Modélisation d'un arbre en TaF

- Structure de données très classique... On doit pouvoir modéliser ça en TaF !
- Dans Taf (et pour le test), on se concentre sur des structures finies:
  - Générer un arbre de taille n et de hauteur h bornées.
- Problème : TaF ne permet pas les définitions récursives (ex: un arbre est composé de sous-arbres).
- La seule modélisation possible est de mettre tous les nœuds « à plat » sous une racine commune.
- Un nœud peut être en relation avec un autre, en stockant son id dans un attribut . Ex: un attribut father pour la relation « a pour père ».
- On doit modéliser les relations inter-noeuds par des contraintes, qui définissent ce qu'est un arbre.

Elements du modèle et contraintes sur leur nombre d'instances

Rq: le modèle dépend d'une unique constante MAX_INT, que je fixe ici arbitrairement à 50.

Root
size: 1..50
height: 0..50
Node[0..50]

Contraintes **TreeHeight1,2**:
height ≤ size -1
height = 0 <==> size = 1

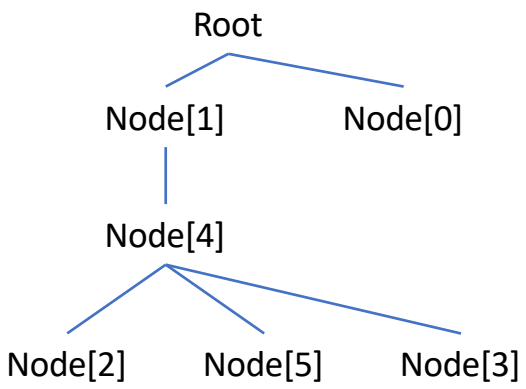
Contrainte **TreeSize**:
Node.nb_instances = size -1

Node
depth: 1..50
father: -1..50

Contrainte **TreeHeight3**:
depth ≤ ../height

Contrainte **FatherId**:
father < ../ Node.nb_instances

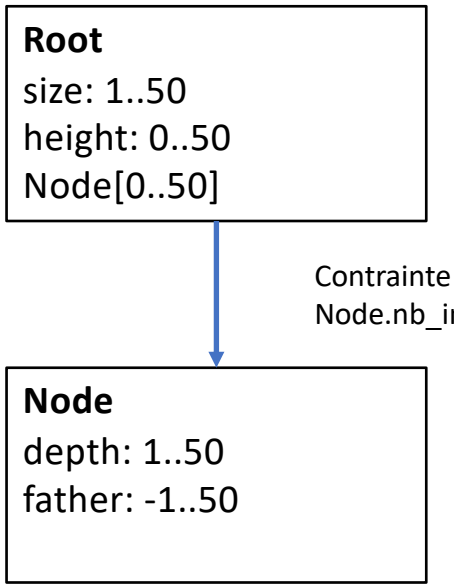
Exemple d'encodage d'arbre:



size = 7
height = 3
Node.nb_instances = 6
Node[0].depth = 1
Node[0].father = -1
Node[1].depth = 1
Node[1].father = -1
Node[2].depth = 3
Node[2].father = 4
Node[3].depth = 3
Node[3].father = 4
Node[4].depth = 2
Node[4].father = 1
Node[5].depth = 3
Node[5].father = 4

L'attribut father pointe sur un id de nœud.
La racine est encodée comme le père d'id -1.

Autres contraintes pour définir un arbre



Contraintes **TreeHeight1,2**:
height ≤ size -1
height = 0 <==> size = 1

Contrainte **TreeSize**:
Node.nb_instances = size -1

Contrainte **TreeHeight3**:
depth ≤ ../height

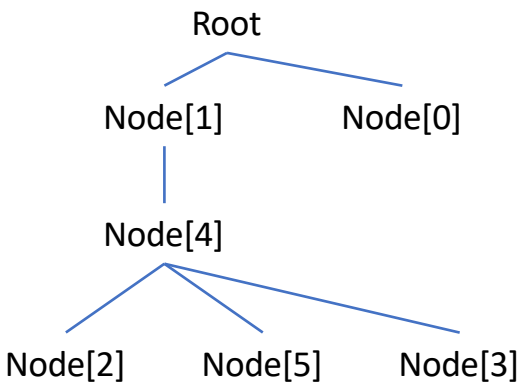
Contrainte **FatherId**:
father < ../ Node.nb_instances

Contrainte **TreeHeight4** (exprimée au niveau de Root):
height > 0 ==> (∃ n ∈ 0 .. Node.nb_instances - 1 :
Node[n].depth = height)

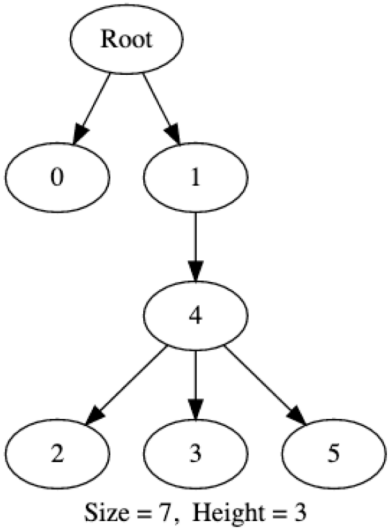
Contrainte **InductiveDefTree1** (exprimée au niveau de Node):
depth = 1 <==> father = -1

Contrainte **InductiveDefTree2** (exprimée au niveau de Node):
depth > 1 ==> (∃ n ∈ 0 .. Node.nb_instances - 1 :
n = father and ../Node[n].depth = depth -1)

Export vers GraphViz




```
digraph {
label = "Size = 7,  Height = 3";
Root;
Root -> 0;
Root -> 1;
4 -> 2;
4 -> 3;
1 -> 4;
4 -> 5;
}
```



- Export de Root:

```
print ("digraph { ")
print  ( "label = \"  Size = ", size,   ",Height =  ", height, ";")
print ("Root; ")
for i in range(0,Node.nb_instances)
    Node[i].Export(i)
```

- Export d'un nœud (l'appel lui passe son id i):

```
If father == -1
print  ("Root -> ", i, " ;")
else
print  (father, " -> ", i,   " ;")
```

Mesures de couverture: éléments à couvrir au moins une fois

- Plages de valeurs de size: [1, 16] [17, 33] [34, 50]
  low medium high
- Plages de valeurs de height: [1, 16] [17, 32] [33, 49]
  low medium high
- Pour les autres aspects plus sémantiques, on va s'intéresser à des plages de valeurs dans [0..1]. C'est-à dire qu'on va normaliser les indicateurs à couvrir.
- Custom1: regarder le choix de height dans 1 .. size -1. **N'est mesuré que pour size ≥ 4**, c.à.d. qu'il y a au moins 3 possibilités pour height. Normalisé (height – Min) / (Max – Min), avec Min = 1 et Max = size-1

  Height_vs_size = $\frac{height - 1}{size - 2}$ [0, 0.33[[0.33, 0.67]]0.67, 1.0]
  low medium high
- Custom 2, 3, 4: métriques spécifiques pour caractériser le degré selon lequel l'arbre généré est équilibré. Les métriques s'inspirent de différentes définitions de l'équilibrage : leaf-balanced, height-balanced, size balanced.

  On Complete and Size Balanced k-ary Tree Integer Sequences, S. Cha in *International Journal of Applied Mathematics and Informatics* Vol 6 issue 2, 2012, pp67-75

Leaf-balanced

- Propriété: toutes les feuilles sont de profondeur height ou height-1. Intéressant à mesurer lorsque **height > 2 et height < size – 1**, sinon l'arbre est trivialement équilibré. On en déduit aussi **size > 4.**

- Métrique [0.0,..1.0] proposée: proportion de feuilles de profondeur height ou height-1.
- A noter qu'il y a nécessairement au moins une feuille de profondeur height (puisque l'arbre est de hauteur height!). Pour que le ratio puisse prendre la valeur 0.0, il faut donc enlever le compte de cette feuille au numérateur et au dénominateur.

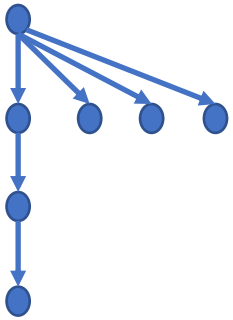
- Métrique à couvrir:

$$\text{Leafb\_rate} = \frac{\text{Number of leaves at depth } \textit{height} \text{ or } \textit{heigth-1} \; - \; 1}{\text{Total number of leaves} \; - \; 1}$$

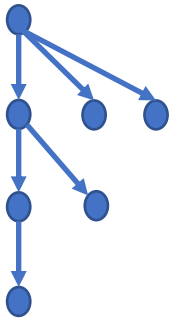
[0, 0.33[[0.33, 0.67]]0.67, 1.0]
low medium high

- Note 1: la métrique est bien définie, car il y a plus d'une feuille. On a exclu le cas trivial height =size – 1, qui correspond à un arbre linéaire.
- Note2: une métrique plus compliquée serait de considérer la profondeur moyenne des feuilles. J'ai fait l'analyse pour la normaliser, mais préférer la métrique ci-dessus dans un premier temps.

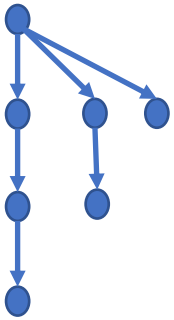
Leaf-balanced : exemples



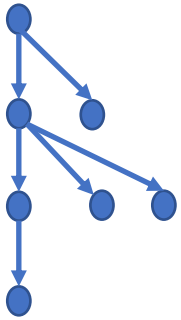
Leafb_rate = 0.0



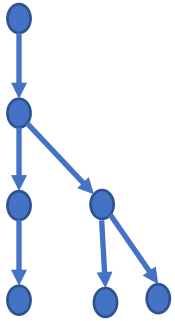
Leafb_rate = 0.33



Leafb_rate = 0.5



Leafb_rate = 0.67



Leafb_rate = 1.0

Height-balanced

- Propriété (cas arbre k-ary, cf. article Sha cité plus haut) : Tous les sous-arbres d'un nœud non terminal doivent avoir la même hauteur à +/- 1 près. Attention, on considère des nœuds à k fils, un sous-ensemble de fils pouvant éventuellement être l'arbre nul (de hauteur -1).

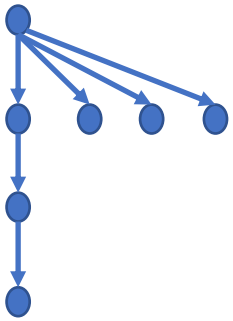
- Ma reformulation de la propriété, sans contraintes sur le nombre k de fils: Tous les sous-arbres d'un nœud non terminal doivent avoir la même hauteur à +/- 1 près. Si un nœud n'a qu'un fils, on compare sa hauteur à celle de l'arbre nul de hauteur -1. Intéressant à mesurer quand **height > 1**. En fait, tout (sous-)arbre de hauteur 1 est trivialement height-balanced. Il y a au moins un nœud de profondeur height -1 dans l'arbre, qui introduit un sous-arbre trivialement équilibré. On l'enlève du compte, pour permettre à la métrique de prendre la valeur 0.0.

- Métrique à couvrir: proportion de nœuds non terminaux qui sont racines d'un arbre équilibré, en enlevant 1 au numérateur et au dénominateur.

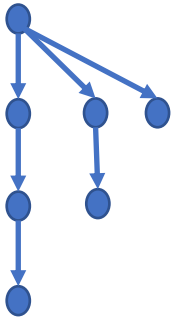
$$\text{Heightb\_rate} = \frac{\text{Number of internal nodes with subtrees of similar height} - 1}{\text{Total number of internal nodes} - 1}$$

[0, 0.33[	[0.33, 0.67]	]0.67, 1.0]
low	medium	high

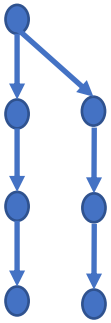
Height-balanced : exemples



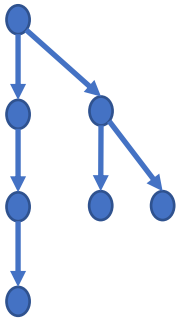
Heightb_rate = 0.0



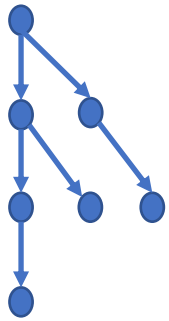
Heightb_rate = 0.33



Heightb_rate = 0.5



Heightb_rate = 0.67



Heightb_rate = 1.0

Size-balanced

- Propriété (cas arbre k-ary, cf. article Sha cité plus haut) : Tous les sous-arbres d'un nœud non terminal doivent avoir la même taille à +/- 1 près. Attention, on considère des nœuds à k fils, un sous-ensemble de fils pouvant éventuellement être l'arbre nul (de taille 0).

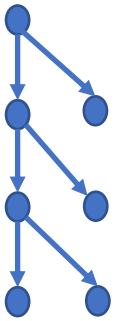
- Ma reformulation de la propriété, sans contraintes sur le nombre k de fils: Tous les sous-arbres d'un nœud non terminal doivent avoir la même taille à +/- 1 près. Si un nœud n'a qu'un fils, on compare sa taille à celle de l'arbre nul de taille 0. Intéressant à mesurer quand **height > 1**. En fait, tout (sous-)arbre de hauteur 1 est trivialement size-balanced.

- Métrique à couvrir: proportion de nœuds non terminaux qui sont racines d'un arbre équilibré, en enlevant 1 au numérateur et au dénominateur.

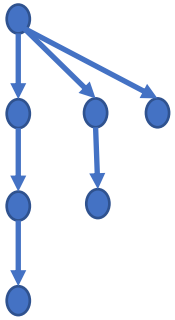
$$\text{Sizeb\_rate} = \frac{\text{Number of internal nodes with subtrees of similar size} - 1}{\text{Total number of internal nodes} - 1}$$

[0, 0.33[	[0.33, 0.67]	]0.67, 1.0]
low	medium	high

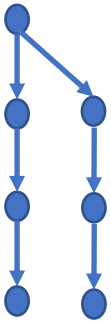
Size-balanced : exemples



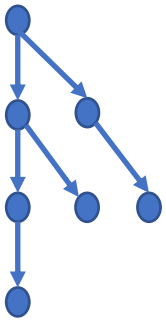
Sizeb_rate = 0.0



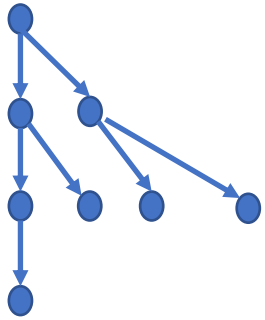
Sizeb_rate = 0.33



Sizeb_rate = 0.5



Sizeb_rate = 0.67



Sizeb_rate = 1.0

Note: d'après l'article Sha, tout arbre size-balanced est aussi leaf-balanced et height-balanced.
Les valeurs élevées de Sizeb_rate vont probablement être difficiles à couvrir.

Récapitulatif couverture custom

- Height_vs_size (mesuré pour **size ≥ 4**)
- Leafb_rate (mesuré pour **size > 4 et 2 < height < size – 1**)
- Sizeb_rate, Heightb_rate (mesurés pour **height > 1**).

- Je propose d'homogénéiser tout ça. A mesurer quand **size > 4 et 2 < height < size – 1** dans tous les cas.