BROUILLON - DES ARBRES DONT LES FEUILLES SONT... DES ARBRES !

CHRISTOPHE BAL

Table des matières

1. Notre type de base

Définition 1. *Le type $arbre(\alpha)$ est défini comme suit où α est un type.*

$$\dfrac{a : \alpha}{Leaf(a) : arbre(\alpha)}\ \textit{(feuille)} \qquad \dfrac{t_1 : arbre(\alpha) \qquad t_2 : arbre(\alpha)}{Node(t_1, t_2) : arbre(\alpha)}\ \textit{(noeud)}$$

Exemple 1. *A l’aide des règles précédentes, on peut définir l’arbre suivant de type $arbre(\mathbb{N})$ où nous utilisons les abréviations suivantes $N = Node$, $L = Leaf$ et $AN = arbre(\mathbb{N})$.*

$$\dfrac{\dfrac{\dfrac{1 : \mathbb{N}}{l_1 = L(1) : AN}\ \textit{(feuille)} \quad \dfrac{\dfrac{2 : \mathbb{N}}{l_2 = L(2) : AN}\ \textit{(feuille)} \quad \dfrac{3 : \mathbb{N}}{l_3 = L(3) : AN}\ \textit{(feuille)}}{n_{23} = N(l_2, l_3) : AN}\ \textit{(noeud)}}{n_{123} = N(l_1, n_{23}) : AN}\ \textit{(noeud)} \qquad \dfrac{4 : \mathbb{N}}{l_4 = L(4) : AN}\ \textit{(feuille)}}{N(n_{123}, l_4) : AN}\ \textit{(noeud)}$$

Dans la suite, on utilisera aussi une écriture fonctionnelle pour représenter un arbre. Pour notre exemple, ceci donne :

```
Node(
    Node(
        Leaf(1),
        Node(
            Leaf(2),
            Leaf(3)
        )
    ),
```

Date: 21 Mars 2019 - 10 Avril 2019.

```
    Leaf(4)
)
```

Avec cette représentation, le plus « simple » *des arbres de type* $arbre(\mathbb{N})$ *est le suivant avec* $k \in \mathbb{N}$ *quelconque.*

```
Leaf(k)
```

2. Un type dérivé

Définition 2. *Le type* $arbre(arbre(\alpha))$ *est défini comme suit où* α *est un type* (nous appliquons juste au type $arbre(\alpha)$ la définition 1).

$$\frac{a : arbre(\alpha)}{Leaf(a) : arbre(arbre(\alpha))}\ \textit{(feuille)} \qquad \frac{t_1 : arbre(arbre(\alpha)) \quad t_2 : arbre(arbre(\alpha))}{Node(t_1, t_2) : arbre(arbre(\alpha))}\ \textit{(noeud)}$$

Exemple 2. *Dans cet exemple, nous noterons* `Leaf''` *et* `Node''` *les applications feuille et noeud pour le type* $arbre(arbre(\mathbb{N}))$*. Le plus* « simple » *des arbres de type* $arbre(arbre(\mathbb{N}))$ *est le suivant avec* $k \in \mathbb{N}$ *quelconque.*

```
Leaf''(Leaf(k))
```

On peut alors fabriquer un arbre similaire, mais pas identique, au 1*er* *arbre de l'exemple 1.*

```
Node''(
    Node''(
        Leaf''(Leaf(1)),
        Node''(
            Leaf''(Leaf(2)),
            Leaf''(Leaf(3))
        )
    ),
    Leaf''(Leaf(4))
)
```

3. Une comparaison sans formalité de nos deux types

Il est facile de passer des arbres de l'exemple 1 à ceux de l'exemple 2 : chaque `Leaf(k)` doit être remplacé par `Leaf(Leaf(k))` pour $k \in \mathbb{N}$.

De façon similaire, on peut passer des arbres de l'exemple 2 à ceux de l'exemple 1.

Mais n'allons pas trop vite... En effet, pour tout arbre `t` de type $arbre(\mathbb{N})$, nous avons `Leaf(t)` qui est de type $arbre(arbre(\mathbb{N}))$. Ceci a des implications fortes. Pour le voir, considérons l'arbre `TEX` suivant qui est de type $arbre(\mathbb{N})$.

```
Node(
    Leaf(1),
    Node(
        Node(
```

```
            Leaf(2),
            Leaf(3)
        ),
        Leaf(4)
    )
)
```

Voici un premier arbre de type $arbre(arbre(\mathbb{N}))$ pouvant être *« interprété naturellement »* comme étant l'arbre `TEX`. De nouveau, nous noterons `Leaf''` et `Node''` les applications feuille et noeud pour le type $arbre(arbre(\mathbb{N}))$.

```
Leaf''(
    Node(
        Leaf(1),
        Node(
            Node(
                Leaf(2),
                Leaf(3)
            ),
            Leaf(4)
        )
    )
)
```

Voici un autre exemple.

```
Node''(
    Leaf''(Leaf(1)),
    Node''(
        Node''(
            Leaf''(Leaf(2)),
            Leaf''(Leaf(3))
        ),
        Leaf''(Leaf(4))
    )
)
```

On peut aussi proposer l'arbre ci-dessous.

```
Node''(
    Leaf''(Leaf(1)),
    Node''(
        Leaf''(
            Node(
                Leaf(2),
                Leaf(3)
            )
        ),
```

```
        Leaf''(Leaf(4))
    )
)
```

Pour finir, voici une dernière proposition.

```
Node''(
    Leaf''(Leaf(1)),
    Leaf''(
        Node(
            Node(
                Leaf(2),
                Leaf(3)
            ),
            Leaf(4)
        )
    )
)
```

Si l'on regarde bien chacun des exemples proposés, nous notons que les `Leaf''` *« polluent »* les arbres mais pas les `Node''`. En fait, en transformant syntaxiquement `Node''(...)` en `Node(...)` , puis `Leaf''(Leaf(...))` en `Leaf(...)` et enfin `Leaf''(Node(...))` en `Node(...)` alors pour chaque exemple nous retombons sur notre arbre initial `TEX` de type $arbre(\mathbb{N})$.

Intuitivement le type $arbre(arbre(\mathbb{N}))$ ne contient pas plus d'information que le type $arbre(\mathbb{N})$. L'objectif de la section suivante va être de définir rigoureusement ce que nous entendons par *« information »* puis ensuite de voir qu'effectivement le type $arbre(arbre(\mathbb{N}))$ ne donne pas plus d'information que le type $arbre(\mathbb{N})$.

4. Comparer (?) des types

Sans donner de signification aux arbres que nous présentons, les arbres de type $arbre(arbre(\mathbb{N}))$ et ceux de type $arbre(\mathbb{N})$ sont de natures structurelles totalement différentes comme le montrent les exemples précédents.

D'un autre côté si nous nous intéressons non aux structures de nos arbres mais à ce qu'ils stockent, alors la section précédente nous pousse à trouver une grande similitude entre les arbres de type $arbre(arbre(\mathbb{N}))$ et ceux de type $arbre(\mathbb{N})$.

Autrement dit, la comparaison des types $arbre(arbre(\mathbb{N}))$ et $arbre(\mathbb{N})$ ne peut faire sens que du point de vie sémantique, et non simplement via une analyse syntaxique. Nous devons donc arriver à donner une signification à nos arbres. C'est ce que proposent les deux définitions *« naturelles »* suivantes.

Définition 3. *KKKKK*

Définition 4. *KKKKK*

Exemple 3. *? ? ? ? type* $arbre(\mathbb{N})$

```
???
```

Ici nous avons ? ? ? ?

Exemple 4. *? ? ? ? type* $arbre(arbre(\mathbb{N}))$

```
???
```

Ici nous avons ? ? ? ?

de n souhaite définir l'ingformation contenue
on définit une fonciton $info\text{-}arbre$ stockant profondeur usivi de la valeur d'un noued
l'info définit l'arbre de façon unique !
on montre que les type ont le smême info si on définit une fino récursivement sue arbre(arbre(N))
on crée une ou deux fnction qui fabrique une liste qui stocke -2 ou 2 si on va à gaiche ou à droite dans un arbre, et si c'est une feuille on stocke valeur 10^n
de la sorte on repère valeur et déplacement
pb pour arbre(arbre(N)), comment gfircer vélautaion de la feuille ? ? ?

PLUS SIMPLE !

$info\text{-}arbre$ et $info\text{-}arbre\text{-}arbre$ qui décrive le mêm enselble à carcatériser proprment ! ! ! !