Optimisations de la traduction en CPS

Dans ce document, on décrit les optimisations réalisées dans fouine, notamment au niveau de la traduction d'un code fouine en code fouine CPS-ifié.

Dans fouine, nous avons implémenté deux optimisations :
1. précalcul lorsque cela est possible ;
2. optimisation concernant l'application de fonctions.

I. Précalcul.

On définit une relation $\rightsquigarrow$ entre une expression fouine et un résultat précalculé. C'est une fonction partielle.

Dans le cas de la somme d'entiers, on a :
- $n_1 \oplus n_2 \rightsquigarrow n$ où $n = n_1 + n_2$.

On peut faire de même avec les autres opérations arithmétiques simples ($\ominus$, $\otimes$). Pour la division d'entiers $\oslash$ (et le modulo), on s'assure que n_2 n'est pas toujours nul :
- $n_1 \oslash n_2 \rightsquigarrow n$ où $n_2 \neq 0$ et $n = n_1/n_2$.

On fait de même pour les opérations unaires.

Lorsqu'on a une fonction, c'est une valeur et on s'arrête donc de précalculer. Il faut cependant se rappeler de traduire par continuation la suite de e, noté $[\![e]\!]$:
- `fun` x `->` $e \rightsquigarrow$ `fun` x `->` $[\![e]\!]$.

Lorsqu'on a une variable x, un entier, un flottant, une chaîne de caractères, un caractère, un *unit* `()`, ou un booléen, ces objets se réduisent en eux-mêmes :
- $x \rightsquigarrow x$.

Lorsqu'on a une application, on ne réduit plus :
- $e_1\ e_2 \not\rightsquigarrow \cdots$.

En effet, on pourrait engendrer des *effets de bord* : par exemple, le code `prInt 3` est sous forme de deux expressions précalculées mais

pré-effectuer le calcul engendrerait une absence des effets de bords lors de l'exécution.

Si l'on a un type algébrique $\mathrm{C}(e_1, \ldots, e_n)$, alors on a la forme précalculé en précalculant chacun des éléments :
- $\mathrm{C}(e_1, \ldots, e_n) \rightsquigarrow \mathrm{C}(e'_1, \ldots, e'_n)$ si, pour tout $1 \leq i \leq n$, $e_i \rightsquigarrow e'_i$.

Il suffit qu'un des e_i ne soit pas pré-calculable pour que l'on ne puisse pas calculer le type algébrique.

Lorsqu'on a une séquence, on a :
- e_1 ; $e_2 \rightsquigarrow e'_2$ si $e_1 \rightsquigarrow e'_1$ et $e_2 \rightsquigarrow e'_2$.

Comme c'est une séquence, on s'en fiche du résultat de e'_1, on sait juste qu'il n'est pas utile. En passant, s'il ne génère pas d'effets de bords, pourquoi avait-on une séquence alors ? C'est exactement équivalent d'écrire e_2.

Lorsqu'on a un `if`, et qu'on précalcule que la condition est `true` ou `false`, alors on élimine le `if` (peu importe si e_1 et e_2 sont réductibles) et on tente de réduire la branche correspondante.

Dans les autres cas, on ne se réduit pas.

Avant de commencer à traduire, on parcours l'arbre de syntaxe nœud par nœud en commençant par les feuilles et en tentant de précalculer toutes les expressions possibles. Ceci a pour but de simplifier l'expression finale

```
1 + 2 * (if 4 > 2 then 4 * 3 * 2 * 1 else 34 mod 17)
```

en

```
49
```

directement.

II. Application.

Comme expliqué dans le document `CPS-transformation.pdf`, on décrit la transformation classique de l'application :
- $[\![e_1\ e_2]\!] := \mathtt{fun}\ k \to [\![e_2]\!]\ (\mathtt{fun}\ v \to [\![e_1]\!]\ (\mathtt{fun}\ f \to f\ v\ k))$.

Dans ce document, on ne s'intéresse pas aux variables libres, ni à la 2nde continuation (la continuation *boom* ne demande que d'ajouter $\mathit{snd}\ k$ à dans les définitions de continuations).

Lorsqu'on peut précalculer (au sens de la section précédente) un terme, e_1 ou e_2, il n'est pas nécessaire de le traduire, puis de l'évaluer, et d'attendre que la continuation nous rappelle. Il suffit d'intercaler le résultat précalculé au moment de la traduction.

On a donc la disjonction de cas suivante :

- si $e_1 \rightsquigarrow e_1'$ et $e_2 \rightsquigarrow e_2'$ alors
$$[\![e_1\ e_2]\!]_{\text{opt}} := \texttt{fun}\ k \to e_1'\ e_2'\ k\ ;$$
- si $e_1 \rightsquigarrow e_1'$ et $e_2 \not\rightsquigarrow \cdots$ alors
$$[\![e_1\ e_2]\!]_{\text{opt}} := \texttt{fun}\ k \to [\![e_2]\!]_{\text{opt}}\ (\texttt{fun}\ v \to e_1'\ v\ k)\ ;$$
- si $e_1 \not\rightsquigarrow \cdots$ et $e_2 \rightsquigarrow e_2'$ alors
$$[\![e_1\ e_2]\!]_{\text{opt}} := \texttt{fun}\ k \to [\![e_1]\!]_{\text{opt}}\ (\texttt{fun}\ f \to f\ e_2'\ k)\ ;$$
- si $e_1 \not\rightsquigarrow \cdots$ et $e_2 \not\rightsquigarrow \cdots$ alors
$$[\![e_1\ e_2]\!]_{\text{opt}} := \texttt{fun}\ k \to [\![e_2]\!]_{\text{opt}}\ \Big(\texttt{fun}\ v \to [\![e_1]\!]_{\text{opt}}\ (\texttt{fun}\ f \to f\ v\ k)\Big).$$