en supposant que ces variables n'apparaissent pas dans Γ (*i.e.*, $\alpha_i \# \Gamma$).

where $\tau\{\tau_i/\alpha_i\}_{i\in I}$ denotes the simultaneous substitution of types τ_i for the type variable α_i, provided that none of these type variables occur in Γ (*i.e.*, $\alpha_i \# \Gamma$).

Question 8

Donnez une dérivation pour le jugement de la question 1 qui utilise la règle (Instance) ci-dessus, ainsi que deux instances différentes de la dérivation de typage D suivante :

Give a derivation for the judgement in Question 1, that uses the (Instance) rule above as well as two different instances of the following typing derivation D:

$$\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\,(\text{Ax})}{\emptyset \vdash \lambda x.\, x : \alpha \to \alpha}\,(\rightarrow\text{Intro})$$

Answer:

$$\dfrac{D \quad \dfrac{D}{\emptyset \vdash \lambda x.x : (\alpha \to \beta) \to \alpha \to \beta}\,(\text{Instance})}{\emptyset \vdash \lambda x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to \alpha \to \beta)}\,(\wedge\text{Intro})$$

En fait, un schéma de type de ML de la forme $\forall\alpha.\tau$ peut être vu comme une intersection infinie de toutes les instances de la forme $\tau\{\tau'/\alpha\}$ pour un certain type τ'.

Cela pourrait laisser à penser que les schémas de types de ML sont plus expressifs que les types intersections. En vérité, c'est le contraire qui est vrai : les types intersections sont plus expressifs que les schémas de ML car ils typent strictement plus de termes.

As a matter of fact, an ML type schema $\forall\alpha.\tau$ is nothing but an infinite intersection of all the instances of the form $\tau\{\tau'/\alpha\}$ for some type τ'.

This may give the impression that ML-types can express more than intersection types. Actually it is the contrary: intersection types are more expressive than ML-types since they type strictly more terms.

Question 9

Donnez un terme qui est typable avec des types intersections mais qui n'est pas typable dans ML.

Give a term that is typable by intersection types but it is not typable in ML.

Answer: The term that does an auto-application, already met in question 3: $\lambda x.\, x\, x$ is not typable in ML.

Question 10

Pouvez-vous expliquer de manière intuitive pourquoi, bien que les schémas de type de ML représentent des intersections infinies, des intersections finies sont suffisantes pour accepter tous les termes typables de ML ?

Can you give the intuition why, although ML has infinite intersections, finite intersections are enough to type all the terms that are typable in ML?

Answer:

If we consider the occurrences of a polymorphic function in a program, since a program is finite then it occurs only finitely many times. So its principal type needs to be instantiated only finitely many times. It suffices to assign the intersection of all these instances to the polymorphic function to type the program.

Question 11

L'instanciation ne suffit pas à elle seule pour rendre compatible le type principal d'une fonctions avec celui de son argument. Par exemple, considérez les deux termes suivants et leurs types principaux (sous l'hypothèse que $z : \alpha$) :

Instantiation alone does not suffice to make the principal type of a function be compatible with the principal type of its argument. For instance consider the following two terms with their principal types (under the hypothesis that $z : \alpha$).

$$\begin{array}{lcll} a_1 & \equiv & \lambda x.\, x\,(\lambda y.\, y\, z) & : (((\alpha \to \beta) \to \beta) \to \delta) \to \delta \\ a_2 & \equiv & \lambda f.\, \lambda x.\, f\,(f\, x) & : ((\alpha \to \beta) \wedge (\delta \to \alpha)) \to \delta \to \beta \end{array}$$

1. Donnez la dérivation de

1. Give the derivation of

$$\emptyset \vdash a_2 : ((\alpha \to \beta) \wedge (\delta \to \alpha)) \to \delta \to \beta$$

Answer:

Let us write Γ_1 for $f : ((\alpha \to \beta) \wedge (\delta \to \alpha)), x : \delta$ and τ_f for $((\alpha \to \beta) \wedge (\delta \to \alpha))$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_1 \vdash f : \tau_f}\,(\text{Ax})}{\Gamma_1 \vdash f : \alpha \to \beta}\,(\wedge \text{E}_L) \qquad \dfrac{\dfrac{\dfrac{\Gamma_1 \vdash f : \tau_f}{\Gamma_1 \vdash f : \delta \to \alpha}\,(\wedge \text{E}_R) \qquad \dfrac{}{\Gamma_1 \vdash x : \delta}\,(\text{Ax})}{\Gamma_1 \vdash f\, x : \alpha}\,(\to \text{E})}{\Gamma_1 \vdash f\,(f\, x) : \beta}\,(\to \text{E})}{f : \tau_f \vdash \lambda x.\, f\,(f\, x) : \delta \to \beta}\,(\to \text{I})}{\emptyset \vdash a_2 : ((\alpha \to \beta) \wedge (\delta \to \alpha)) \to \delta \to \beta}\,(\to \text{I})$$

2. Il n'est pas possible d'utiliser la règle (INSTANCE) pour déduire des types de a_1 et a_2 que l'application $a_1\, a_2$ est bien typée. Pourquoi ?

2. It is not possible to use (INSTANCE) to deduce from the principal types of a_1 and a_2 that the application $a_1 a_2$ is well typed. Why?

Answer: There does not exist any substitution that unifies $(\alpha \to \beta)$ *and* $(\alpha \to \beta) \wedge (\gamma \to \alpha)$

3. Prouvez que l'application $a_1\, a_2$ est bien typée sans donner de dérivation de typage.

3. Prove that the application $a_1\, a_2$ is well typed without giving a typing derivation.

Answer: $a_1\, a_2$ *strongly normalizes into* $\lambda x.\,(x\, z)\, z$.

4.[FACULTATIF] Donnez une dérivation de typage du jugement $z : \tau' \vdash a_1\, a_2 : \tau$ pour des certains types τ et τ' à déterminer.

4. [OPTIONAL] Give a derivation for the typing judgment $z : \tau' \vdash a_1\, a_2 : \tau$ for some τ and τ' to be found.

Answer: Sketch. It is possible to prove:

$$z : \alpha_1 \wedge \alpha_2 \vdash a_1 : (\tau_1 \to \tau_2) \to \tau_2 \qquad \textit{and} \qquad \emptyset \vdash a_2 : \tau_1 \to \tau_2$$

where $\tau_1 = ((\alpha_1 \to \beta_1) \to \beta_1) \wedge ((\alpha_2 \to \alpha_1 \to \beta_1) \to \alpha_1 \to \beta_1)$ *and* $\tau_2 = (\alpha_2 \to \alpha_1 \to \beta_1) \to \beta_1$

Therefore by weakening and (→ELIM) *we obtain* $\tau = \tau_2$ *where* $\tau' = \alpha_1 \wedge \alpha_2$

Une autre propriété importante des systèmes de type à intersection est que non seulement les termes ont des *types principaux*, mais aussi des typages principaux.

Un *type principal* pour un environnement de typage Γ et un terme a est un type τ tel que $\Gamma \vdash a : \tau$ et tel que tout autre type τ' tel que $\Gamma \vdash a : \tau'$ peut-être obtenu à partir de τ par subsomption et instantiation.

Un *typage principal* pour un terme a est une paire Γ et τ telle que $\Gamma \vdash a : \tau$ et telle que toute autre paire Γ' et τ' telle que $\Gamma' \vdash a : \tau'$ peut être obtenu à partir de Γ et τ par subsomption et instanciation. ML a des types principaux mais pas des typages principaux.

Another important property with intersection types systems is that not only typable terms have a *principal type*, but also *principal typings*.

A *principal type* for a type environment Γ and a term a is a type τ such that $\Gamma \vdash a : \tau$ and such that every other type τ' such that $\Gamma \vdash a : \tau'$ can be obtained from τ by subsumption and instantiation.

A *principal typing* for a term a is a pair Γ, τ such that $\Gamma \vdash a : \tau$ and such that every other pair Γ', τ' such that $\Gamma' \vdash a : \tau'$ can be obtained from Γ and τ by subsumption and instantiation. ML has principal types but not principal typings.

Question 12

L'existence de typages principaux est une propriété importante pour pouvoir inférer de types très précis pour les termes récursifs de la forme **rec** $x.a$. Pouvez-vous expliquer pourquoi ?

Principal typing is a property important to infer very precise types for recursive terms of the form **rec** $x.a$. Can you explain why?

Answer: The typing rule for the term above is:

$$\frac{\Gamma, x : \tau \vdash a : \tau}{\Gamma \vdash \textbf{\textit{rec}}\ x.a : \tau}\ \text{REC}$$

So in the absence of an explicit typing for the recursion variable x the type system has to find the best type and type environment to type recursion.

Just for information, notice that typing is undecidable for (both ML and decidable restrictions) of intersection type systems that include the rule above.

However, Trevor Jim proved that the more restrictive rule

$$\frac{\Gamma, x : \tau' \vdash a : \tau}{\Gamma \vdash \textbf{\textit{rec}}\ x.a : \tau}\ \text{REC-SIMPLE}$$

where τ' is a monomorphic instance of τ together with rank-2 intersection types yield a decidable system.

3 Algorithms

Comme le typage en présence de types intersections est indécidable, plusieurs restrictions décidables ont été étudiées. Un cas très étudié consiste à limiter le "rang" des intersections. Par exemple, la restriction au rang 2 (dans laquelle tout chemin qui part de la racine de l'arbre de syntaxe d'un type jusqu'à une intersection passe au plus une fois à gauche d'une flèche) est connu pour être décidable tout en étant plus expressive que les schémas de type de ML.

Since typing is undecidable several restrictions have been studied to make it decidable. A well-studied variant is to limit the "rank" of intersections. For example, the rank-2 restriction (in which every path that goes to the root of the syntax tree of a type to an intersection can pass at most once on the left of an arrow) is known to be decidable and still more expressive than ML type schemes.

Dans ce sujet, nous allons étudier une autre façon d'opérer cette restriction, qui consiste à décorer les λ-abstractions par leurs types (intersections). Ainsi, nous allons considérer des termes de la forme suivante :

Here, we will study another variant, that is to decorate λ-abstractions with their (intersection) types. Hence, we consider term of the following form:

$$a ::= x \mid a\, a \mid \lambda^{\wedge_{i\in I}\tau_i\to\tau'_i} x.\, a$$

L'idée consiste à dire que la construction $\lambda^{\wedge_{i\in I}\tau_i\to\tau'_i} x.\, a$ est bien typée avec le type $\wedge_{i\in I}\tau_i \to \tau'_i$ à condition que $\lambda x.\, a$ ait le type $\tau_i \to \tau'_i$ pour tout $i \in I$.

The idea is that $\lambda^{\wedge_{i\in I}\tau_i\to\tau'_i} x.\, a$ is well-typed with type $\wedge_{i\in I}\tau_i \to \tau'_i$ if we can prove that $\lambda x.\, a$ has type $\tau_i \to \tau'_i$ *for every* $i \in I$.

Question 13

Donnez la règle de typage de :

Give the typing rule for:

$$\lambda^{\wedge_{i\in I}\tau_i\to\tau'_i} x.\, a$$

Answer:

$$\dfrac{\Gamma, x : \tau_i \vdash a : \tau'_i \qquad (\forall i \in I)}{\Gamma \vdash \lambda^{\wedge_{i\in I}\tau_i \to \tau'_i} x.\, a : \wedge_{i\in I}\tau_i \to \tau'_i} \;\to\textsc{Intro}$$

Les règles de typage des variables et des applications sont inchangées. Les règles de subsomptions et d'introduction des intersections sont également inchangées. Cependant, la règle d'instanciation n'est plus admissible. Cela veut dire que le système est plus faible. En particulier, on ne peut pas déduire comme dans la question 1 que $\lambda^{\alpha\to\alpha} x.\, x$ a le type $(\alpha \to \alpha) \wedge ((\alpha \to \beta) \to (\alpha \to \beta))$. On considère donc le système formé des règles suivantes :

The typing rules for variables and application are unchanged. The subsumption and the introduction of intersection are preserved. However, the instantiation rule is no longer admissible. This means that the system is weaker. In parcticular, we cannot deduce as in Question 1 that $\lambda^{\alpha\to\alpha} x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to (\alpha \to \beta))$. We thus consider the system composed of the following rules:

$$\textsc{Ax}\ \dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \to\textsc{Elim}\ \dfrac{\Gamma \vdash a_1 : \tau_2 \to \tau_1 \qquad \Gamma \vdash a_2 : \tau_2}{\Gamma \vdash a_1\, a_2 : \tau_1} \qquad \wedge\textsc{Intro}\ \dfrac{\Gamma \vdash a : \tau_1 \qquad \Gamma \vdash a : \tau_2}{\Gamma \vdash a : \tau_1 \wedge \tau_2}$$

$$\textsc{Subsumption}\ \dfrac{\Gamma \vdash a : \tau_1 \qquad \tau_1 \leq \tau_2}{\Gamma \vdash a : \tau_2}$$

ainsi que de la règle (→Intro) que vous avez explicitée dans la question précédente et la règle (Instance) que nous avons déjà rencontrée :

and also the (→Intro) rule that you gave in the previous question and the (Instance) rule introduced above:

$$\textsc{Instance}\ \dfrac{\Gamma \vdash a : \tau \qquad \alpha_i \# \Gamma \qquad (i \in I)}{\Gamma \vdash a : \tau\{\tau_i/\alpha_i\}_{i\in I}}$$

Question 14

Donnez une dérivation de typage pour :

Give a typing derivation for:

$$\emptyset \vdash \lambda^{\alpha\to\alpha} x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to (\alpha \to \beta))$$

Answer: If we write D for

$$\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\ (\textsc{Ax})}{\emptyset \vdash \lambda^{\alpha\to\alpha} x.\, x : \alpha \to \alpha}\ (\to\textsc{Intro})$$

the typing derivation is the same as in Question 8.

$$\dfrac{D \qquad \dfrac{D}{\emptyset \vdash \lambda^{\alpha\to\alpha} x.\, x : (\alpha \to \beta) \to \alpha \to \beta}\ (\textsc{Instance})}{\emptyset \vdash \lambda^{\alpha\to\alpha} x.\, x : (\alpha \to \alpha) \wedge ((\alpha \to \beta) \to (\alpha \to \beta))}\ (\wedge\textsc{Intro})$$

L'objectif du reste de cette section est de trouver une forme normale pour les dérivations qui correspondent à l'exécution d'un système algorithmique. Le cœur du problème est induit par les trois règles qui ne sont pas dirigées par la syntaxe, c'est-à-dire (∧Intro), (Subsumption), et (Instance).

The goal of the rest of this section is to find a normal form for derivations that correspond to the execution of an algorithmic system. The core of the problem are the three rules that are not syntax directed, namely, (∧Intro), (Subsumption), and (Instance).

La première étape consiste à montrer que nous pouvons fusionner deux de ces trois règles. En particulier, si nous remplaçons les règles (∧Intro) et (Instance) par la règle suivante :

The first step is to show that we do not need three rules but we can merge two of the three rules. In particular if we replace the rules (∧Intro) and (Instance) by the following rule

$$\frac{\Gamma \vdash a : \tau \qquad \alpha_i^j \# \Gamma \qquad (i \in I_j)(j = 1..n)}{\Gamma \vdash a : \tau\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau\{\tau_i/\alpha_i\}_{i\in I_n}} \text{Instance-}\wedge$$

alors nous obtenons un système équivalent. Ainsi, (∧Intro) et (Instance) deviennent admissibles.
Cela implique que nous n'avons pas besoin de toute la généralité de la règle (∧Intro). Cette règle est nécessaire uniquement quand elle est combinée avec la règle (Instance). En fait, dans ce système, combiner les règles (Instance) et (Subsumption) n'est pas utile comme le montre la question suivante.

then we obtain an equivalent system. That way, (∧Intro) and (Instance) become admissible.
This means that we do not need the full generality of the (∧Intro). We need it only when it is combined with the (Instance) rule. As a matter of fact in this system combining (Instance) and (Subsumption) is not useful as shown by the following question.

Question 15

Supposez que les jugements de typage suivants $\Gamma \vdash a : \tau_1$ et $\Gamma \vdash a : \tau_2$ ont été déduits en appliquant la règle de subsomption au type principal de la paire Γ et a (vous pouvez supposer que tout terme dans ce système possède un type principal). Déduisez par subsomption que $\Gamma \vdash a : \tau_1 \wedge \tau_2$

Suppose that the following typings $\Gamma \vdash a : \tau_1$ and $\Gamma \vdash a : \tau_2$ have been deduced by applying subsumption to the principal type of the pair Γ and a (you can suppose that such a principal type always exists). Deduce by subsumption that $\Gamma \vdash a : \tau_1 \wedge \tau_2$.

Answer:

Let τ be the principal type of τ under Γ. Then $\tau \leq \tau_1$ and $\tau \leq \tau_2$. Therefore we have the following deduction

$$\frac{\Gamma \vdash a : \tau \qquad \dfrac{\tau \leq \tau_1 \qquad \tau \leq \tau_2}{\tau \leq \tau_1 \wedge \tau_2}\ (\text{GLB})}{\Gamma \vdash a : \tau_1 \wedge \tau_2}\ (\text{subsumption})$$

Question 16

1. Montrez que pour toute preuve, il existe une preuve équivalente dans laquelle il n'y a jamais deux applications consécutives de la règle de subsomption.
2. Montrez que pour toute preuve, il existe une preuve équivalente dans laquelle il n'y a jamais deux applications consécutives de la règle (Instance-∧).

1. Show that for every proof there exists an equivalent proof in which there are never two consecutive applications of the subsumption rule.
2. Show that for every proof there exists an equivalent proof in which there are never two consecutive applications of the (Instance-∧) rule.

Answer:

1. *The case for two consecutive applications is a direct consequence of the transitivity of the pre-order. The case with more consecutive rules follow by induction.*

2. *Straightforward: if there are two consecutive applications of the* (Instance-$\wedge$) *rule you can merge it in just one in which the substitution is the composition of the two substitutions and the expansion is the product of the two expansions.*

Question 17

Utilisez le lemme suivant :

Lemma 10 *Si* $\tau \leq \tau'$ *alors* $\tau\{\tau_i/\alpha_i\}_{i\in I} \leq \tau'\{\tau_i/\alpha_i\}_{i\in I}$

pour prouver que pour toute preuve, il existe une preuve équivalente dans laquelle une règle de subsomption n'est jamais suivie d'une règle (instance-$\wedge$).

Use the following Lemma:

Lemma 11 *If* $\tau \leq \tau'$ *then* $\tau\{\tau_i/\alpha_i\}_{i\in I} \leq \tau'\{\tau_i/\alpha_i\}_{i\in I}$

to prove that for every proof there exists an equivalent proof in which a subsumption rule is never followed by a (instance-$\wedge$) rule.

Answer: Consider the following derivation:

$$\dfrac{\dfrac{\dfrac{D}{\Gamma \vdash a : \tau}\,(\mathrm{r}) \qquad \tau \leq \tau'}{\Gamma \vdash a : \tau'}\,(\textsc{Subsumption}) \qquad \alpha_i \# \Gamma}{\Gamma \vdash a : \tau'\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau'\{\tau_i/\alpha_i\}_{i\in I_n}}\,(\textsc{Instance-}\wedge)$$

*Using the result of the previous two questions it is possible to deduce that (r) is a structural rule (*i.e., *either* (Ax), *or* ($\to$Intro), *or* ($\to$Elim)*) and that this chunk of derivation is followed by a structural rule. So all it remains to show is that these two rules can be inverted.*

From $\tau \leq \tau'$ *and the lemma above we can deduce that* $\tau\{\tau_i/\alpha_i\}_{i\in I_j} \leq \tau'\{\tau_i/\alpha_i\}_{i\in I_j}$ *for every* $j = 1..n$*. By repeated application of the* (GLB) *we obtain* $\tau\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau\{\tau_i/\alpha_i\}_{i\in I_n} \leq \tau'\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau'\{\tau_i/\alpha_i\}_{i\in I_n}$*. Then we can deduce:*

$$\dfrac{\dfrac{\dfrac{D}{\Gamma \vdash a : \tau}\,(\mathrm{r}) \qquad \alpha_i \# \Gamma}{\Gamma \vdash a : \tau''}\,(\textsc{Instance-}\wedge) \qquad \tau'' \leq \tau'\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau'\{\tau_i/\alpha_i\}_{i\in I_n}}{\Gamma \vdash a : \tau'\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau'\{\tau_i/\alpha_i\}_{i\in I_n}}\,(\textsc{subs})$$

with $\tau'' \equiv \tau\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau\{\tau_i/\alpha_i\}_{i\in I_n}$

Question 18

En utilisant les résultats des deux exercices précédents, déduisez la forme normale des dérivations.

Using the result of the previous two exercises, deduce normal forms of derivations.

*Answer: The previous two exercises show that for every derivation there is an equivalent one where two structural rules (*i.e., *either* (Ax), *or* ($\to$Intro), *or* ($\to$Elim)*) are separated exactly by a rule* (instance-$\wedge$) *followed by a* (Subsumption) *rule (these rules may be denegerated, that is the identity subsumption or instance).*

Let use $\overline{\tau'\{\tau/\alpha\}_n}$ *as a shorthand for the term* $\tau'\{\tau_i/\alpha_i\}_{i\in I_1} \wedge \ldots \wedge \tau'\{\tau_i/\alpha_i\}_{i\in I_n}$*. Then normal form derivations will be formed by composing the following two blocks (we omitted the conditions on*

Γ*), and possibly ending by a rule* (INSTANCE-$\wedge$) *followed by a* (SUBSUMPTION) *rule.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma, x : \tau_j^1 \vdash a : \tau'_j}\,(\mathrm{Rj})}{\Gamma, x : \tau_j^1 \vdash a : \overline{\tau'_j\{\tau^j/\alpha^j\}_{n_j}} \leq \tau_j^2}\,(\textsc{Inst-}\wedge)}{\Gamma, x : \tau_j^1 \vdash a : \tau_j^2}\,(\textsc{Sub}) \qquad \forall j \in J}{\Gamma \vdash \lambda^{\wedge_{j\in J}\tau_j^1\to\tau_j^2} x.\, a : \bigwedge_{j\in J} \tau_j^1 \to \tau_j^2}\,(\to\textsc{Intro})$$

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash a_1 : \tau'}\,(\mathrm{R1})}{\Gamma \vdash a_1 : \overline{\tau'\{\tau/\alpha\}_n} \leq \tau_2 \to \tau_1}\,(\textsc{Inst-}\wedge)}{\Gamma \vdash a_1 : \tau_2 \to \tau_1}\,(\textsc{Sub}) \qquad \dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash a_2 : \tau''}\,(\mathrm{R2})}{\Gamma \vdash a_2 : \overline{\tau''\{\tau^\circ/\beta\}_m} \leq \tau_2}\,(\textsc{Inst-}\wedge)}{\Gamma \vdash a_2 : \tau_2}\,(\textsc{Sub})}{\Gamma \vdash a_1 a_2 : \tau_1}\,(\to\textsc{Elim})$$

where $R1$, $R2$, Rj *are structural rules.*

For information, as usual the normal forms of derivations suggest the corresponding algorithmic rules. In this case the algorithm is given by the rule (AX) *and the following two rules directly derived from the two blocs above:*

$$\dfrac{\Gamma, x : \tau_j^1 \vdash a : \tau'_j \qquad \exists \tau^j, n_j \text{ such that } \overline{\tau'_j\{\tau^j/\alpha^j\}_{n_j}} \leq \tau_j^2 \qquad (\forall j \in J)}{\Gamma \vdash \lambda^{\wedge_{j\in J}\tau_j^1\to\tau_j^2} x.\, a : \bigwedge_{j\in J} \tau_j^1 \to \tau_j^2}\,(\to\textsc{Intro-Alg})$$

$$\dfrac{\Gamma \vdash a_1 : \tau' \qquad \Gamma \vdash a_2 : \tau'' \qquad \exists n, m, \tau, \tau^\circ \text{ such that } \overline{\tau'\{\tau/\alpha\}_n} \leq \overline{\tau''\{\tau^\circ/\beta\}_m} \to \mathsf{Any}}{\Gamma \vdash a_1 a_2 : \overline{\tau'\{\tau/\alpha\}_n} \bullet \overline{\tau''\{\tau^\circ/\beta\}_m}}\,(\to\textsc{Elim-Alg})$$

where $\tau_1 \bullet \tau_2 \stackrel{def}{=} \min\{\tau \mid \tau_1 \leq \tau_2 \to \tau\}$ *and* Any *denotes the top type, that is, the empty intersection.*

The previous rules describe a semidecision algorithm for the typing relation since only semidecision procedures are known to check whether the $\exists$ *occurring in the rules are satisfiable. Decidability is an open problem, yet.*