Static differentiation / Différentiation statique

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

2017/03/09 — Duration / durée: 3h00

Answers are judged by their correctness, but also by their clarity, conciseness, and precision. You don't have to justify answers when not asked.

Math displays are one-column and spread over the whole page width.

Les réponses sont jugées d'après leur correction, mais aussi d'après leur clarté, leur concision et leur précision. Vous n'avez pas besoin de justifier vos réponses lorsque ce n'est pas demandé.

Les formules mathématiques sont en simple colonne et s'étendent sur toute la largeur de la page.

1 Motivations / Motivations

Let A be a set. We call a *change structure* over A, written $\widehat{A}$, the tuple $(A, \Delta, \oplus, \ominus)$ such that:

Soit A un ensemble. On appelle *structure de changements* sur A, noté $\widehat{A}$, le quadruplet $(A, \Delta, \oplus, \ominus)$ tel que :

- If $a \in A$, then $\Delta(a)$ is a set of valid changes on a.
- If $a \in A$ and $da \in \Delta(a)$ then $a \oplus da \in A$.
- If $a_1, a_2 \in A$, then $a_2 \ominus a_1 \in \Delta(a_1)$.
- *Update-fill law:* for all a_1, a_2, $a_1 \oplus (a_2 \ominus a_1) = a_2$.

In the following, we write $\Delta_{\widehat{A}}$, $\oplus_{\widehat{A}}$ and $\ominus_{\widehat{A}}$ to denote the corresponding elements of the tuple $\widehat{A}$. For conciseness, we may omit these subscripts when they can be deduced from the context. Also, if f and g are functions, the notation $f = g$ stands for extensional equality, *ie.* $(f = g) \leftrightarrow (\forall x, f\,x = g\,x)$

Dans la suite, on notera $\Delta_{\widehat{A}}$, $\oplus_{\widehat{A}}$ et $\ominus_{\widehat{A}}$ pour faire référence aux différents composants du quadruplet définissant $\widehat{A}$. Dans un souci de lisibilité, ces indices seront parfois omis lorsqu'ils se déduisent facilement du contexte. Par ailleurs, si f et g sont des fonctions, on écrira $f = g$ pour l'égalité extensionnelle *i.e.* $(f = g) \leftrightarrow (\forall x, f\,x = g\,x)$.

Question 1

In the programming language of your choice, give a typed interface that faithfully translates the above definition of a structure change. This interface must be as accurate as possible, even if that means using dependent types.

Dans le langage de programmation de votre choix, implémenter une interface typée traduisant la définition de structure de changement spécifiée ci-dessus. On s'efforcera d'écrire la spécification la plus précise possible, quitte à utiliser des types dépendants.

□

We suggest the following change structure $\widehat{\mathbb{N}}$ for the type $\mathbb{N}$ of natural numbers:

On propose la structure de changements $\widehat{\mathbb{N}}$ suivante pour le type des entiers naturels $\mathbb{N}$:

$$\begin{aligned} \Delta(n) &\mapsto \{dn : \mathbb{Z} \mid n + dn \geq 0\} \\ m \oplus dn &\mapsto m + dn \\ m \ominus n &\mapsto m - n \end{aligned}$$

Question 2

Justify the validity of this definition: prove that the operations $\oplus$ and $\ominus$ are well-defined and that the "update-fill" law is verified.

Justifiez la validité de cette définition : les opérations $\oplus$ et $\ominus$ sont bien définies et la loi "update-fill" est vérifiée.

□

Question 3

Let $\widehat{A}$ (respectively $\widehat{B}$) be a change structure over A (respectively B). Give a change structure $\widehat{A \times B}$ for the cartesian product $A \times B$. Justify the validity of your answer.

Soit $\widehat{A}$ (respectivement $\widehat{B}$) une structure de changements sur A (respectivement B). Proposez une structure de changements $\widehat{A \times B}$ pour le produit cartésien $A \times B$. Justifiez la validité de votre réponse.

□

Question 4

Let $\widehat{A}$ (respectively $\widehat{B}$) be a change structure over A (respectively B). Give a change structure $\widehat{A + B}$ for the disjoint union $A + B$. It is not necessary to justify the validity of your answer.

Soit $\widehat{A}$ (respectivement $\widehat{B}$) une structure de changements sur A (respectivement B). Proposez une structure de changements $\widehat{A + B}$ pour l'union disjointe $A + B$. Il n'est pas nécessaire de justifier la validité de votre réponse.

□

Let f be a function of type $A \to B$. We call *derivative of f* any term f' such that

Soit f une fonction de $A \to B$. On appelle *dérivée de f* tout terme f' tel que

$$f\,(a \oplus_{\widehat{A}} da) = f\,a \oplus_{\widehat{B}} f'\,a\,da$$

where $a \in A$ and $da \in \Delta(a)$.
Put otherwise, the binary function f' computes the update to apply to the result of the computation f to take into account a change da of its input.

où $a \in A$ et $da \in \Delta(a)$.
En d'autres termes, la fonction binaire f' calcule la mise-à-jour à appliquer au résultat du calcul effectué par f pour prendre en compte une variation da de son entrée.

Question 5

Give a derivative of the function $\mathsf{sum} : \mathbb{N} \to \mathbb{N}$ with $\mathsf{sum}\ n = \Sigma_{k=0}^{n} k$. Provide its type signature, following the interface specified in Question 1.

Proposez une dérivée pour la fonction $\mathsf{sum} : \mathbb{N} \to \mathbb{N}$ où $\mathsf{sum}\ n = \Sigma_{k=0}^{n} k$. On précisera le type de cette fonction en utilisant l'interface spécifiée en Question 1.

□

In the following, we shall write $\mathbf{0}_a$ to denote the null change over a value a, that is $a \ominus a$.

Dans la suite, on notera $\mathbf{0}_a$ pour dénoter le changement nul sur la valeur a, c'est-à-dire $a \ominus a$.

2 Change structure for functions / Structure de changements pour les fonctions

Let $\widehat{A}$ (respectively $\widehat{B}$) be a change structure over A (respectively B). We define a change structure $\widehat{A \to B}$ over functions of type $A \to B$ by the tuple $(A \to B, \Delta, \oplus, \ominus)$ such that

Soit $\widehat{A}$ (respectivement $\widehat{B}$) une structure de changements sur A (respectivement B). On définit une structure de changements $\widehat{A \to B}$ pour les fonctions de type $A \to B$ par le quadruplet $(A \to B, \Delta, \oplus, \ominus)$ tel que :

- $\Delta(f)$, for $f \in A \to B$, is defined as the set of functions $df \in (a : A) \to \Delta_{\widehat{A}}(a) \to \Delta_{\widehat{B}}(f\,a)$ that verify the identity $f\,a \oplus_{\widehat{B}} df\,a\,da = f\,(a \oplus_{\widehat{A}} da) \oplus_{\widehat{B}} df\,(a \oplus_{\widehat{A}} da)\,\mathbf{0}_{a \oplus_{\widehat{A}} da}$
- $(f \oplus df)\,a = f\,a \oplus_{\widehat{B}} df\,a\,\mathbf{0}_a$
- $(g \ominus f)\,a\,da = g\,(a \oplus da) \ominus_{\widehat{B}} f\,a$

Question 6

Prove that $g \ominus f \in \Delta(f)$.

Montrez que $g \ominus f \in \Delta(f)$.

□

Question 7

Justify the validity of the change structure $\widehat{A \to B}$.

Justifiez la validité de la structure de changements $\widehat{A \to B}$.

□

Question 8

Prove that $(f \oplus df)\,(a \oplus da) = f\,a \oplus df\,a\,da$

Montrez que $(f \oplus df)\,(a \oplus da) = f\,a \oplus df\,a\,da$

□

Question 9

Prove that, for any function f, $\mathbf{0}_f$ is a derivative of f.

Montrez que pour toute fonction f, $\mathbf{0}_f$ est une dérivée de f.

□

3 Weak typing of changes / Typage faible des changements

Whereas operations over change structures are naturally specified with dependent types, it can be arduous to ensure that such operations are used in a well-typed manner, which hinders the scalability of this approach.

Si les opérations des structures de changements sont naturellement spécifiées avec des types dépendants, le bon typage des termes utilisant des structures de changements peut s'avérer assez complexe à établir, limitant ainsi le passage à l'échelle de cette approche.

Question 10

Let n be a natural number. On what conditions is the following change well-typed according to the interface introduced in Question 1 and its instantiation over natural numbers introduces in Question 2?

Soit n un entier naturel. Sous quelles conditions le changement suivant est-il bien typé d'après l'interface introduite en Question 1 et son instanciation pour les entiers naturels introduite en Question 2 ?

$$(\mathsf{sum}\, n) \oplus (-42)$$

□

We therefore study a weakly-typed treatment of incremental programs, focusing on the application of change structures in the simply-typed λ-calculus.

On étudie donc une version plus faible du typage des programmes incrémentaux en s'intéressant à l'usage des structures de changements dans le λ-calcul simplement typé.

We first define the syntax of simple types and contexts:

Pour commencer, on se donne une syntaxe pour les types simples et les contextes :

$$\begin{array}{rcll} \tau & ::= & \iota & \text{Base types} \\ & | & \tau \to \tau & \text{Functional types} \end{array} \qquad \begin{array}{rcll} \Gamma & ::= & \epsilon & \text{Empty context} \\ & | & \Gamma, x : \tau & \text{Context extension} \end{array}$$

as well as a type family $\nabla : \star \Rightarrow \star$ defined as follow:

ainsi qu'une famille de types $\nabla : \star \Rightarrow \star$ définie comme suit :

$$\begin{array}{rcl} \nabla(\iota) & = & \nabla_\iota \\ \nabla(\tau_1 \to \tau_2) & = & \tau_1 \to \nabla(\tau_1) \to \nabla(\tau_2) \end{array}$$

As suggested by this definition, each base type ι comes with its type of changes ∇_ι and two operations:

Comme le suggère cette définition, chaque type de base ι vient avec son type des changements ∇_ι et deux opérations :

$$\begin{array}{rcl} \oplus_\iota & : & \iota \to \nabla_\iota \to \iota \\ \ominus_\iota & : & \iota \to \iota \to \nabla_\iota \end{array}$$

We extend ∇ to typing evironments as follows:

On étend ∇ aux environnements de typage comme suit :

$$\begin{aligned}\nabla(\epsilon) &= \epsilon \\ \nabla(\Gamma; (x : \tau)) &= \nabla(\Gamma); dx : \nabla(\tau)\end{aligned}$$

For any type τ, we call *erased change structure*, denoted $\widetilde{\tau}$, the tuple $(\tau, \nabla(\tau), \oplus_\tau, \ominus_\tau)$.

Pour tout type τ, on appelle *structure de changements effacée* et on note $\widetilde{\tau}$ le quadruplet $(\tau, \nabla(\tau), \oplus_\tau, \ominus_\tau)$.

Question 11

In which sense does the simply-typed presentation fail to capture the invariants of structure changes?

En quoi le typage simple ne capture-t-il pas totalement le bon typage des calculs utilisant des structures de changements ?

□

4 Static differentiation in STLC / Différentiation statique dans STLC

We introduce the following syntax of terms:

On introduit maintenant la syntaxe des termes suivantes :

$$\begin{array}{rcll} t & ::= & x & \text{Variables} \\ & | & c & \text{Constants} \\ & | & t\,t & \text{Applications} \\ & | & \lambda x.t & \text{Abstractions} \end{array}$$

For a given term, we suppose that variables bound by a λ are all pairwise distinct.

Dans un terme donné, on supposera que tous les noms liés par un λ sont distincts deux-à-deux.

The function Derive(t) is a program transformation defined by induction over the structure of t whose purpose is to build an efficient implementation of the derivative:

La fonction Derive(t) est une transformation de programme définie par induction sur la structure de t et qui vise à construire une implémentation efficace de la dérivée :

$$\begin{aligned}\text{Derive}(x) &= dx \\ \text{Derive}(\lambda x.t) &= \lambda x\, dx.\text{Derive}(t) \\ \text{Derive}(t\,u) &= \text{Derive}(t)\, u\, (\text{Derive}(u)) \\ \text{Derive}(c) &= \delta_c\end{aligned}$$

As suggested by this definition, we suppose that each constant $c : \tau$ is associated with a derivative $\delta_c : \nabla(\tau)$. It is actually the derivatives of these constants that drives incrementalization: Derive(t) is designed to transport this incrementalization across the constructors of the λ-calculus in a compositional calculus.

Comme le suggère cette définition, on suppose que chaque constante $c : \tau$ est associée à une dérivée $\delta_c : \nabla(\tau)$. Ce sont les dérivées des constantes qui accomplissent le travail d'incrémentalisation du calcul : Derive(t) est conçue pour transporter cette incrémentalisation à travers les constructions du λ-calcul de façon compositionnelle.

Question 12

Apply the transformation to the following term:

Appliquez la transformation sur le terme suivant :

$$\lambda f\, x.f\, x$$

□

Question 13

Let f be the following function:

Soit la fonction f suivante :

$$f \stackrel{def}{=} (\lambda f x.f\, x)\,\mathsf{sum}$$

Explain why Derive(f) is more efficient (in terms of additions performed) than the naive derivative f' of f defined as follows:

Expliquez pourquoi Derive(f) est plus efficace (en nombre d'addition) que la dérivée naïve f' de f définie comme suit :

$$f' \stackrel{def}{=} \lambda x\, dx.f\,(x \oplus dx) \ominus f\, x$$

□

Question 14

Let $\Gamma \vdash t : \tau$. Formalise and prove the fact that Derive(t) is well-typed.

Soit $\Gamma \vdash t : \tau$. Formalisez et prouvez la propriété de bon typage de Derive(t).

□

5 Differential semantics of STLC / Sémantique différentielle de STLC

As illustrated in Question 11, there exists simply-typed terms that perform well-typed changes, at the expense of being partially defined. We must therefore strengthen the hypothesis implying the correctness of the transformation Derive(•).

The correctness statement is based on a denotational semantics of the simply-typed λ-calculus called differential semantics. This semantics will help us characterize the terms of the λ-calculus for which the derivative obtained by Derive(•) is defined.

We write $[\![\bullet]\!]$ the standard interpretation of λ-terms: as seen in class, it consists in a function associating to each term of the λ-calculus its interpretation as a term in type theory. We also consider a differential interpretation function of simply-typed λ-terms into type theory, denoted $d[\![\bullet]\!]$. This differential interpretation is parameterized by an environment ρ for identifiers of values and an environment $d\rho$ for identifiers of changes. The standard interpretation of λ-terms is only parameterized by an environment ρ. The syntax of environments is the following:

Comme l'illustre la Question 11, il existe des termes effectuant des changements bien typés avec des types simples mais partiellement définis. Il faut donc renforcer les hypothèses impliquant la correction de la transformation Derive(•).

L'énoncé de correction qui va nous intéresser s'appuie sur une sémantique dénotationnelle du $\lambda-$calcul simplement typé appelée sémantique différentielle. Cette sémantique va aider à caractériser les termes du λ-calcul pour lesquels la dérivée obtenue par Derive(•) est définie.

On note $[\![\bullet]\!]$ pour l'interprétation standard des λ-termes : comme en cours, il s'agit d'une fonction qui associe à chaque terme du λ-calcul une interprétation sous la forme d'un terme exprimé en théorie des types. On se dote aussi d'une fonction d'interprétation différentielle des termes du λ-calcul dans la théorie des types notée $d[\![\bullet]\!]$. Cette interprétation différentielle est paramétrée par un environnement ρ pour les identificateurs de valeurs et par un environnement $d\rho$ pour les identificateurs de changements. L'interprétation standard des λ-termes est seulement paramétrée par un environnement ρ. Voici la syntaxe de ces environnements d'évaluation :

$$
\begin{array}{rcll}
\rho & ::= & \epsilon & \text{Empty value environment} \\
 & | & \rho; (x = v) & \text{Extension of } \rho \text{ with the denotation of } x \\
d\rho & ::= & \epsilon & \text{Empty change environment} \\
 & | & d\rho; (dx = dv) & \text{Extension of } d\rho \text{ with the denotation of } dx
\end{array}
$$

For each base type ι, we are given a set I_ι whose elements are the denotations of the inhabitants of ι. We also assume that this set is equipped with a change structure $\widehat{I_\iota}$ such that $\forall v \in I_\iota, \Delta_{\widehat{I_\iota}} v \subseteq [\![\nabla_\iota]\!]$.
Finally, for every constant c, we are given a standard interpretation C. Note that since a constant has no free variable, the differential interpretation of a constant c is $\mathbf{0}_C$. This differential interpretation must coincide with the standard interpretation of δc, *i.e.* we have $\mathbf{0}_C = [\![\delta_c]\!]$.

Pour chaque type de base ι, on suppose donné un ensemble I_ι, dont les éléments sont les dénotations des habitants de ι. On suppose aussi que cet ensemble est muni d'une structure de changements $\widehat{I_\iota}$ tel que $\forall v \in I_\iota, \Delta_{\widehat{I_\iota}} v \subseteq [\![\nabla_\iota]\!]$. Enfin, pour chaque constante c, on suppose donnée une interprétation standard C. Notez que comme une constante n'a pas de variables libres, l'interprétation différentielle d'une constante c est $\mathbf{0}_C$. Cette interprétation différentielle doit coïncider avec l'interprétation standard de δc i.e. on a $\mathbf{0}_C = [\![\delta_c]\!]$.

The differential interpretation function is defined by induction over the structure of terms, as follows:

La fonction d'interprétation différentielle est définie par induction sur la structure des termes comme suit :

$$
\begin{array}{rcl}
d[\![x]\!]\,\rho\,d\rho & = & d\rho(x) \\
d[\![\lambda x.t]\!]\,\rho\,d\rho & = & \lambda v\,dv.d[\![t]\!]\,(\rho; x = v)\,(d\rho; dx = dv) \\
d[\![t\,u]\!]\,\rho\,d\rho & = & (d[\![t]\!]\,\rho\,d\rho)\,([\![u]\!]\,\rho)\,(d[\![u]\!]\,\rho\,d\rho) \\
d[\![c]\!]\,\rho\,d\rho & = & \mathbf{0}_C
\end{array}
$$

Question 15

Recall the definition of $[\![\bullet]\!]$ for simple types, typing environments and terms of the simply-typed λ-calculus.

Rappelez la définition de $[\![\bullet]\!]$ pour les types simples, les environnements de typage et les termes du λ-calcul simplement typé.

□

The correctness of the differential semantics is stated as follows:

Le lemme de correction de la sémantique différentielle s'énonce comme suit :

Lemma 1 *If $\Gamma \vdash t : \tau$, then $d[\![t]\!]$ is a derivative of $[\![t]\!]$.*

Question 16

To give meaning to Lemma 1, we must introduce a change structure over environments. Let $\rho \in [\![\Gamma]\!]$. Define $d\rho \in d[\![\Delta_\Gamma(\rho)]\!]$ as well as the associated operations $\oplus$ and $\ominus$.

Pour donner du sens au lemme 1, il faut introduire une structure de changements sur les environnements d'évaluation. Soit $\rho \in [\![\Gamma]\!]$. Définissez $d\rho \in d[\![\Delta_\Gamma(\rho)]\!]$ ainsi que les opérations $\oplus$ et $\ominus$ associées.

□

Question 17

Give the proof of Lemma 1.

Donnez la preuve du lemme 1.

□

6 Soundness of the transformation / Correction de la transformation

To prove the correctness of the transformation Derive(•), we must show that $[\![\text{Derive}(t)]\!]$ and $d[\![t]\!]$ coincide.

Pour montrer la correction de la transformation Derive(•), on va montrer que $[\![\text{Derive}(t)]\!]$ et $d[\![t]\!]$ coïncident.

Assume that $\Gamma \vdash t : \tau$. Note that for any $\rho \in [\![\Gamma]\!]$ and $d\rho' \in [\![\nabla(\Gamma)]\!]$, $[\![\text{Derive}(t)]\!]\,(\rho; d\rho') \in [\![\nabla(\tau)]\!]$ whereas for any $d\rho \in d[\![\Delta_\Gamma(\rho)]\!]$, $d[\![t]\!]\,\rho\,d\rho \in d[\![\Delta_\tau(v)]\!]$ given a certain value v resulting from the evaluation of $[\![t]\!]\,\rho$.

Supposons $\Gamma \vdash t : \tau$. Notez que pour tout $\rho \in [\![\Gamma]\!]$ et $d\rho' \in [\![\nabla(\Gamma)]\!]$, $[\![\text{Derive}(t)]\!]\,(\rho; d\rho') \in [\![\nabla(\tau)]\!]$ tandis que pour tout $d\rho \in d[\![\Delta_\Gamma(\rho)]\!]$, $d[\![t]\!]\,\rho\,d\rho \in d[\![\Delta_\tau(v)]\!]$ pour une certaine valeur v résultante de l'évaluation de $[\![t]\!]\,\rho$.

As mentioned earlier, there exists inhabitants of $[\![\nabla(\tau)]\!]$ which are not valid changes for $d[\![\Delta_\tau(v)]\!]$. We wish to restrict our attention to those inhabitants of $[\![\nabla(\tau)]\!]$ that are the *erasure* of inhabitants of $d[\![\Delta_\tau(v)]\!]$. To this end, we define the following logical relation:

Comme dit plus tôt, il existe des habitants de $[\![\nabla(\tau)]\!]$ qui ne sont pas des changements valides pour $d[\![\Delta_\tau(v)]\!]$. On souhaite donc se restreindre aux habitants de $[\![\nabla(\tau)]\!]$ qui sont des *effacements* des habitants de $d[\![\Delta_\tau(v)]\!]$. Pour cela, on se donne la relation logique suivante :

Definition 1 *Let $dv \in d[\![\Delta_\tau(v)]\!]$ and $dv' \in [\![\nabla(\tau)]\!]$. We say that dv' is the erasure of dv, and we write $dv \triangleright_\tau^v dv'$ if one of these two conditions is satisfied:*

- *τ is a base type and $dv = dv'$.*
- *$\tau = \tau_1 \to \tau_2$ and for all w, dw and dw' such that $dw \triangleright_{\tau_1}^w dw'$, we have $dv\,w\,dw \triangleright_{\tau_2}^{v\,w} dv'\,w\,dw'$.*

Question 18

Prove the following property:

Montrer la propriété suivante :

Lemma 2 *If $dv \triangleright_\tau^v dv'$ then $v \oplus dv = v \oplus dv'$*

□

We naturally extend the erasure relation to environments.

On étend naturellement la relation d'effacement aux environnements d'évaluation.

Question 19

Prove the following property:

Montrer la propriété suivante :

Lemma 3 *If $\Gamma \vdash t : \tau$ and $d\rho \triangleright^\rho d\rho'$, then $d[\![t]\!]\,\rho\,d\rho \triangleright_\tau^{[\![t]\!]\,\rho} [\![Derive(t)]\!]\,(\rho; d\rho')$.*

□

Question 20

Prove the following correctness theorem:

Montrer le théorème de correction suivant :

Theorem 1 *Let $f : \tau_1 \to \tau_2$ be a closed term. For all closed term t of type τ_1, and for all closed change term dt' of type $\nabla(\tau_1)$ such that $dt \triangleright^t_{\tau_1} dt'$ for some $dt : \Delta_{\tau_1}(t)$, then $[\![f\,(t \oplus dt)]\!] = [\![f\,t \oplus \mathit{Derive}(f)\,t\,dt']\!]$.*

□

Question 21

What are the limits of this correctness proof? In particular, is it well-suited for a programming language featuring general recursion? Suggest another proof technique which would allow us to prove this result in a partial setting.

Quelles sont les limites de cette preuve de correction ? En particulier, est-elle adaptée à un langage de programmation muni de la récursion arbitraire ? Proposez une autre technique de preuve de correction qui s'accomoderait de la potentielle non terminaison des calculs.

□

Whereas the term produced by Derive(•) is efficient, it is not entirely satisfactory in its handling of applications. Indeed, if we assume that we have already computed "$t\,u$", the computation performed by Derive$(t\,u)$ (which is equal to "Derive$(t)\,u\,($Derive$(u))$") recomputed u once again.

Bien que la transformation Derive(•) soit efficace, elle n'est pas totalement satisfaisante dans son traitement des applications. En effet, si on suppose avoir déjà calculé "$t\,u$", le calcul effectué par Derive$(t\,u)$ (qui vaut "Derive$(t)\,u\,($Derive$(u))$") recalcule de nouveau l'argument u.

Question 22

Suggest an approach to address this inefficiency of the derivative.

Proposez une solution à cette inefficacité de la dérivée.

□