TP n°4

Types algébriques

Les declarations de types permettent de representer et manipuler facilement des arbres de toutes natures en OCaml. Dans ce TP, on s'interesse a un modele arborescent simple de documents structures.

1 Blocs de texte

Pour commencer, on de nit un *bloc de texte* comme etant :
{ soit une cha^ne de caractere,
{ soit un bloc de texte a uble d'un *style* (gras, italique ou souligne),
{ soit la *concaténation* de plusieurs (une liste de) blocs de texte.

Exercice 1. Declarer un type `style` pour representer les trois di erents styles possibles.

Exercice 2. Declarer un type `text` pour representer un bloc de texte. Il contiendra par exemple la valeur

```
Concat [Text "Bonjour, ";
        Style (Bold, Concat [Text "je suis gras ";
                             Style (Italics, Text "et italique.")])]
```

qui denote le bloc de texte « Bonjour, je suis gras ***et italique.*** ». Comment pourrait-on representer un bloc de texte vide ?

Astuce. L'expression `function ...` est un raccourci syntaxique pour `fun x → match x with ...`. On peut donc ecrire `let f = function ...` au lieu de `let f x = match x with ...`.

Exercice 3. Ecrire une fonction `paren` de type `text → text` qui met entre parentheses tout les blocs de texte en italique. Pensez a la fonction `List.map` !

Exercice 4. Ecrire une fonction `plain` de type `text → text` qui supprime tout le formattage d'un bloc de texte.

Exercice 5. Ecrire une fonction `nobold : text → text` qui enleve tous les blocs en gras.

Exercice 6. Modi ez la fonction precedente pour qu'elle enleve tous le texte (seulement) en gras, mais conserve le texte en gras *et* italique (elle renverra «Bonjour, ***et italique***» sur l'exemple precedent).

Exercice 7. Ecrire une fonction `italics` de type `text → text list` qui renvoie la liste des blocs en italique. Vous pourrez vous servir de la fonction `List.fold_left` : $(\alpha \to \beta \to \alpha) \to \alpha \to \beta$ `list` $\to \alpha$ qui \itere" l'application d'une fonction f sur une liste :

$$\texttt{List.fold\_left}\ f\ a\ [b_1; \ldots; b_n] = f\ (\ldots\ (f\ (f\ a\ b_1)\ b_2)\ \ldots) b_n$$

Attention aux italiques imbriques !

Exercice 8. Ecrire une fonction `to_html` de type `text → string` qui convertit un bloc de texte en sa representation HTML. On rappelle que les balises `<b>`, `<i>`, `<u>` mettent respectivement leur contenu en gras, italique et souligne.

Exercice 9. Cette representation n'est pas *normalisée*, dans le sens ou a plusieurs valeurs de types `text` correspondent le même a chage (la même cha^ne HTML). Par exemple,

```
Concat [Text "Bon"; Text "jour, ";
        Style (Bold, Concat [
          Text "je suis gras, "; Text "";
          Style (Italics, Concat [Text "et italique."; Concat []])])
```

a la même representation HTML que l'exemple plus haut.

1. Ecrire une fonction d'aide `map : (text list → text list) → text → text` telle que `map` f t applique f a chaque liste dans les `Concat` de t.
2. Ecrire une fonction `remove : text list → text list` qui enleve de la liste de blocs les blocs `Text` et `Concat` vides.
3. Ecrire une fonction `flatten : text list → text list` qui aplatit les `Concat` imbriques.
4. Ecrire une fonction `concat : text list → text list` qui concatene les cha^nes de caracteres `Text s` adjacentes dans la liste d'entree.
5. Ecrire une fonction `normalize : text → text` qui normalise la representation : elle iterera les fonctions precedentes jusqu'a arriver a un point xe. Attention a l'ordre dans lequel vous appliquez les transformations !

Exercice 10 ($\star$). Sauriez-vous refaire l'exercice precedent en une seule passe ?

2 Documents

On de nit un *document* comme etant :
- soit un paragraphe, c'est-a-dire un bloc de texte,
- soit une section, avec un titre (un bloc de texte) et contenant une liste de documents.

Une sous-section est donc une section a l'interieur d'une autre section.

Exercice 11. De nir un type `document` pour les representer.

Exercice 12. De nir une fonction `map_text` telle que `map_text` f d applique f sur tous les blocs de texte de d. De nir une fonction `plain : document → document` qui enleve le formattage de tout un document.

Exercice 13. De nir une fonction `to_html` qui convertit un document en sa representation HTML a l'aide des balises `<h1>`, `<h2>` etc.

Exercice 14. On de nit le type des tables des matieres `type toc = Toc of text * toc list`. Ecrire une fonction `toc : document → toc` qui calcule la table des matieres d'un document. Ecrire une fonction `to_html` qui la convertit en HTML a l'aide des balises de listes `<ul>` et `<li>`.

Exercice 15 ($\star$). De nir une fonction `numerote : document → document` qui rajoute en tête du titre de chaque section une numerotation comme «1.», «1.2.», «1.3.2.»...