Module MLBDA
Master Informatique
Spécialité DAC

COURS 6 –XPATH

XPath

Langage permettant de sélectionner des nœuds dans un arbre XML

La construction de base est l'expression de chemin.

- prédicats pour spécifier les valeurs d'éléments ou d'attributs
- l'expression de chemin s'applique à un nœud de l'arbre et sélectionne les nœuds qu'on peut atteindre en suivant tous les chemins définis par cette expression.

XPath est une spécification W3C (version 1.0 en 1999).

XPath

http://www.w3.org/TR/xpath.html

Brique de base pour d'autres standards XML

- XSchema
- XLink : spécification des hyperliens
- XPointer : pointer des éléments de documents avec des expressions XPath dans les URL
- XSLT : langage de transformation de documents
- XQuery : langage de requêtes
- …

Exemple : Xpath pour la définition de clés XML Schema

```
<xs:element name="Librairie">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Livre" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Titre" type="xs:string"/>
                        <xs:element name="Auteur" type="xs:string"/>
                        <xs:element name="ISBN" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:key name="Lclef">
        <xs:selector xpath="Livre"/>
        <xs:field xpath="ISBN"/>
    </xs:key>
</xs:element>
```

Contenu

Arbres XML

- Définition
- Ordre du document
- Sérialisation

Langage Xpath

- Expression de chemin
- Evaluation
- Prédicats
- Attributs
- Axes

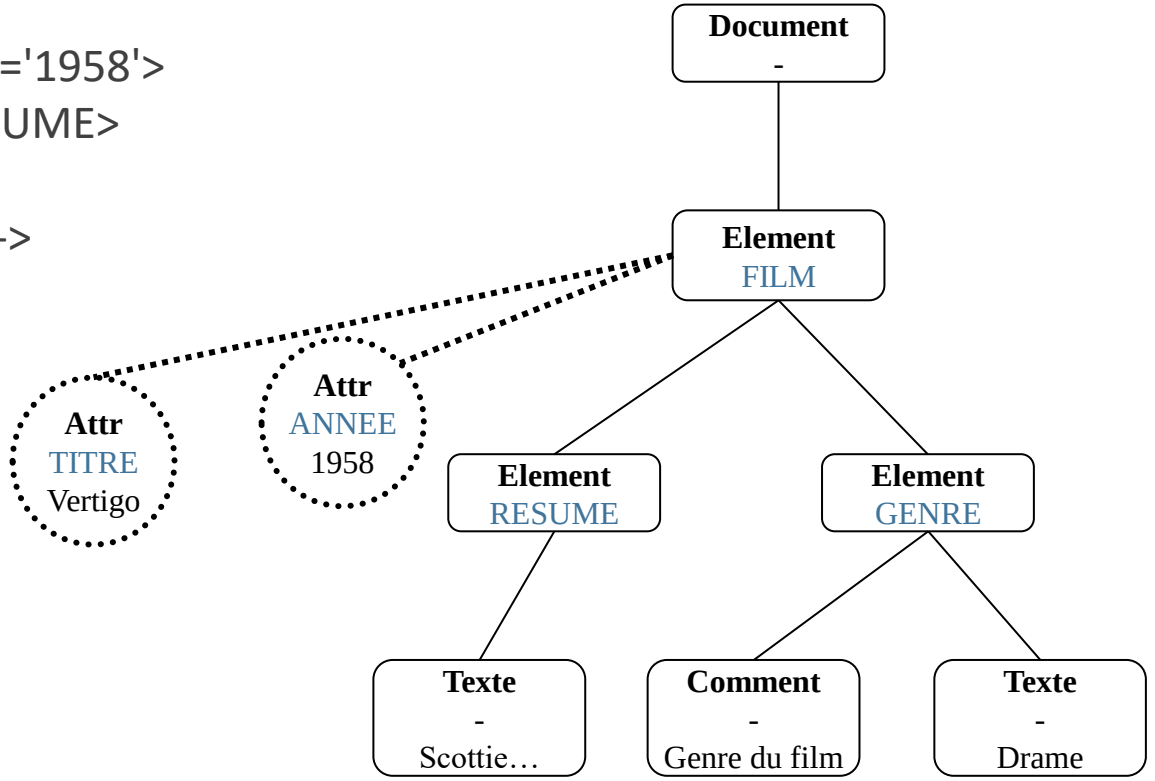
Arbres XML

L'arbre d'un document comporte 7 types de nœuds associés à chaque constituant d'un document :

- Document (document XML)
- Element (élément XML)
- Attr (attribut XML)
- Text (valeurs textuelles)
- Comment (commentaire)
- ProcessingInstruction (instruction de traitement)
- Namespace (espace de noms)

Exemple : document XML et arbres DOM

```
<? xml version= ''1.0'' ?>
<FILM titre='vertigo' annee='1958'>
<RESUME> Scottie…..</RESUME>
<GENRE>
        <-- genre du film -->
Drame
</GENRE>
</FILM>
```



Structure d’un Arbre DOM

Seuls les nœuds racine (documents) et les éléments ont des *enfants* :

- **nœuds fils** de type
  - élément
  - texte
  - commentaire
  - instruction de traitement.
- **mais pas** : attributs ou espaces de nom

Les *descendants* du nœud racine ou d'un nœud élément sont ses enfants ou les enfants de ses enfants.

L'ensemble des nœuds est muni d'un ordre qui correspond à l'ordre dans le document XML.

Analyse d'un document XML et sérialisation d'un arbre DOM

Analyse d'un document XML : génération de l'arbre DOM à partir du fichier XML :

- Lecture séquentielle du document XML
  - Génération d'un nœud document au début de la lecture;
  - A chaque balise ouvrant, créer un éléments fils du nœud actuel (document ou élément) avec le nom de la balise;
  - Traiter ensuite le contenu jusqu'à la balise fermante : attributs, texte, commentaires, balises (récursion)
  - A chaque balise fermante, remonter au parent du nœud actuel.

Sérialisation : passage de la forme arborescente à la forme sérialisée (document)

- Parcours de l'arbre par la racine, puis de gauche à droite et en profondeur d'abord :
  - Le nœud Document génère <? xml version= '1.0' ?>
  - A la première visite d'un nœud Element, on crée une balise ouvrante du nom de l'élément, et on ajoute un attribut pour chaque nœud attribut visité
  - Lorsqu'on sort d'un élément, on crée la balise fermante de l'élément.

Contenu et Valeur textuelle d'un noeud

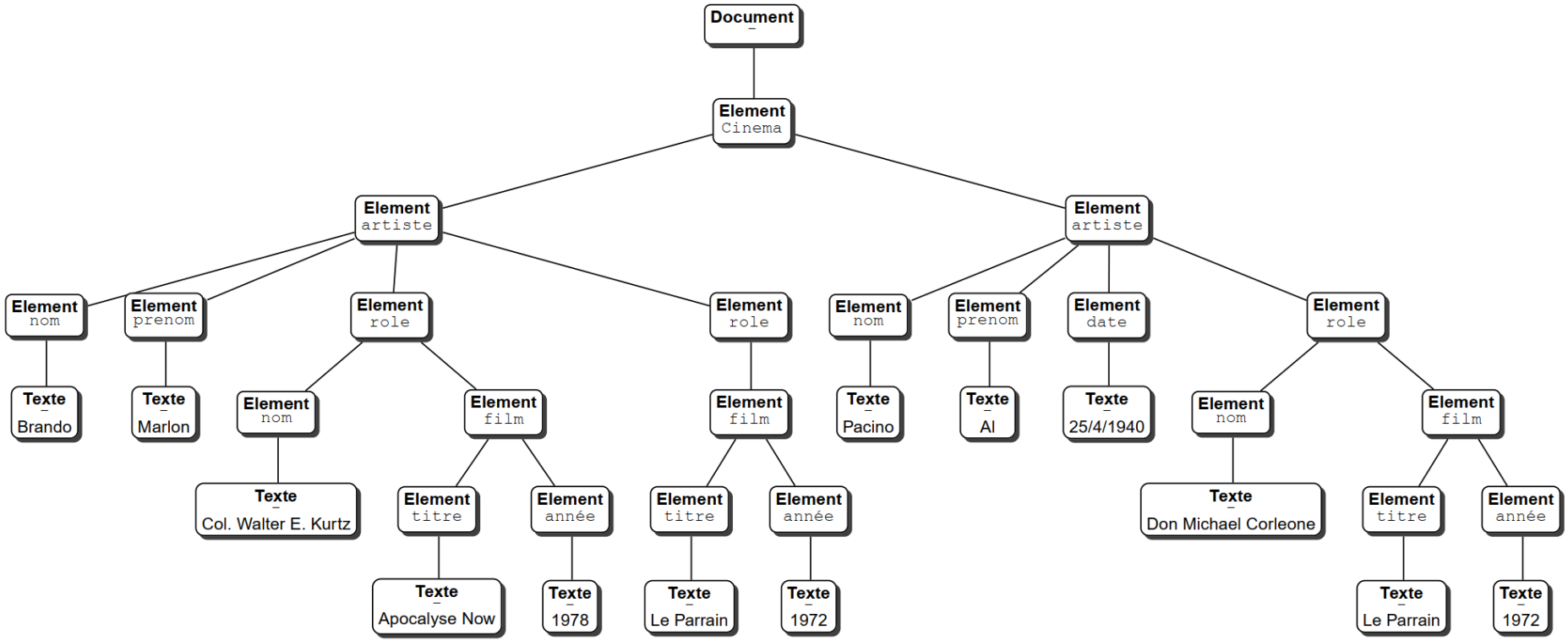
Chaque nœud a une valeur textuelle :

- la valeur textuelle du nœud racine (document) ou d'un nœud élément est la concaténation des valeurs textuelles de ses descendants de type texte, dans l'ordre du document;
- la valeur textuelle d'un nœud texte est la chaîne de caractères constituant ce texte;
- la valeur textuelle d'un nœud attribut est la chaîne de caractères constituant la valeur de cet attribut.

Attention :

- Ne pas confondre le **contenu d'un élément** qui est le **fragment XML** (ou un sous-arbre) délimité par ses balises avec sa **valeur** qui est la **concaténation des valeurs de ses nœuds descendants** de type **texte**.

Arbre DOM


Document
Element Cinema
Element artiste
Element artiste
Element nom
Element prenom
Element role
Element role
Element nom
Element prenom
Element date
Element role
Texte Brando
Texte Marlon
Element nom
Element film
Element film
Texte Pacino
Texte Al
Texte 25/4/1940
Element nom
Element film
Texte Col. Walter E. Kurtz
Element titre
Element année
Element titre
Element année
Texte Don Michael Corleone
Element titre
Element année
Texte Apocalyse Now
Texte 1978
Texte Le Parrain
Texte 1972
Texte Le Parrain
Texte 1972

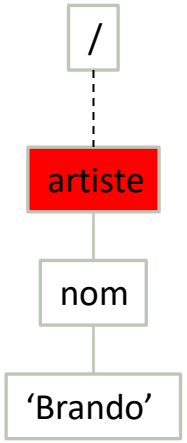
Principe : Motifs d'arbres conjonctifs

Un motif d'arbre conjonctif est un arbre étiqueté où

- Les noeuds sont étiquetés par des noms d'éléments ou d'attributs, des valeurs, ou le symbole '*'
- Les arcs correspondent aux relations enfant (arc simple) ou descendant (arc double).
- Les noeuds resultat sont entourés.

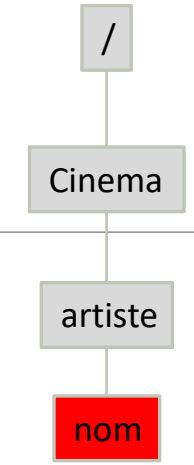
Films avec M. Brando :

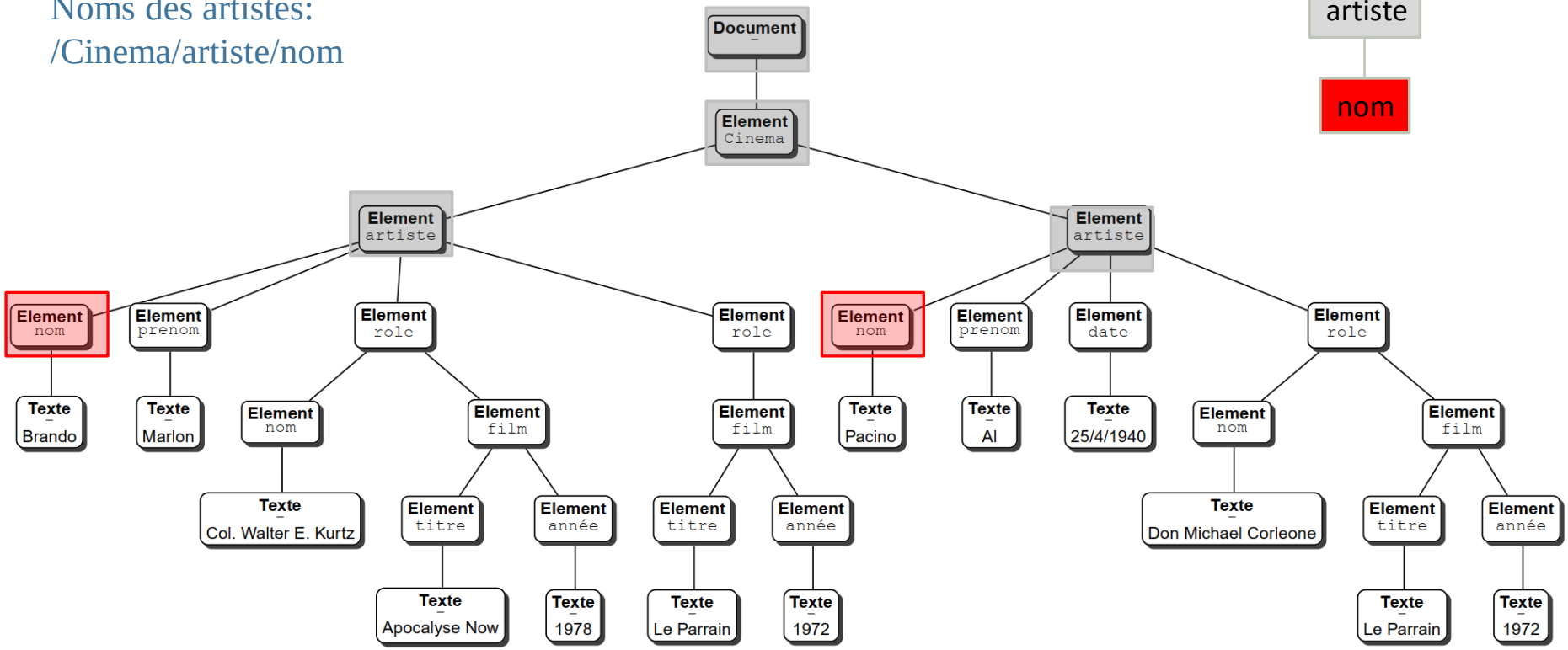
- //artiste[nom='M. Brando']



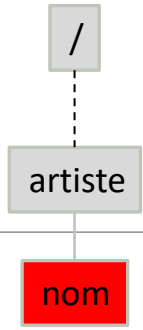
XPath : Principe

Noms des artistes:
/Cinema/artiste/nom

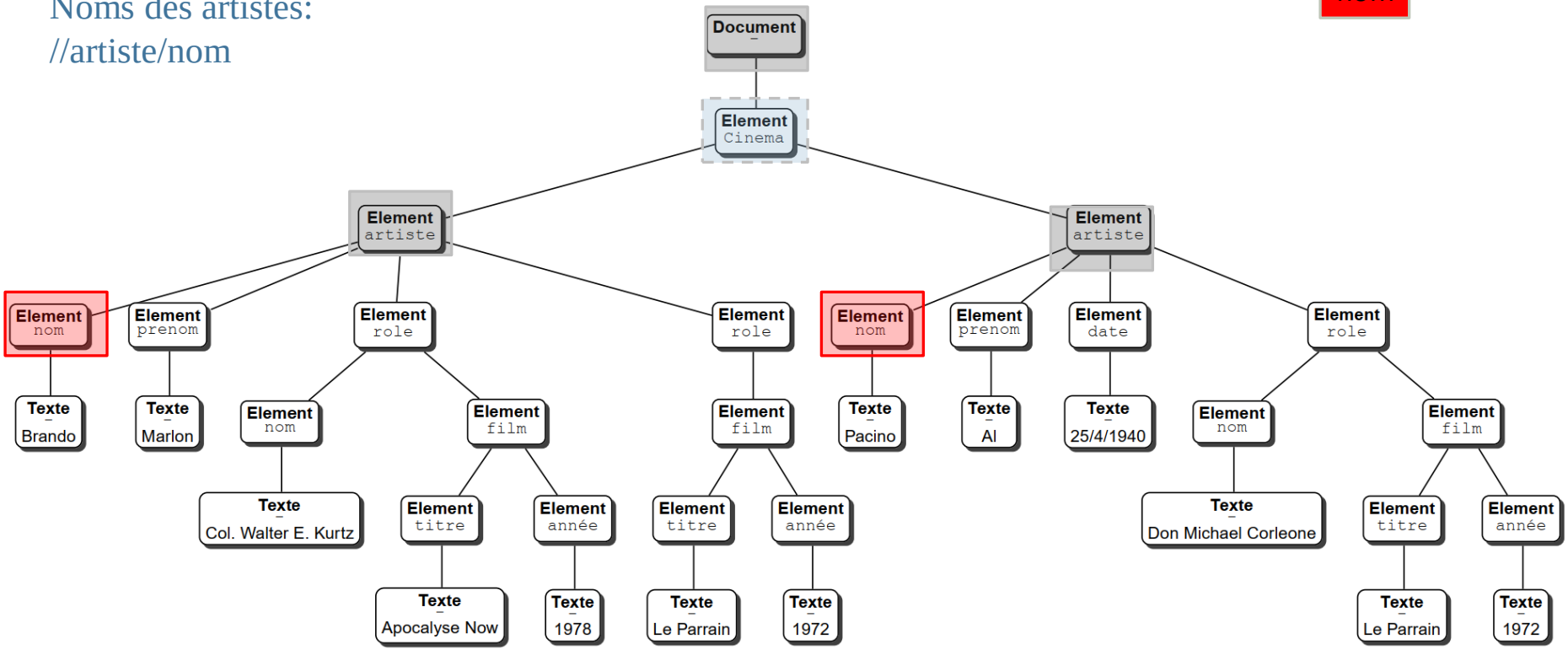




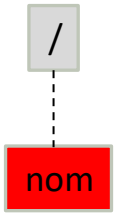
XPath : Principe



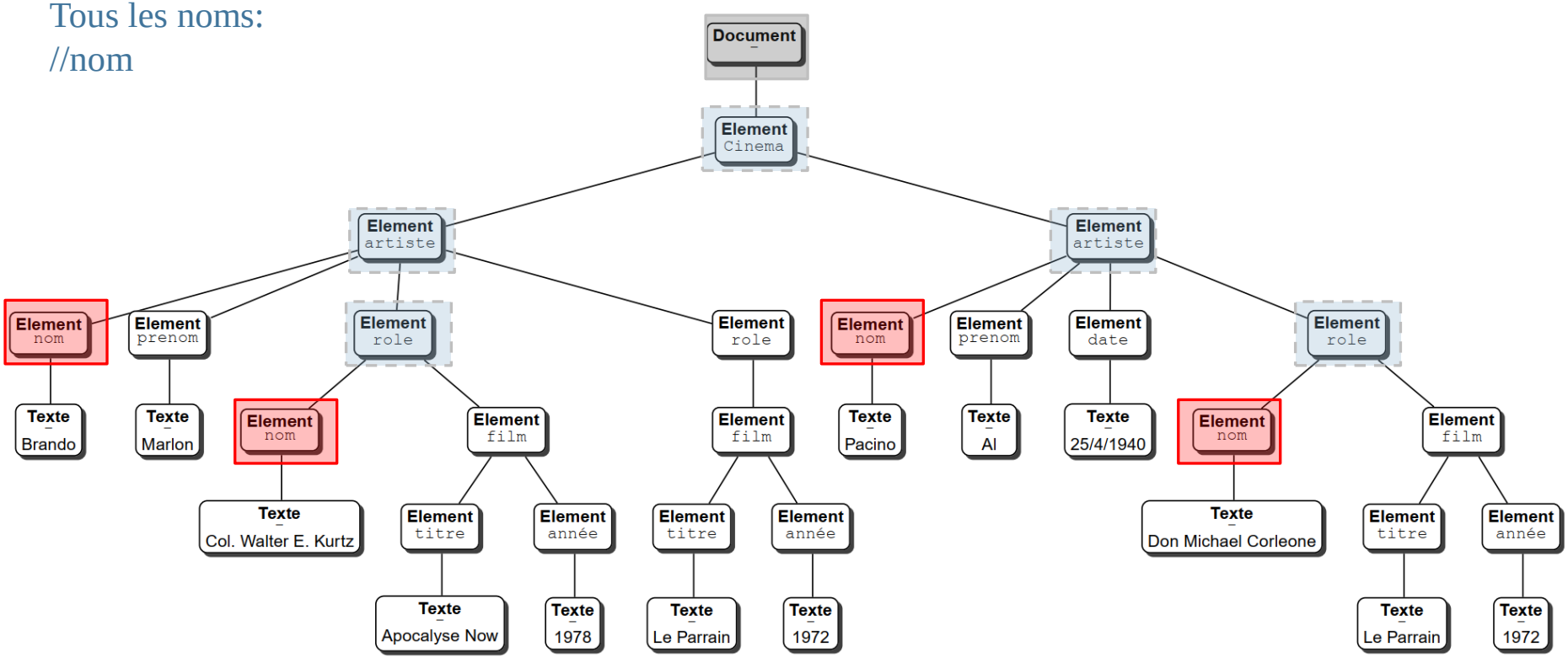
Noms des artistes:
//artiste/nom



XPath : Principe

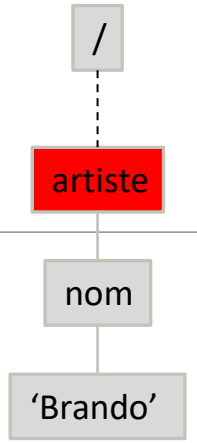


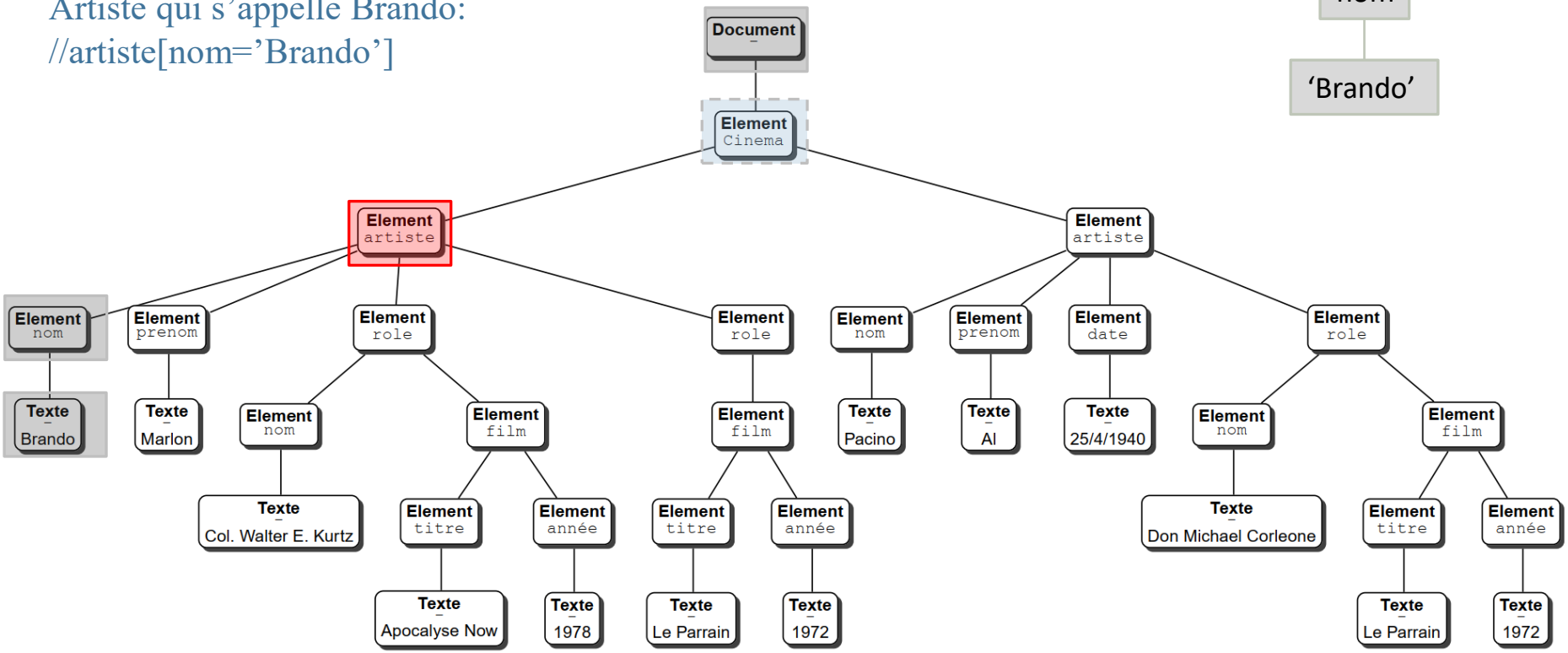
Tous les noms:
//nom



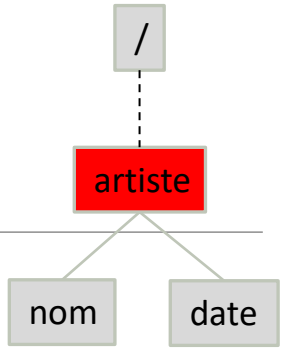
XPath : Principe

Artiste qui s'appelle Brando:
//artiste[nom='Brando']

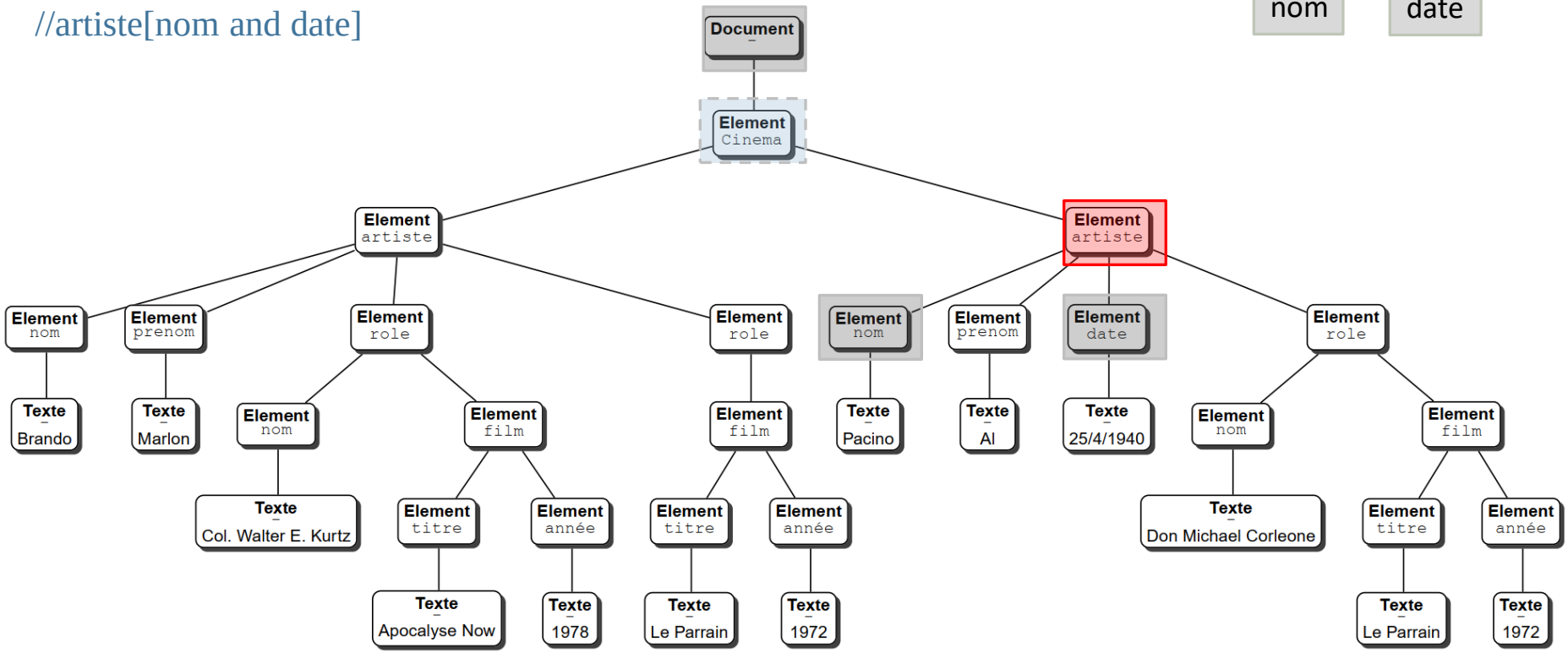




XPath : Principe

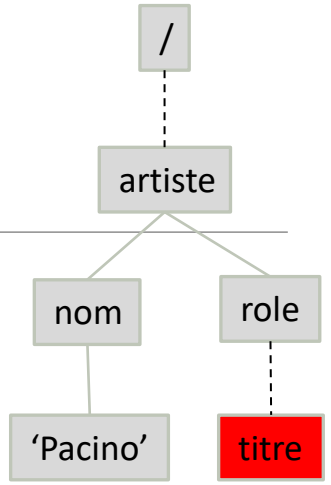


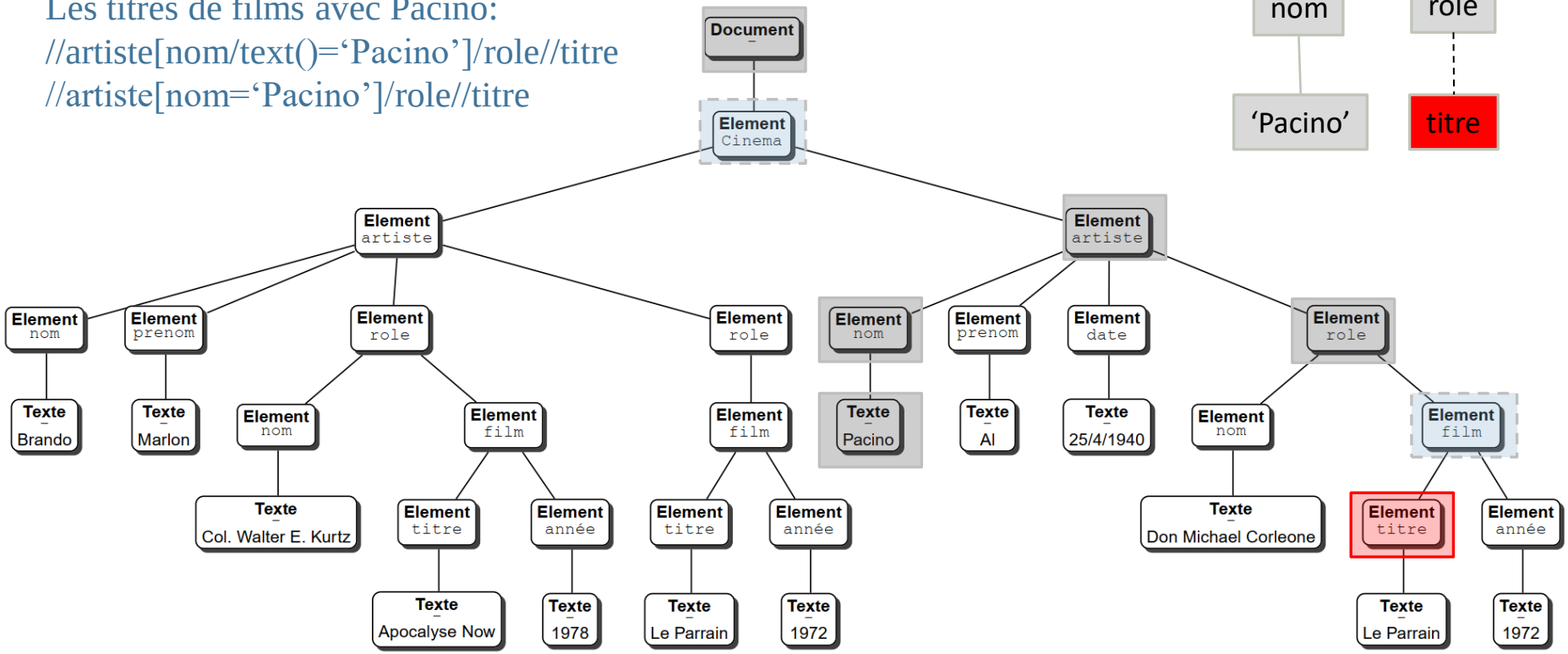
Artistes avec un nom et une date de naissance:
//artiste[nom and date]



XPath : Principe

Les titres de films avec Pacino:
//artiste[nom/text()='Pacino']/role//titre
//artiste[nom='Pacino']/role//titre

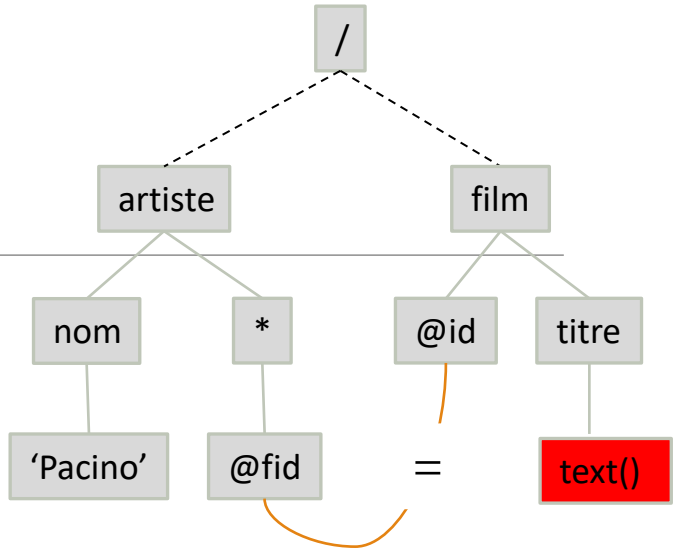


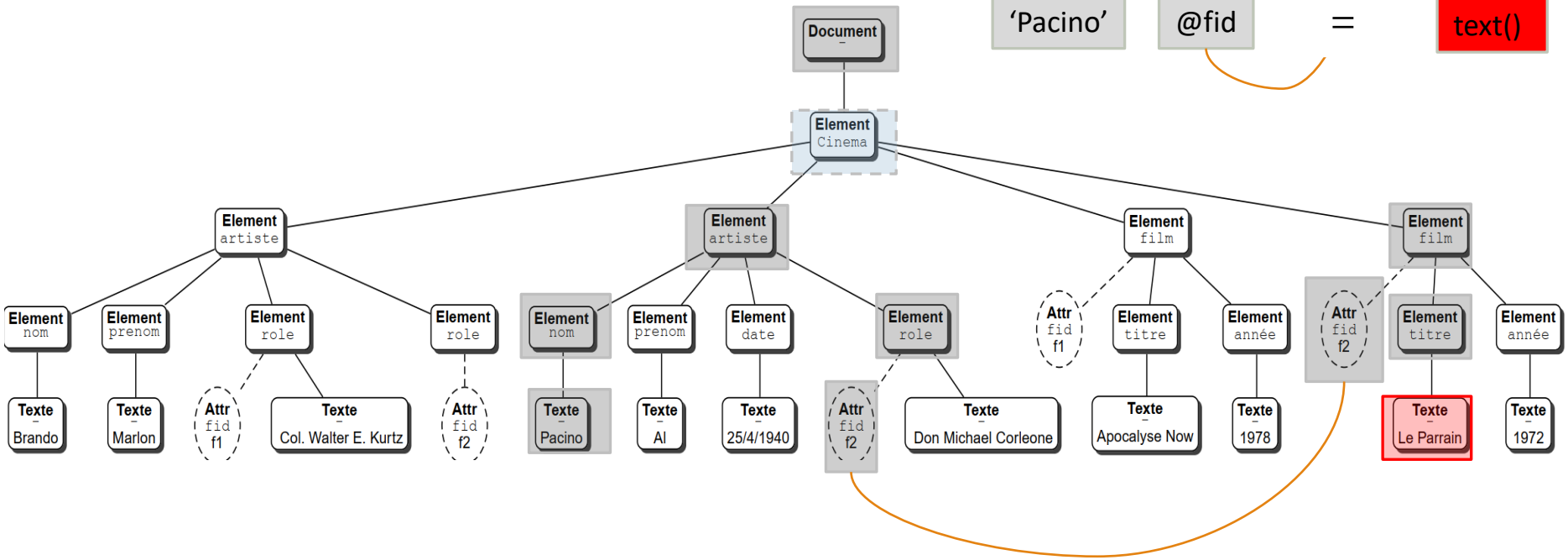


Xpath : Principe

Les titres de films avec Pacino:

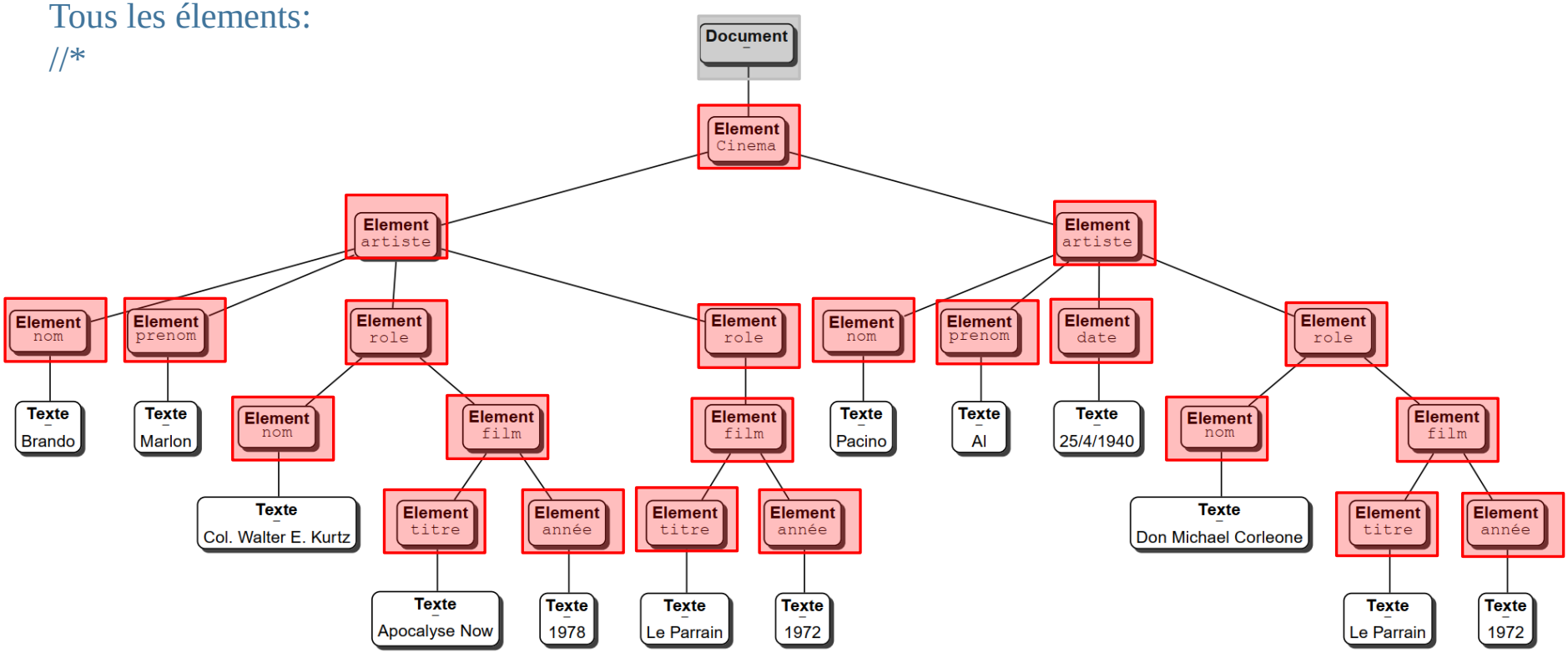
//film[@fid=//artiste[nom ='Pacino']*/@fid]/titre/text()





XPath : Principe

Tous les élements:
//*



XPath : langage de navigation

XPath est un langage pour extraire des noeuds dans un arbre XML :

- On navigue dans l'arbre grâce à des ***chemins de navigation***.
- Un une expression chemin est une séquence **d'étapes**.
- Le résultat d'une étape (d'une séquence d'étapes) est un séquence de nœuds.

Expression de chemin XPath

Une expression Xpath est une séquence d'étapes, séparées par des '/' :

[/]étape1/étape2/…/étapen

Une étape a la forme suivante :

axe::filtre[prédicat1][prédicat2]

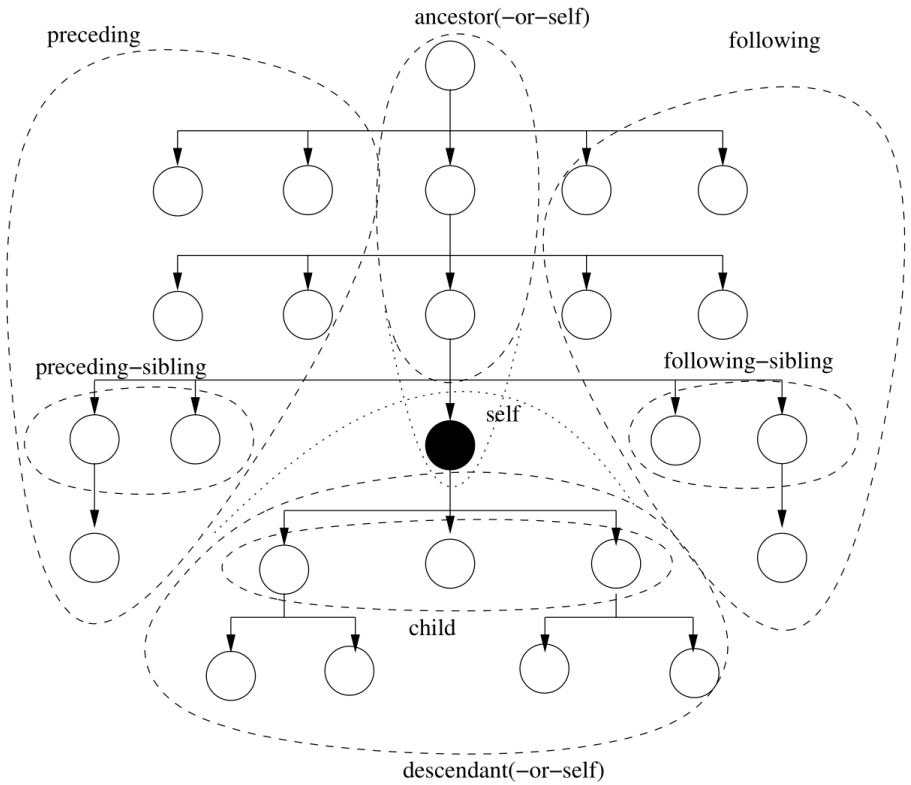
- L'axe définit le sens du parcours des nœuds
- Le filtre indique le type des nœuds qui seront retenus dans l'évaluation de l'expression
- Le (ou les) prédicat(s) expriment des propriétés que doivent satisfaire les nœuds retenus.

Une expression XPath :

- s'évalue en fonction d'un **nœud contexte** (souvent la racine de l'arbre /)
- désigne un ou plusieurs chemins dans l'arbre à partir du nœud contexte
- a pour résultat un **ensemble de nœuds ou une valeur**, numérique, booléenne ou alphanumérique

Axes XPath

- **child** : les enfants du nœud contextuel.
- **self** : le nœud courant.
- **parent** : le parent du nœud contextuel.
- **descendant** : les descendants du nœud contextuel
- **ancestor** :les éléments ancêtres du nœud contextuel.
- **following-sibling** : tous les nœuds frères (nœuds qui ont le même parent) qui suivent le nœud contextuel.
- **preceding-sibling** :tous les frères qui précèdent le nœud contextuel.
- **following:** tous les nœuds qui sont visité après le nœud contextuel dans l'ordre du document, à l'exclusion des descendants.
- **preceding:** tous les prédécesseurs du nœud contextuel, à l'exclusion des ancêtres.
- **attribute** : tous les attributs du nœud contextuel


preceding
ancestor(−or−self)
following
preceding−sibling
following−sibling
self
child
descendant(−or−self)

Filtres XPath

Filtrage par le nom :

- possible pour les types de nœuds qui ont un nom : Element, ProcessingInstruction et Attr

Filtrage par le type DOM :

- * : nœuds de type Element ou Attribute
- text() : nœuds de type Text
- comment() : nœuds de type Comment
- processing-instruction() : nœuds de type ProcessingInstruction
- node() recouvre tous les types de nœud

Expressions XPath

Un chemin XPath est une suite d'étapes :

[/]étape 1 /étape 2 /.../étape n

Un chemin peut être absolu :

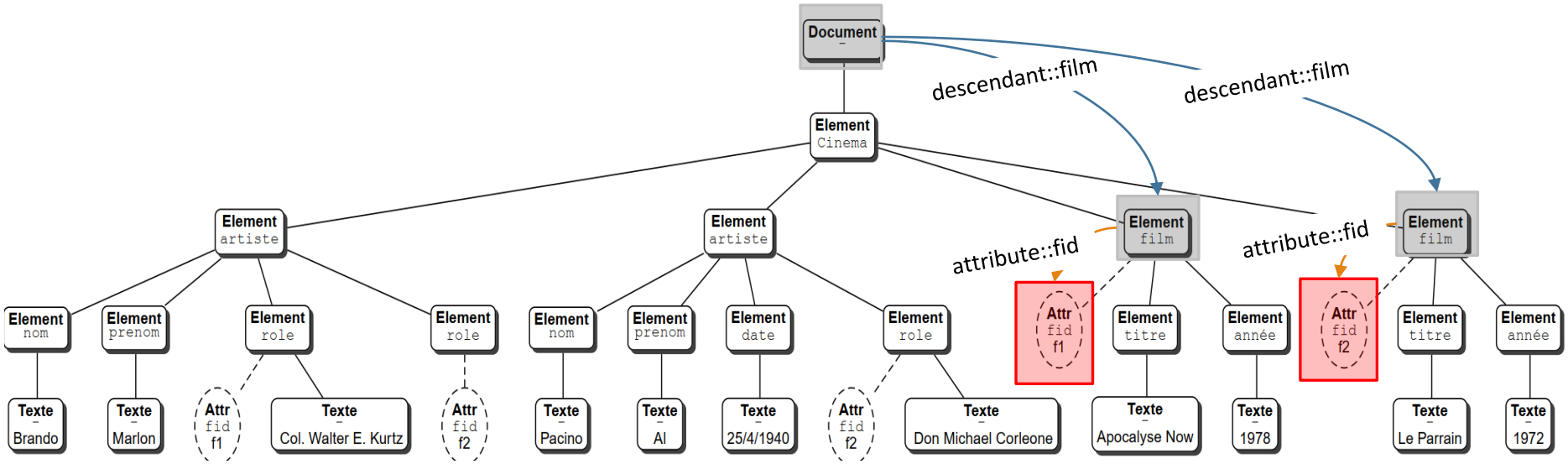
- /child::FILM/child::RESUME
- Le nœud contexte est la racine du document.

ou relatif :

- child::RESUME/child::text()
- Le nœud contexte est un nœud dans le document (pas forcément la racine).

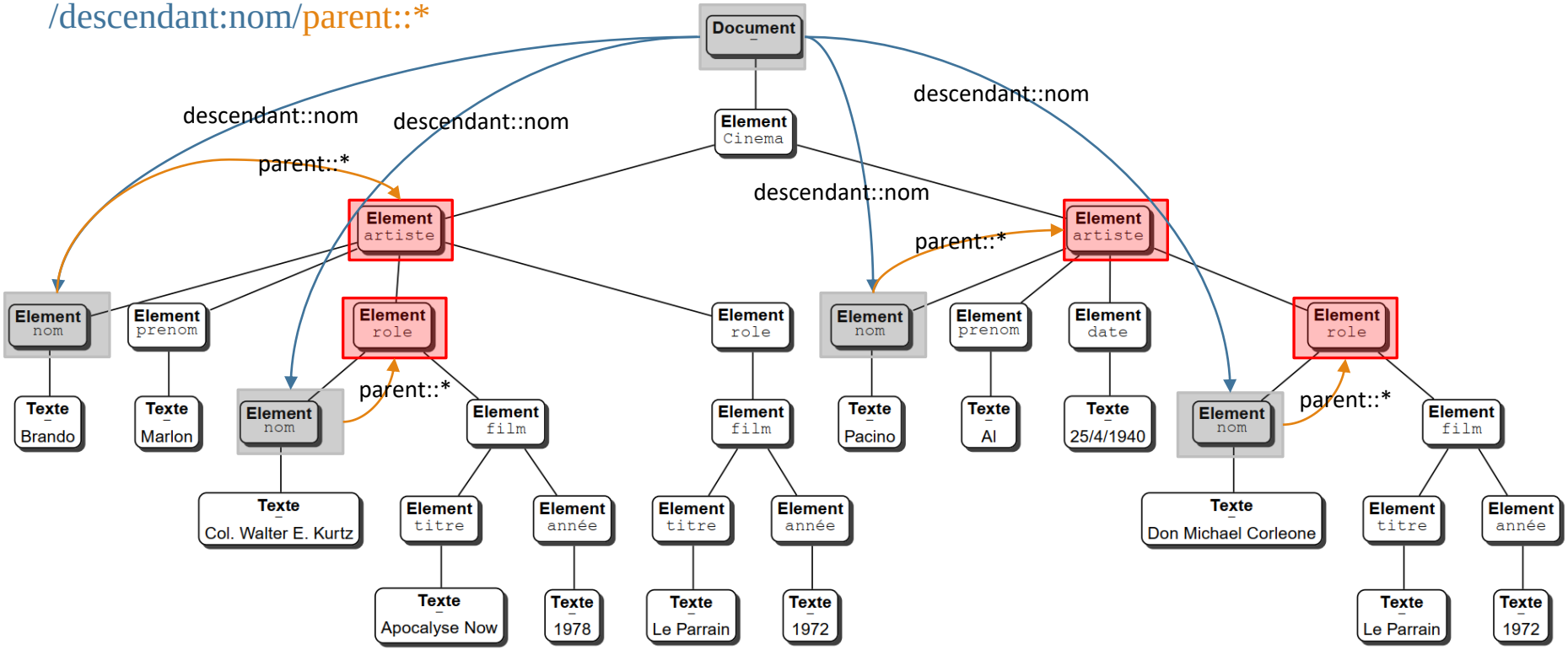
Xpath : Exemple

/descendant::film/attribute::fid
/descendant::film/@fid

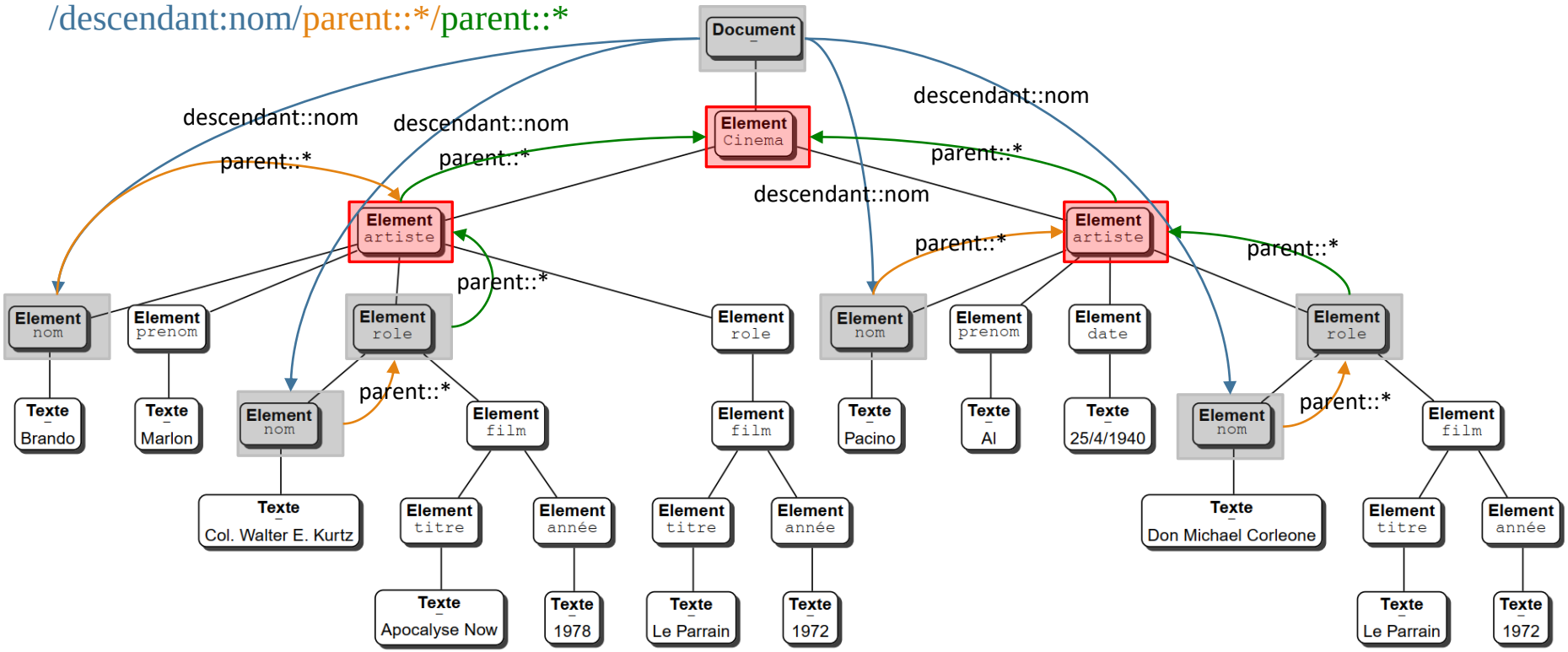


Xpath : Exemple

/descendant:nom/parent::*

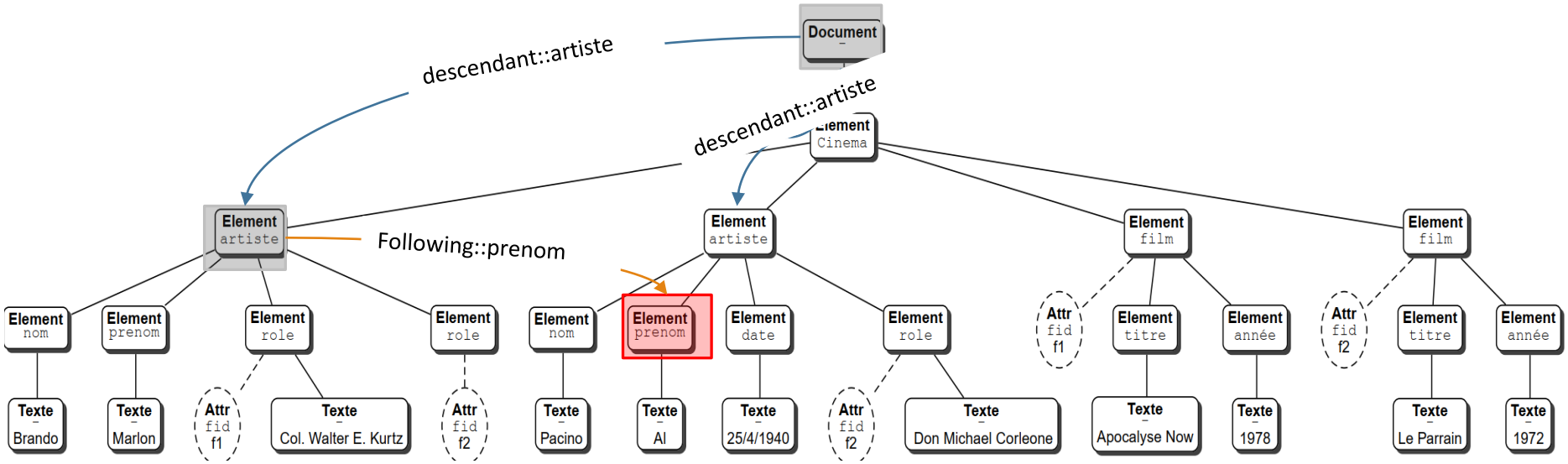


Xpath : Exemple



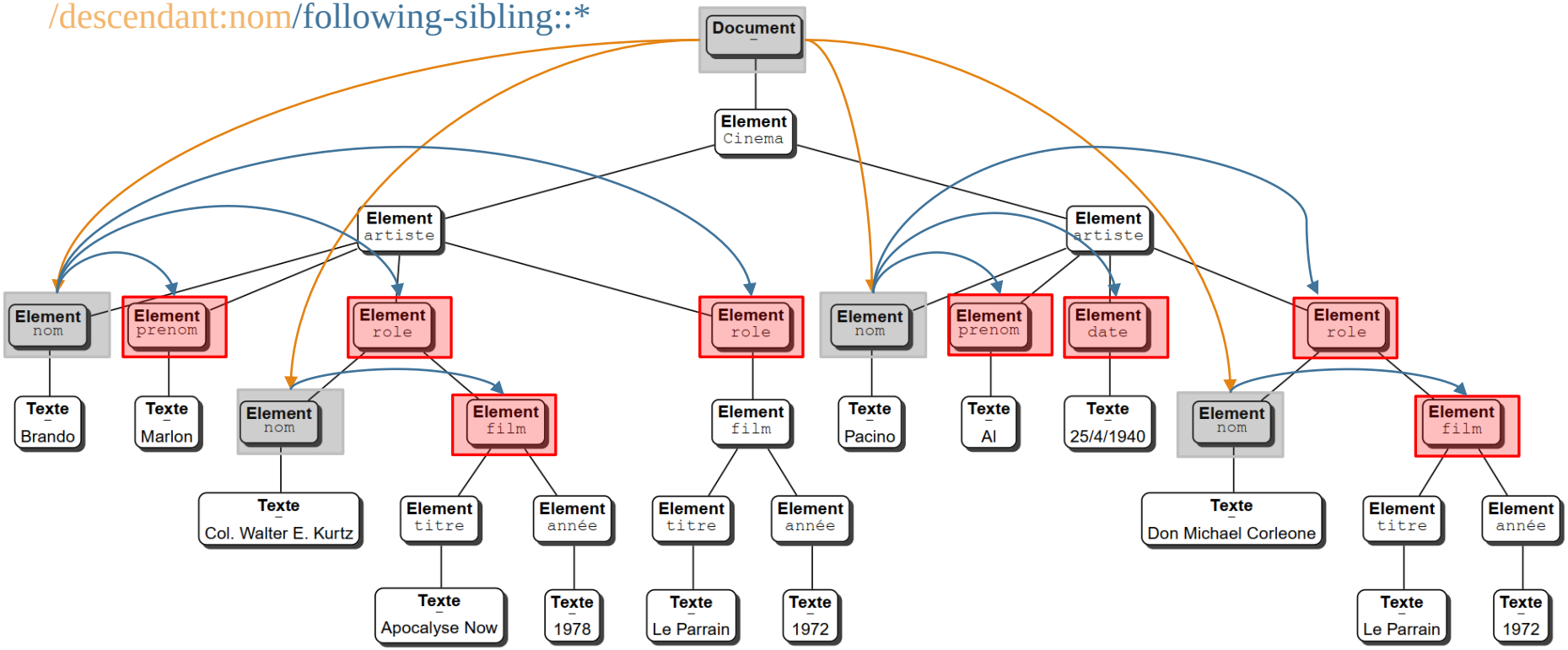
Xpath : Exemple

/descendant:artiste/following::prenom



Xpath : Exemple

/descendant:nom/following-sibling::*



Syntaxe abrégée et syntaxe étendue

	abrégée	étendue
Axe par défaut	E	child::E
Étape par défaut	//	/descendant-or-self::node()/
Parent	..	parent::node()
Self	.	self::node()
Attribut	@A	attribute::A

Exemples :

film/annee	child::film/child::annee
/cinema//artiste	/child::cinema/descendant-or-self::node()/child::artiste
//*/artiste/role	/descendant-or-self::node()/child::*/child::artiste/child::role
film/../film/@lang	child::film/parent::node()/child::film/attribute::lang

Prédicats XPath

Un **prédicat** (expression entre crochets) permet de spécifier plus précisément un élément.

Un prédicat est une expression booléenne constituée d'un ou plusieurs tests, composés avec les connecteurs logiques habituels not, and et or

Test :

- toute expression XPath, dont le résultat est converti en booléen;
- une comparaison, un appel de fonction.

Règles de conversion : Xpath → valeur Booléenne

CONVERSION

Pour les numériques :

- 0 ou NaN → false
- sinon → true

Pour les chaînes :

- chaîne vide → false
- sinon → true

Pour les ensembles de nœuds :

- ensemble vide → false
- sinon → true

EXEMPLES

Tous les éléments qui ont un élément fils A:

- /descendant::*[child::A]
- /descendant::*[count(child::A)]
- /descendant::*[count(child::A)>0]

Tous les éléments qui n'ont pas d'éléments fils A:

- /descendant::*[not(child::A)]
- /descendant::*[not(count(child::A))]
- /descendant::*[count(child::A)=0]

Fonctions

Positions relatives et information locale

- position() : position dans le contexte
- name() : retourne le nom de l'élément
- count() : cardinalité d'un node-set (compte le nb d'éléments sélectionnés)
- last() : indicateur de dernière position
- opérateurs booléens : and, or, not, >, >=, <, <=, =, !=
- opérateurs numériques : mod, div, +, -, *
- fonctions de chaîne : contains, substring-before, string-length, ...
- fonctions d'environnement : normalize-space (supprime les espaces de début et de fin, remplace les séquences d'espaces blancs par un seul espace.)

Permettent des requêtes de base

- analyse des contenus et noms de balises/attributs

Axes et position

La position d'un nœud dépend de l'axe choisi :

Axes «d'avancement» :

- child::*[position()=3] : le 3e enfant du nœud contexte
- child::*[3] : le 3e enfant du nœud contexte (syntaxe abrégée)
- child::*[last()] : le dernier enfant du nœud contexte
- /descendant::*[last()] : le dernier nœud (descendant) du document

Axes «inverses» :

- ancestor:: *[1]: le premier ancêtre du nœud contexte (parent).
- preceding-sibling::*[last()] : le dernier frère qui précède le nœud contexte.

= et !=

Attention : != n'est pas la négation de =

Comparaison d'une noeud A à un ensemble de nœuds N:

- A=N est vrai si la valeur textuelle d'un des nœuds est égale à la valeur textuelle de A
- A!=N est vrai si la valeur textuelle d'un des nœuds est différente de la valeur textuelle de A.

```
//a[.=//b]

//a[.!=//b]

//a[not(.=//b)]

//a[not(.!=//b)]
```

```
<r>
        <a>1</a>
        <a>5</a>
        <b>1</b>
        <b>2</b>
        <b>3</b>
</r>
```

Remarque (1)

```
<X>
 <A>
        <B>
                <C/>
                <C/>
        </B>
        <D>
                <C/>
                <C/>
                <C/>
        </D>
 </A>
</X>
```

```
//C[3]
```

3eme élément C d'un élément fils de la racine.
Équivalent à

```
/descendant-or-self::node()/child::C[3]
```

Troisième élément C du document :

```
/descendant::C[3]
```

Remarque (2)

/r/a[not(preceding-sibling::a=.)]

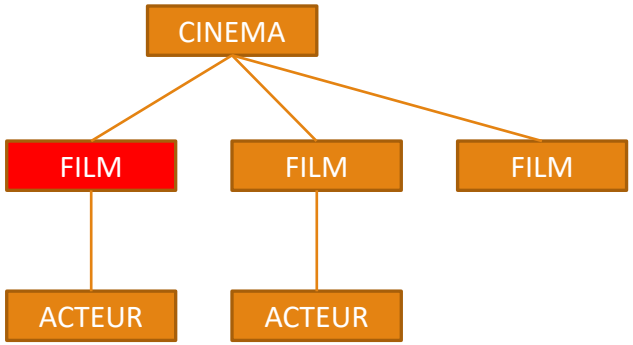
```
<r>
        <a>1</a>
        <a>5</a>
        <a>1</a>
        <a>2</a>
        <a>5</a>
</r>
```

Peut être utilisé pour éliminer les doublons

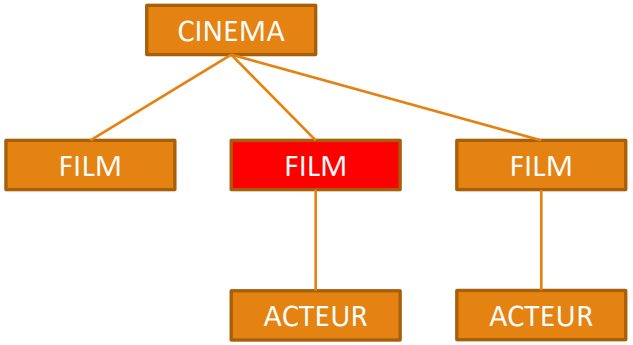
Remarque (3)

Attention : Les deux premières expressions sont 2quivalentes, mais pas la troisième! Pourquoi?

- FILM[ACTEUR and position()=1]
- FILM[position()=1][ACTEUR]
- FILM[ACTEUR][position()=1]



Réponse pour les trois expression



Réponse pour la 3e expression
(vide pour les deux premières)

Valeur textuelle

Remarque : la fonction text() a différentes implémentations selon les outils.

```
<?xml version="1.0" encoding="utf-8"?>
<A>
	<B att1='1'>
		<D>text 1</D>
		<D>text 2</D>
	</B>
	<B att1='2'>
		<D>text 3</D>
	</B>
	<C att2='a' att3='b'/>
</A>
```

//B//text()[1]
renvoie « text1 text3 »
//B/D/text()[1]
renvoie « text1 text2 text3 »
//B[1]//text()
renvoie « text1 text2 »
//B[1]//text() [1]
renvoie « text1 »

Evaluation d'une expression XPath

L'évaluation d'une étape peut être une valeur, ou un ensemble de nœuds (node-set).

Le nœud origine du chemin est le premier nœud contexte.

Les étapes sont évaluées les unes après les autres :

- À partir du nœud contexte, on évalue l'étape 1, qui renvoie un ensemble de nœuds.
- Chaque nœud de cet ensemble est considéré comme nœud contexte pour l'évaluation de l'étape 2.
- On procède de la même manière pour les autres étapes.

Remarques

- Les nœuds ne sont pas extraits du document
- Un nœud ne peut être référencé qu'une seule fois dans un même ensemble.

Exemple d'évaluation /A/B/@att1

1ère étape : /A/B/@att1

- A partir de la racine, on cherche les nœuds de type élément de balise A

2ème étape : B/@att1

- À partir d'un nœud contexte A, on cherche les nœuds de type élément de balise B

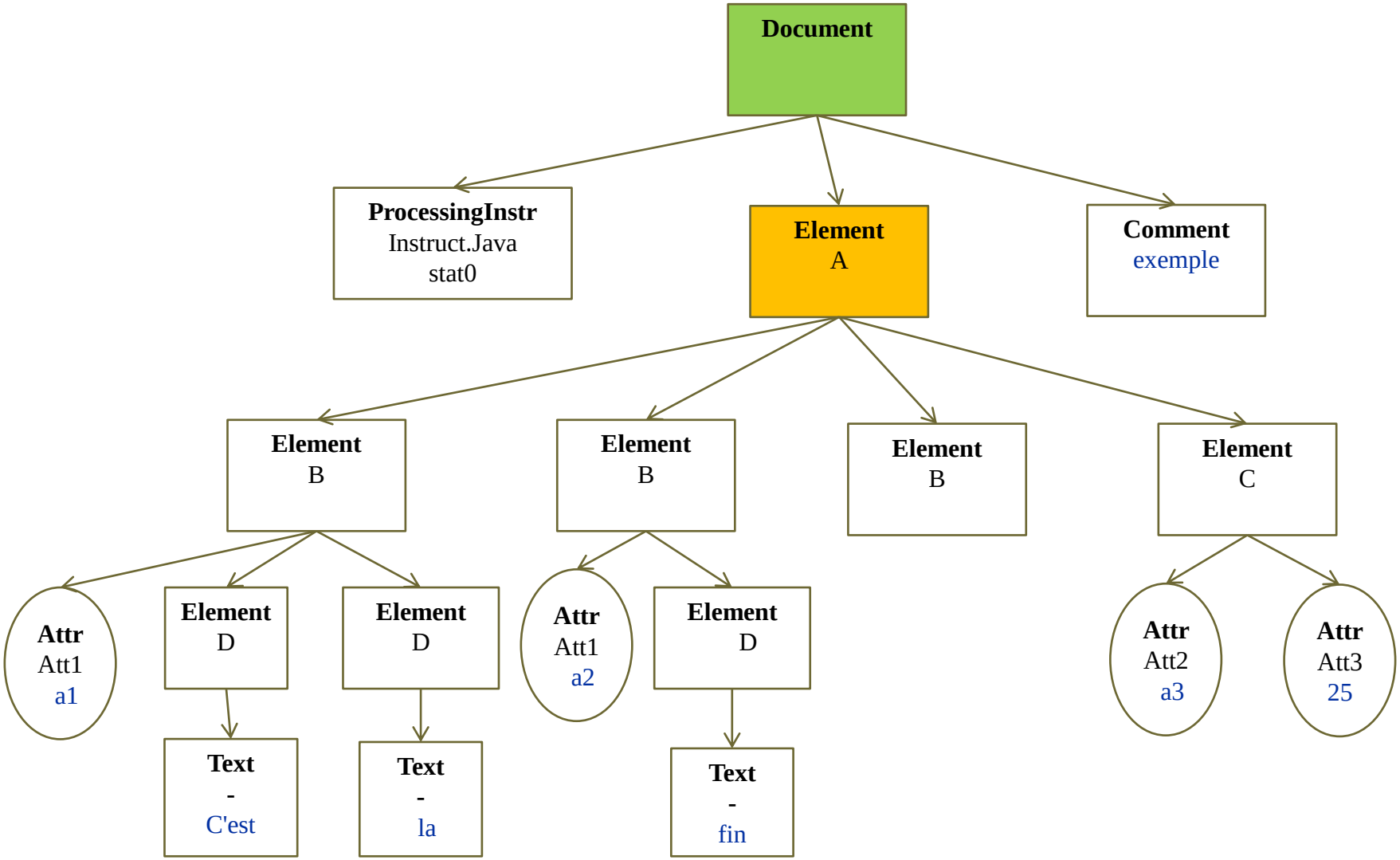
3ème étape : @att1

- À partir d'un nœud contexte B on cherche les nœuds de type attribut att1.

Etape 1:

/A

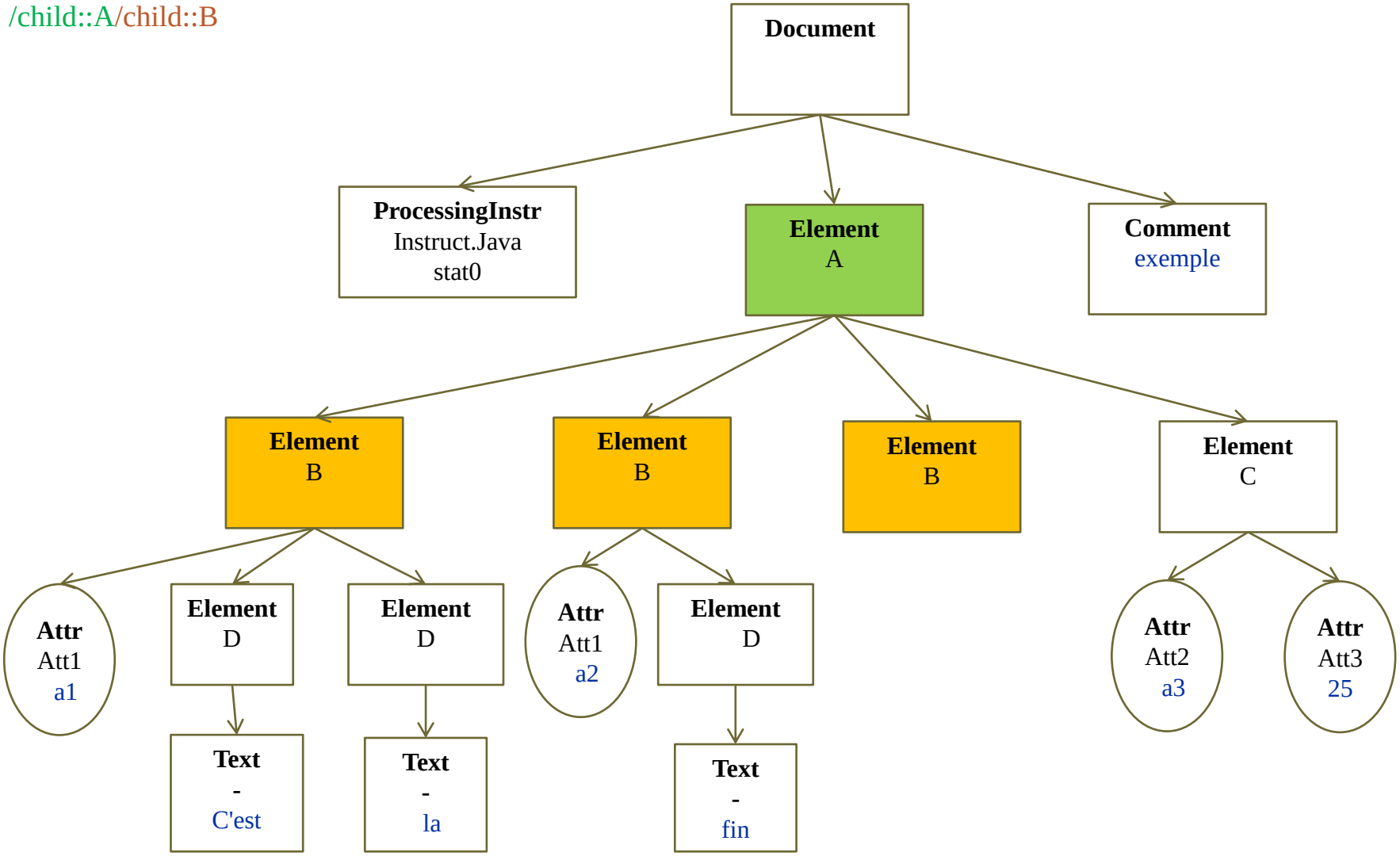
/child::A



Etape 2

/A/B

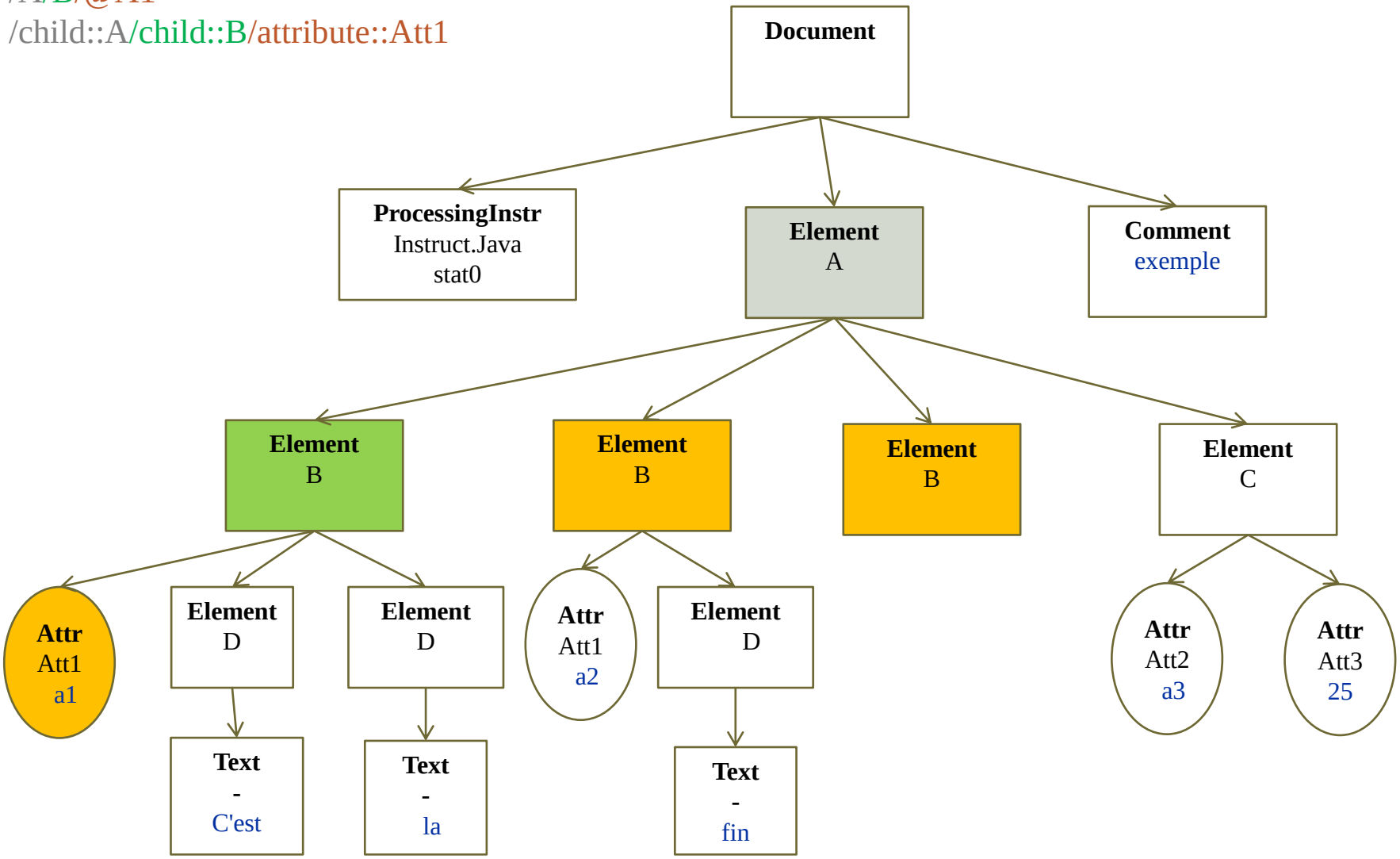
/child::A/child::B



Etape 3

/A/B/@A1

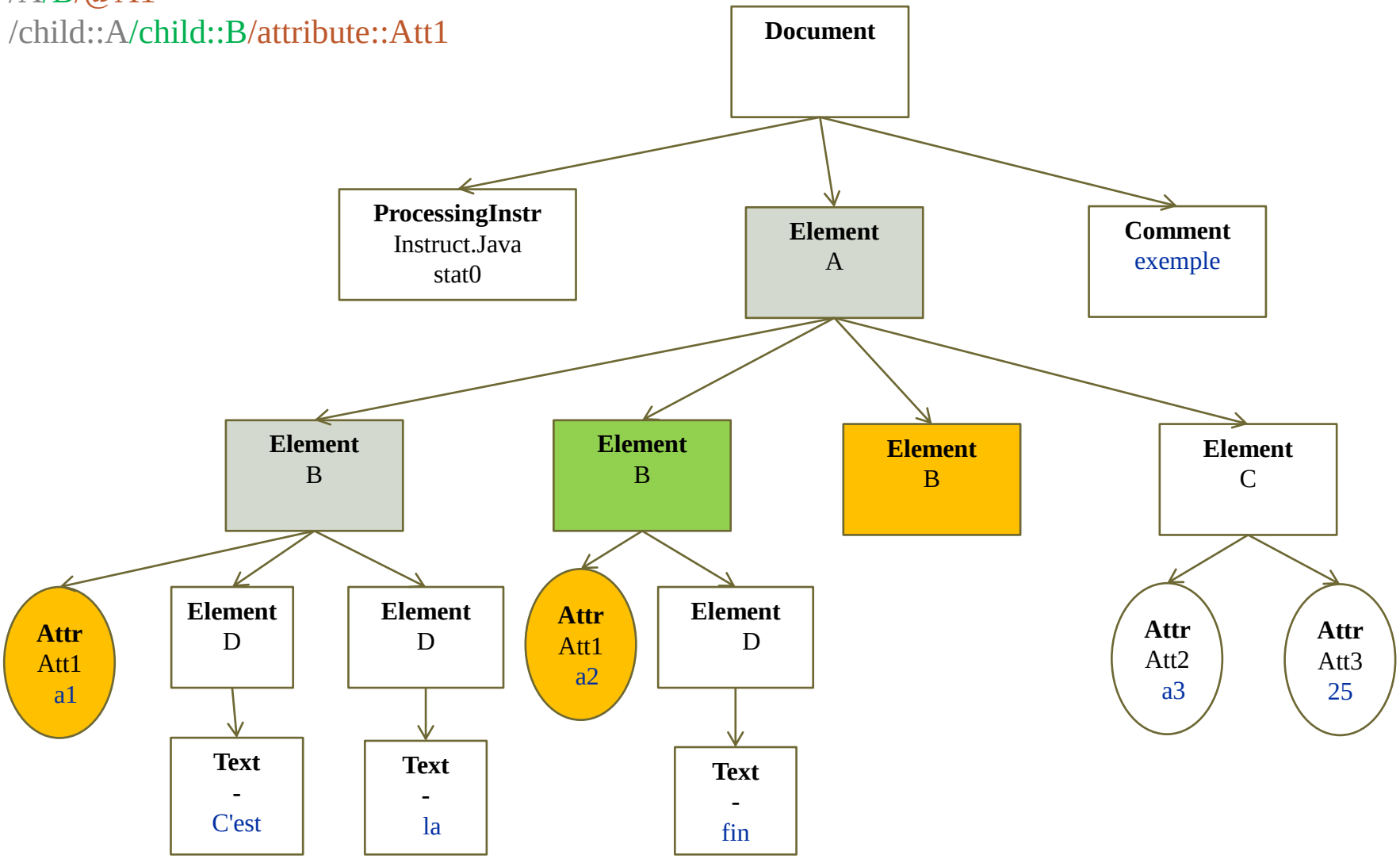
/child::A/child::B/attribute::Att1



Etape 3

/A/B/@A1

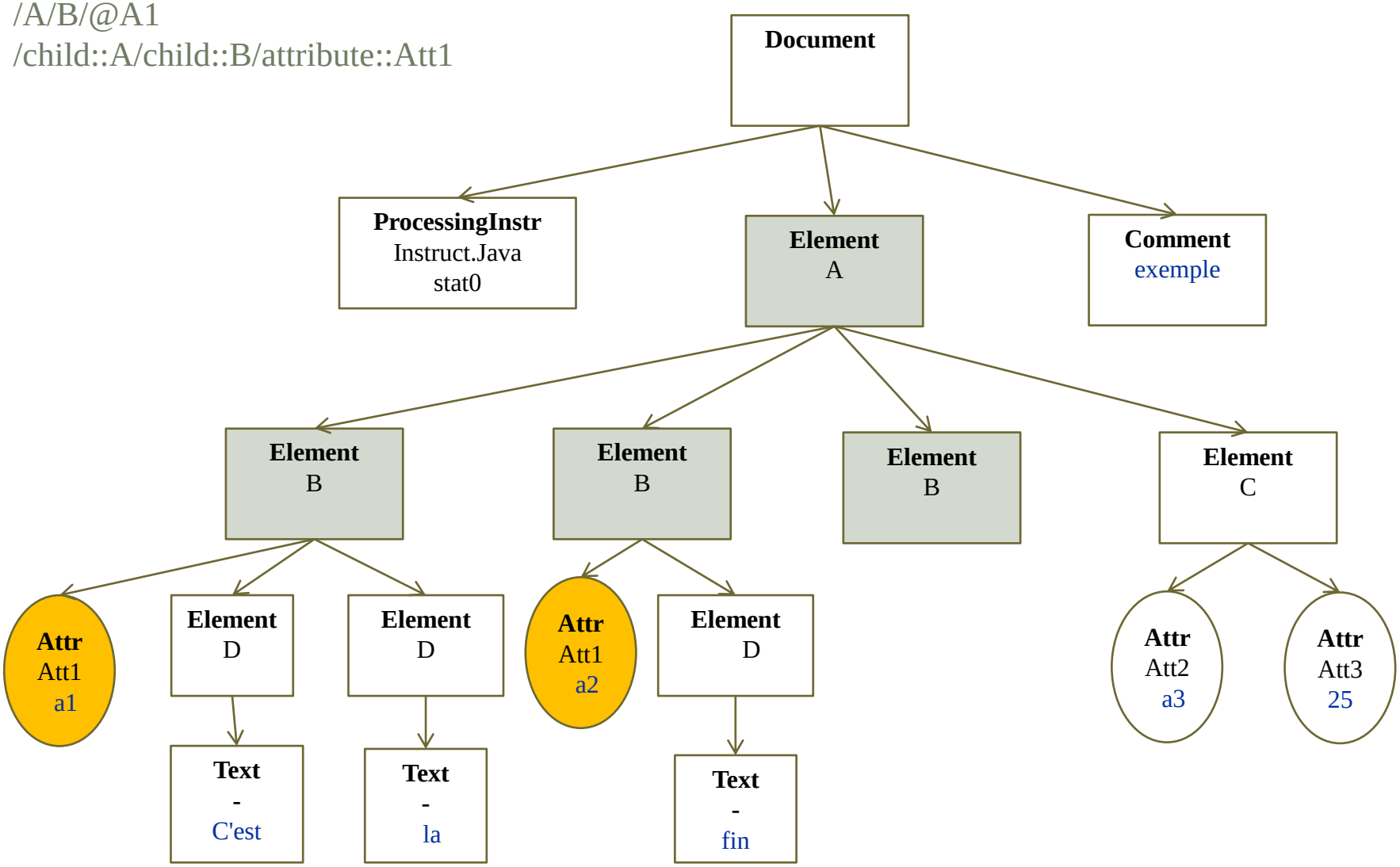
/child::A/child::B/attribute::Att1



Fin d'évaluation

/A/B/@A1

/child::A/child::B/attribute::Att1



XPath, langage de requêtes ?

XPath n'est pas un langage de requêtes : le contenu d'un élément (fragment correspondant au sous-arbre de l'élément) destination n'est pas extrait.

Cependant XPath comporte un système riche de fonctions et d'opérations : calcul sur le contenu d'un élément destination, p.ex valeur textuelle

XPath est utilisé pour faire des requêtes sur les documents dans des langages de requêtes (Xquery) ou de transformation de documents comme XSLT.