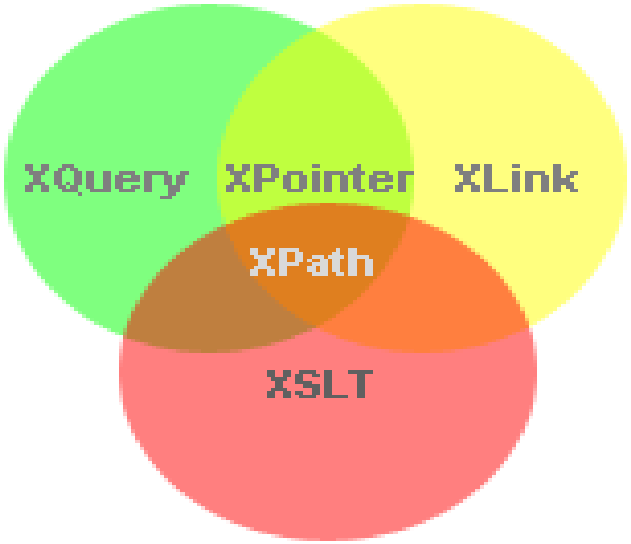
XPath

- XPath pour XML Path Langage
- XPath est un langage pour adresser les sous arbres d'un arbre XML
- . XPath permet de:
  - ✓ Naviguer dans un arbre XML
  - ✓ Rechercher un ou plusieurs éléments dans un document
  - ✓ Référencer tout fragment d'un document



Objectif: exprimer des requêtes pour localiser des parties d'un document XML

Xpath

Ex: quels est le contenu du message n°2 ?

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <messages>
        <message numero="1" date="2018-01-01">
            <dest>promo2018</dest>
            <dest bcc="oui">Pierre Nerzic</dest>
            <contenu>Bonne année !</contenu>
        </message>
        <message numero="2">
            <dest>promo2018</dest>
            <contenu>Bonne rentrée !</contenu>
        </message>
    </messages>
```

/messages/message[@numero=4]/contenu

- Lorsqu'on parcourt un arbre afin d'obtenir une ou plusieurs valeurs, on peut passer par des nœuds intermédiaires, appelés nœuds de ***contexte***
- **XPath** est fondé sur des chemins de localisation.
- Un chemin est une séquence d'étapes et peut être :
  - **absolu**: Si le chemin commence par '/', alors il représente un chemin absolu vers l'élément requis.

    Exemple **/A/B/@att1**
  - **relatif**: accepte n'importe quel nœud de l'arbre XML comme point de départ. relative au nœud courant: **//A/B/@att1**

- Une expression **XPath** est une suite **d’étapes** séparées par des **séparateurs** : [sep] étape1 sep étape2 sep étape3. . .

- **/ :** ce **séparateur** se comporte comme dans Unix, s’il est mis au début du chemin, il représente le document entier (il représente un chemin absolu vers l'élément demandé) ; s’il est mis entre les étapes, c’est un simple séparateur.

Ex:

/AAA/DDD/BBB

Select all elements BBB which are children of DDD which are children of the root element AAA

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC/>
</AAA>
```

- Si le chemin commence par //, tous les éléments du document répondant aux critères suivants sont sélectionnés. *Exemple*: **//dest** sélectionne tous les éléments **<dest>** où qu'ils soient dans l'arbre

//BBB	
Select all elements BBB	

- **Ex:**

```
<AAA>
   <BBB/>
   <CCC/>
   <BBB/>
   <DDD>
      <BBB/>
   </DDD>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
      </DDD>
   </CCC>
</AAA>
```

- * sélectionne tous les éléments localisés par le chemin précédent

- **Ex1:**

/AAA/CCC/DDD/*
Select all elements enclosed by elements /AAA/CCC/DDD

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

- **Ex2:**

/*/*/BBB

Select all elements BBB which have 3 ancestors

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

- Pour désigner un attribut et non pas un sous-élément, on met un **@** devant le nom de l'attribut.

Exemples :

➢ **/messages/message/@numero** sélectionne tous les nœuds **Attributs** nommés **numéro** des éléments <**message**>.

➢ **//@numero** sélectionne tous les nœuds attributs portant ce nom n'importe où dans l'arbre.

➢ **Ex:**

```
                                                    //@id
```

Select all attributes @id

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

- D’autres étapes peuvent être employées :

✓ **.** désigne le nœud courant

✓ **..** désigne le nœud parent,

✓ ***** désigne tous les éléments de ce niveau.

✓ | regroupe les résultats de deux expressions **Xpath**

Exemples :

/messages/*/@numero sélectionne tous les nœuds **Attributs** nommés **numero** des éléments situés sous **messages**

//dest/@* sélectionne tous les nœuds **Attributs** des éléments **dest** du document.

//message/dest|//message/contenu sélectionne les éléments **dest** et **contenu** avec deux chemins complets.

Conditions sur les étapes

L’une des forces de **XPath** est de pouvoir rajouter des conditions appelées **prédicats** sur les étapes d’un chemin. Un prédicat se met entre **[...]** juste après l’élément dont il filtre l’un des enfants. On peut mettre plusieurs prédicats:

element[n] : sélectionne le nième élément **element** dans le nœud courant

element[elt]: sélectionne dans le nœud courant, l'élément **element** qui a comme élément **fils elt**

element[@x]: sélectionne dans le nœud courant, l'élément **element** qui possède un attribut **x**

[@x='valeur']: sélectionne dans le nœud courant, l'élément dont l'attribut **x** a une valeur égale à **valeur**

- **Ex:**

```
//BBB[@id]
```

Select BBB elements which have attribute id

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

- **Ex:**

```
//BBB[@*]
```

Select BBB elements which have any attribute

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

```
//BBB[not(@*)]
```

Select BBB elements without an attribute

```
<AAA>
   <BBB id = "b1"/>
   <BBB id = "b2"/>
   <BBB name = "bbb"/>
   <BBB/>
</AAA>
```

- **Ex:**

```
//BBB[@id='b1']
```

Select BBB elements which have attribute id with value b1

```
<AAA>
   <BBB id = "b1"/>
   <BBB name = " bbb "/>
   <BBB name = "bbb"/>
</AAA>
```

```
//BBB[@name='bbb']
```

Select BBB elements which have attribute name with value 'bbb'

```
<AAA>
   <BBB id = "b1"/>
   <BBB name = " bbb "/>
   <BBB name = "bbb"/>
</AAA>
```

Fonctions Xpath

XPath possède de très nombreuses fonctions, dont :

- **position()** retourne l'index de l'élément dans son parent (premier = n°1)
- **last()** retourne le n° du dernier élément dans son parent
- **count(expression)** compte le nombre de nœuds XML (élément, attributs, textes. . .) sélectionnés par l'expression
- **string-length(s)** retourne la longueur de la chaîne **s**
- **contains(s1, s2)** vrai si s1 contient s2
- **starts-with(s1,s2)** et **ends-with(s1,s2)**

Exemples

- **/messages/message[position()<=3]** sélectionne les 3 premiers éléments **message** du document.
- **//dest[position()>last()-3]** sélectionne les **dest** qui sont parmi les trois derniers enfants de leur parent.
- **//message[count(dest)>2]** retourne les éléments **message** ayant plus de deux enfants **dest**.
- **//message[string-length(contenu)<=15 and not(starts-with(dest, "promo"))]/@numero** retourne les **numéros** des **messages** dont le contenu ne fait pas plus de **15** caractères et aucun **destinataire** ne commence par « **promo** ».

- **Ex:**

```
//*[count(BBB)=2]
```

Select elements which have two children BBB

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

- **Ex:**

/AAA/BBB[1]

Select the first BBB child of element AAA

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
</AAA>
```

/AAA/BBB[last()]

Select the last BBB child of element AAA

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
</AAA>
```

- **Ex:**

```
//BBB[position() mod 2 = 0 ]
```

Select even BBB elements

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
   <CCC/>
   <CCC/>
   <CCC/>
</AAA>
```

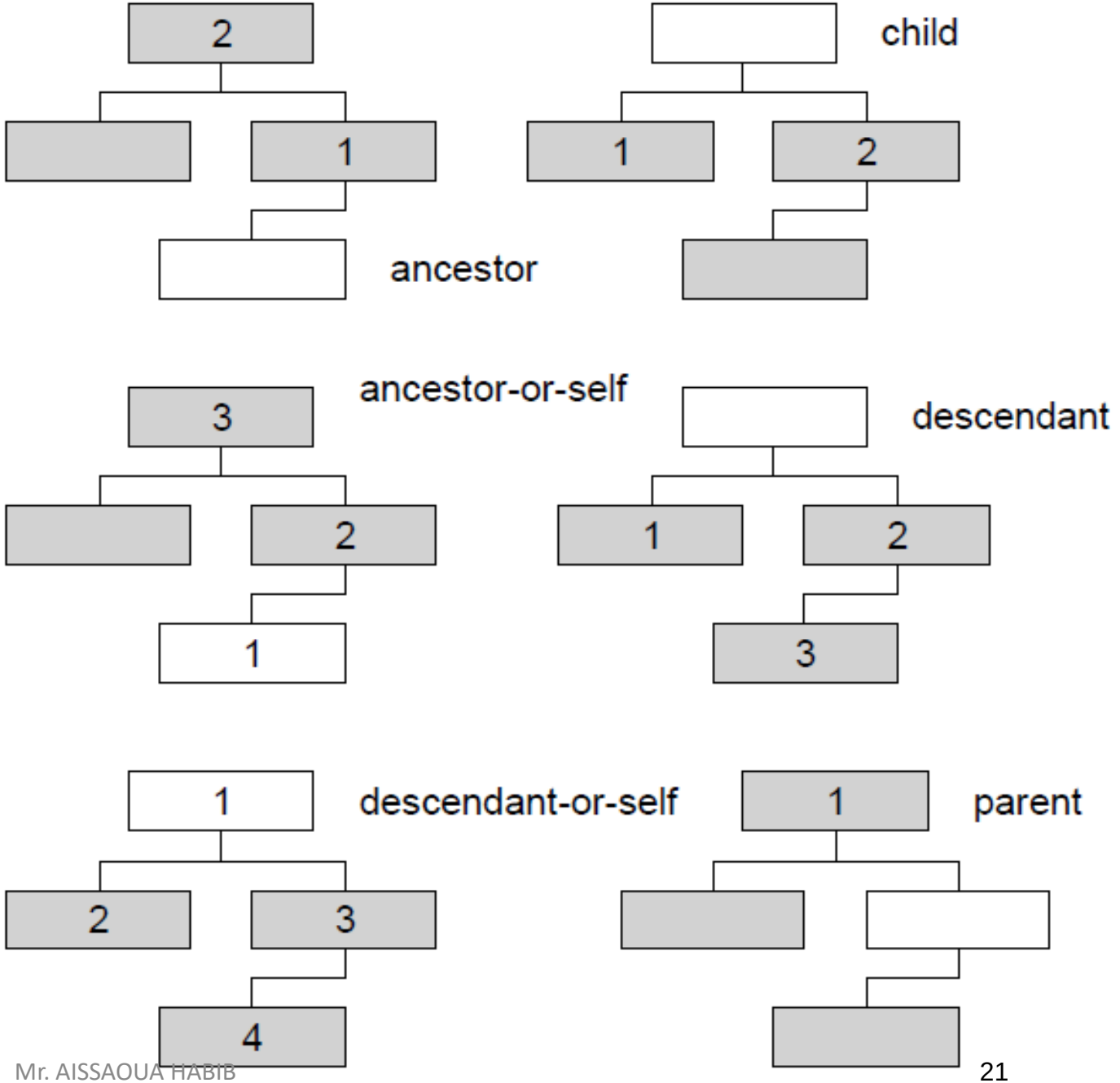
Retour sur les composants d'un chemin:

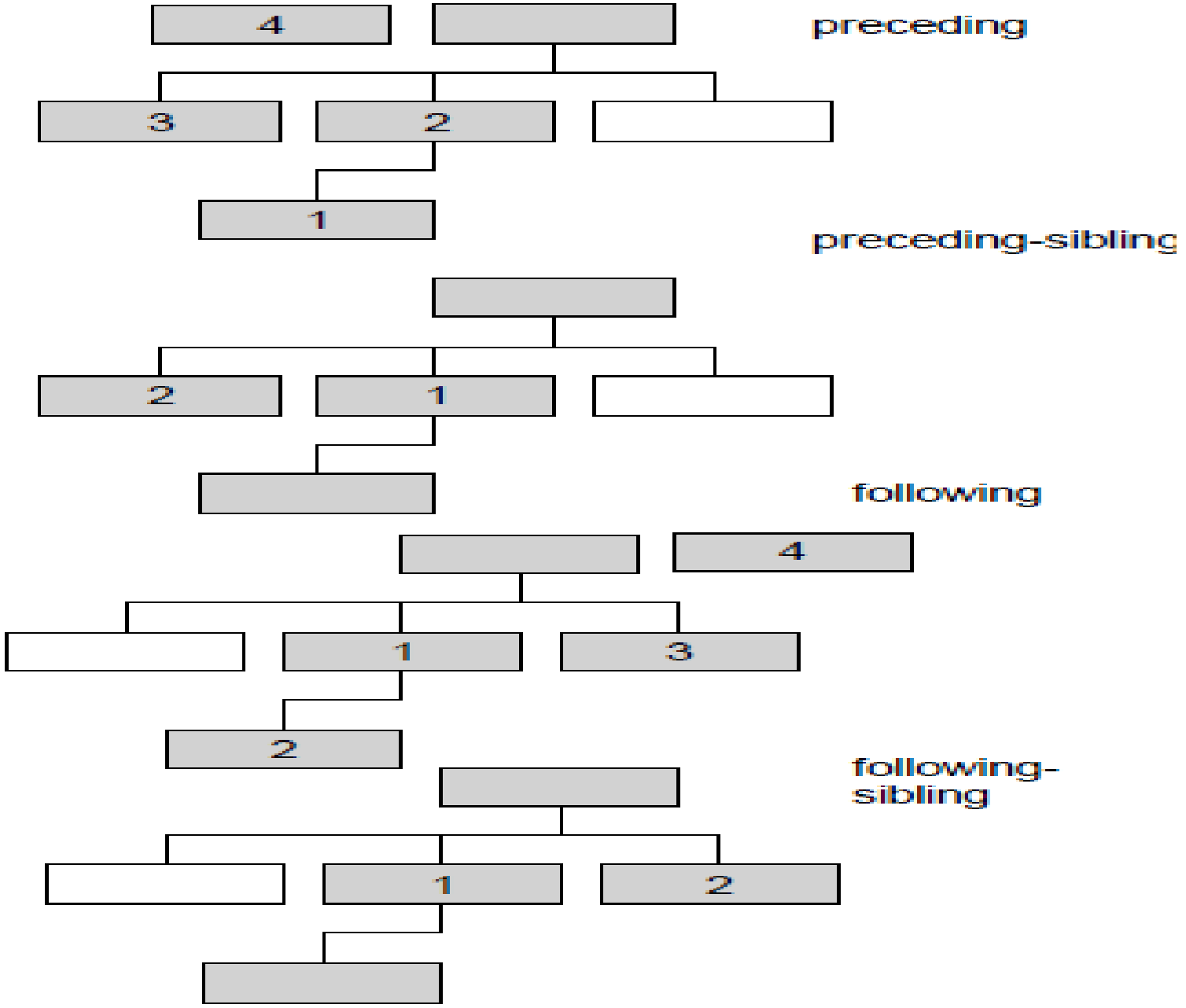
- Une expression XPath est une suite d'étapes: [sep] étape1 sep étape2 sep étape3. . .
- Une étape est de la forme: **`[axe::]filtre[prédicat]*`**
- **Axe** : sens de parcours des nœuds;
- **filtre** : type des nœuds qui seront retenus (à localiser)
- 0 à *n* **prédicats** : des conditions à respecter (expression booléenne à vérifier sur chaque nœud).

XPath supporte plusieurs axes

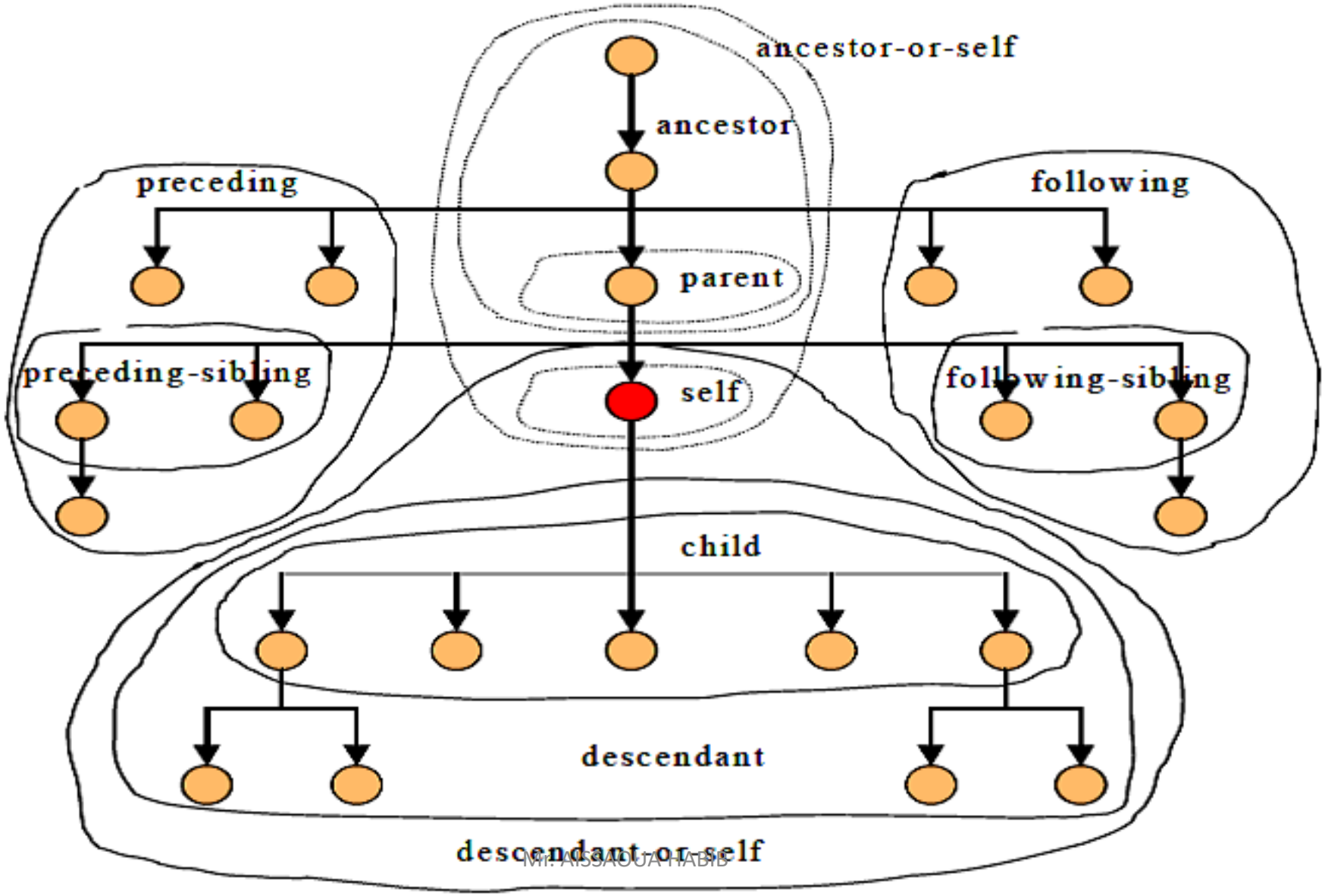
- Les axes en avant: ***child, self, descendant, descendant-or-self***
- Les axes en arrière: ***parent, ancestor, ancestor-or-self***
- Les axes à gauche et à droite: ***following-sibling, preceding-sibling***
- Les axes avant et après: ***following, preceding***
- Les axes pour les attributs: ***attribute***

- Dans cette figure, l'axe est désigné en anglais. Le rectangle blanc correspond au point de départ. Les numéros correspondent à l'ordre de la réponse. Si nous prenons l'axe courant ***child***, on constate que le parcours correspond à la recherche des descendants d'un nœud de premier niveau (les fils).




4
preceding
3
2
1
preceding-sibling
2
1
following
4
1
3
2
following-
sibling
1
2

Axes de déplacement



- **Ex:**

```
//GGG/ancestor::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <FFF>
            <HHH/>
            <GGG>
               <JJJ>
                  <QQQ/>
               </JJJ>
               <JJJ/>
            </GGG>
            <HHH/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

- **Ex:**

```
//GGG/descendant::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <FFF>
            <HHH/>
            <GGG>
               <JJJ>
                  <QQQ/>
               </JJJ>
               <JJJ/>
            </GGG>
            <HHH/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

- **Ex:**

```
//GGG/following::*
```

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <FFF>
        <HHH/>
        <GGG>
          <JJJ>
            <QQQ/>
          </JJJ>
          <JJJ/>
        </GGG>
        <HHH/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

- **Ex:**

```
/AAA/XXX/DDD/EEE/ancestor-or-self::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
      </ZZZ>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

- **Ex:**

```
//GGG/preceding::*
```

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

- **Ex:**

```
//CCC/preceding-sibling::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

- **Ex:**

```
//CCC/following-sibling::*
```

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

- **Ex:**

```
//DDD/parent::*
```

Select all parents of DDD element

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

- **Ex:**

```
//CCC/descendant::*
```

Select all elements which have CCC among its ancestors

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

- **Ex:**

```
//CCC/descendant::DDD
```

Select elements DDD which have CCC among its ancestors

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

- **Ex:**

/child::AAA

Equivalent of /AAA

```
<AAA>
   <BBB/>
   <CCC/>
</AAA>
```

/AAA/BBB

Equivalent of /child::AAA/child::BBB

```
<AAA>
   <BBB/>
   <CCC/>
</AAA>
```

- **Ex:**

/child::AAA/child::BBB

Equivalent of /AAA/BBB

```
<AAA>
   <BBB/>
   <CCC/>
</AAA>
```

/child::AAA/BBB

Both possibilities can be combined

```
<AAA>
   <BBB/>
   <CCC/>
</AAA>
```

Une étape est de la forme: **`[axe::]filtre[prédicat]*`**

Les filtres servent à reconnaître chaque type de nœud:

- ***Nom*** : sélectionner les élément dont le nom est **nom**
- ***:** orienter la recherche vers **tous** les nœuds, mais pas du texte et du commentaire
- **text()**: orienter la recherche vers les nœuds de type **texte**
- **comment()**: orienter la recherche vers les nœuds de type **commentaire**
- **node()**: orienter la recherche vers tous les types de nœuds
- **processing-instruction()**: orienter la recherche vers les instructions de traitement

- **Ex:**

1) **//h1/following-sibling::p[1]**

   extraire uniquement le premier paragraphe après chaque titre

2) **//div[@id='footer']/preceding-sibling::text()[1]**

   sélectionnez uniquement le texte qui se trouve juste avant **footer**

```
<html>
  <body>
    <p>Intro paragraph</p>
    <h1>Title #1</h1>
    <p>A random paragraph #1</p>
    <h1>Title #2</h1>
    <p>A random paragraph #2</p>
    <p>Another one #2</p>
    A single paragraph, with no markup
    <div id="footer"><p>Footer text</p></div>
  </body>
</html>
```