INFO0947: Récursivité et Elimination de la Récursivité

Peissone Dumoulin, s193957

1 Formulation Récursive

1.1 Cas de base

Si $n = 1$:

$$hexa\_dec\_rec(hexa, n) = convert(hexa[n-1])$$

1.2 Cas récursif

Si $n > 0$:

$$hexa\_dec\_rec(hexa, n) = convert(hexa[n-1]) + 16 * hexa\_dec\_rec(hexa, n-1)$$

2 Spécification

```
1 //PréConditions : hexa != NULL, n >= 0
2 //PostConditions : hexa_dec_rec = decimal ∧ hexa = hexa0 ∧ n = n0
3 unsigned int hexa_dec_rec(char *hexa, int n);
```

3 Construction Récursive

3.1 Programmation Défensive

On vérifie que la précondition est respectée en interdisant à hexa d'être NULL et n ne peut pas être strictement négatif

```
1 unsigned int hexa_dec_rec(char *hexa, int n){
2     assert(hexa != (void*)0 && n >= 0);
3     // {PréConditions ≡ hexa ≠ NULL ∧(length(hexa) ≥ 0 ⟹ n ≥ 0)}
4 }
```

3.2 Cas de Base

On gère le cas de base où $n = 1$ après s'être assuré que les préconditions sont bien respectées.

```
// {PréConditions ≡ hexa ≠ NULL ∧(length(hexa) ≥ 0 ⟹ n ≥ 0)}
if(n == 1)
    // {hexa ≠ NULL ∧ n = 1}
    return convert(hexa[n - 1]);
    // {hexa_dec_rec = convert(hexa[n - 1]) ∧ hexa = hexa0 ∧ n = n0}
    // { ⟹ PostCondition}
}
```

3.3 Cas Récursifs

Il y a un seul cas récursif, lorsque n $\neq$ 1. {PréConditions$_{REC}$} et {PostConditions$_{REC}$} sont respectivement les PréConditions et les PostConditions de l'appel récursif.

```
else
    // {hexa ≠ NULL ∧ n ≠ 1}
    // { ⟹ PréConditionsREC}
    return convert(hexa[n - 1]) + 16 * hexa_dec_rec(hexa, n - 1);
    // {PostConditionsREC ≡
    // hexa_dec_rec(hexa, n) = convert(hexa[n - 1]) + 16 * hexa_dec_rec(hexa, n-1)
    // ∧ n = n - 1 ∧ hexa = hexa0}
    // { ⟹ PostConditions}
```

3.4 Code complet

```
unsigned int hexa_dec_rec(char *hexa, int n){
  assert(hexa != (void*)0 && n >= 0);//Préconditions

  if(n == 1)
    return convert(hexa[n - 1]);//Cas de base
  else
    return convert(hexa[n - 1]) + 16 * hexa_dec_rec(hexa, n - 1);//Cas récursif
}
```

4 Traces d'Exécution

5 Complexité

6 Dérécursification