Théorie de la Programmation, L3IF – TD

mercredi 12 décembre 2018, 8h-10h

1 Des polynômes de types pour FUN

1.1 Produits

On étend le langage FUN avec ce qu'on appelle les *types produits*, qui accompagnent la capacité de former des couples. Il y a trois nouvelles constructions dans le langage : les parenthèses avec la virgule pour faire un couple, et deux projections pour "désassembler" un couple. Les notations sont les suivantes : (e_1, e_2), $\texttt{fst}(e)$, $\texttt{snd}(e)$.

Du côté du typage, le type noté $\tau_1 * \tau_2$ est associé à un couple constitué d'une expression de type τ_1 et d'une expression de type τ_2.

Proposez des règles d'inférence supplémentaires pour étendre la définition de

1. la sémantique à grands pas,
2. la sémantique à petits pas,
3. le typage.
4. Quel type est équivalent à $(\tau_1 * \tau_2) \to \tau_3$?

   équivalent au sens où il y a deux opérations inverses l'une de l'autre entre les deux types

1.2 Sommes

La "contrepartie" des types produit sont les types somme, qui correspondent à la disjonction entre deux types : il y a deux manières de construire une expression appartenant à un type somme, l'injection gauche, notée `il`, et l'injection droite, notée `ir`. Pour "désassembler" un tel objet, on utilise une construction `case`, qui raisonne par cas suivant l'injection qui est utilisée.

On utilise naturellement la notation $\tau_1 + \tau_2$ pour un type somme.

Proposez des règles d'inférence supplémentaires pour étendre la définition de

1. la sémantique à grands pas,
2. la sémantique à petits pas,
3. le typage.

1.3 Tout

On suppose que l'on part de la grammaire suivante pour les types de "FUN étendu avec sommes et produits" :

$$\tau \quad ::= \quad \texttt{unit} \mid \tau_1 \to \tau_2 \mid \tau_1 * \tau_2 \mid \tau_1 + \tau_2 \ ,$$

où `unit` est un type avec un seul élément, noté () : la grammaire de FUN est donc modifiée ainsi :

$$e \quad ::= \quad () \mid \texttt{fun}\ x\ \texttt{->}\ e \mid x \mid e_1\ e_2 \ .$$

1. Comment définir les types `bool` ? et τ `option` ?
2. Les types pour les entiers naturels et les listes d'entiers sont solutions de quelles équations ? (On vous demande juste de deviner ces équations, pas de justifier comment on résoud ces équations)

2 Sémantique contextuelle pour FUN

1. Inspirez-vous du DM 1bis (cf. la page www du cours) pour définir une sémantique contextuelle pour FUN. Rappelons que cela passe par la définition de la relation $\hookrightarrow$ et des contextes d'évaluation.

2. Traitez l'extension avec les couples.

3 Exceptions

$$e \quad ::= \quad \cdots \mid \texttt{raise E(e)} \mid \texttt{try}\ e_1\ \texttt{with E(x)}\ \rightsquigarrow\ \texttt{e}_2$$

Une exception, notée E(e), où e doit s'évaluer en un entier. À noter que E(e) n'est pas une expression (une fonction ne peut pas prendre une exception en argument, par exemple).

Dans try e_1 with E(x) $\rightsquigarrow$ $\texttt{e}_2$, x est lié dans e_2.

1. Sémantique opérationnelle, première solution : grands et petits pas "comme d'habitude".

2. puis avec la sémantique contextuelle

3. typage