M1 MPRI : Automates et Applications

Cours 3
Logiques pour les arbres

kn@lri.fr

31 mars 2021

Plan

université PARIS-SACLAY

Dans l'épisode précédent

université PARIS-SACLAY

Nous avons introduit un nouveau modèle de calcul, les *automates d'arbres*. On se souvient que ceux-ci peuvent être :

- Déterministes ou non-déterministes (dans le cas non-det. il peut exister plusieurs run possibles pour un même arbre)
- Ascendants ou descendants (dans le cas ascendant, le run est calculé depuis les feuilles jusqu'à la racine, dans le cas descendant depuis la racine jusqu'aux feuilles)

De plus : $TD_{\text{det}} \subsetneq TD_{\text{nondet}} = BU_{\text{nondet}} = BU_{\text{det}}$

Quel langage de haut niveau pour les arbres?

Si la notion d'expression régulière de mot est standard, il existe de nombreux langages de description de langages d'arbres. Nous allons étudier deux approches :

1. Caractérisation par des formules logiques
2. Exemple de langages concrets (W3C)

Plan

Problèmes avec les automates

université PARIS-SACLAY

Les automates d'arbres (ascendants) permettent de décrire précisément les langages d'arbres, c'est à dire des ensembles d'arbre ayant une certaine propriété.

Leur défaut est d'être très « bas-niveau ». Si une propriété des nœuds de l'arbre doit être vérifiée à un niveau arbitraire il faut :

- dire explicitement comment aller depuis les feuilles jusqu'aux nœuds intéressant
- créer des états indiquant que ces nœuds sont intéressant
- continuer la reconnaissance jusqu'à la racine

Les arbres inintéressants, les états « dans lesquels on ne veut plus rien faire » sont alors explicitement donnés et « bouclent » (par exemple états puits).

Peut-on faire autrement

université PARIS-SACLAY

Si on prend l'exemple des peignes sur l'alphabet $\Sigma = \{f^2, a\}$ on aimerait pouvoir décrire ce langage comme :

« L'ensemble des arbres dont le fils gauche est un a et le fils droit a ou un arbre du langage »

sans se soucier de boucler, devoir refuser des nœuds, ...

Logique du premier ordre

université PARIS-SACLAY

Définition

Une formule logique du premier ordre est une production finie de la grammaire :

$$
\begin{array}{rcll}
\phi & ::= & U(x) & \textit{prédicat unaire} \\
 & | & B(x,y) & \textit{prédicat binaire} \\
 & | & \neg\phi & \textit{négation} \\
 & | & \phi \vee \phi & \textit{disjonction} \\
 & | & \exists x.\phi & \textit{quantification existentielle}
\end{array}
$$

Notations

université PARIS-SACLAY

Pour réduire la taille des formules, on pourra utiliser les notations suivante :

$$\begin{array}{lcll}
\phi_1 \wedge \phi_2 & \equiv & \neg(\neg\phi_1 \vee \neg\phi_2) & \text{conjonction} \\
\phi_1 \Rightarrow \phi_2 & \equiv & \phi_1 \vee \neg\phi_2 & \text{implication} \\
\phi_1 \Leftrightarrow \phi_2 & \equiv & (\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1) & \text{équivalence} \\
\phi_1 \oplus \phi_2 & \equiv & \neg(\phi_1 \Leftrightarrow \phi_2) & \text{« ou » exclusif} \\
\forall x\phi & \equiv & \neg\exists x.\neg\phi & \text{quantification universelle}
\end{array}$$

Prédicats

Étant donné un alphabet gradué Σ on se donne l'ensemble des prédicats suivants :

- $\mathrm{lab}_a(x)$, pour tout $a \in \Sigma$,
- $\mathrm{child}_i(x, y)$, pour tout $1 \leq i \leq \max\{n \in \mathbb{N} \mid n = |f|, \forall f \in \Sigma\}$
- $x \preceq y$
- $x = y$

Sémantique des prédicats

université PARIS-SACLAY

Définition

Soit Σ un alphabet gradué et $t \in \mathcal{T}(\Sigma)$. Soit $\pi, \pi_1, \pi_2 \in dom(t)$, La valeur de vérité des prédicats est donné par :

$$\begin{aligned} lab_a(\pi) &\equiv t(\pi) = a \\ child_i(\pi_1, \pi_2) &\equiv \pi_1 i = \pi_2 \\ \pi_1 \preceq \pi_2 &\equiv \pi_1 \leq_{lex} \pi_2 \end{aligned}$$

Le prédicat $x \preceq y$ est appelé *ordre du document* et signifie que x apparaît avant y lors d'un parcours préfixe de l'arbre.

Sémantique des formules

université PARIS-SACLAY

La sémantique des formules est donnée par le jugement $t, \gamma \vdash \phi$ indiquant que pour l'arbre t et la valuation γ, la formule ϕ est vraie. Une valuation est une fonction des variables libres de la formule dans $\mathrm{dom}(t)$.

$$
\begin{array}{lcl}
t, \gamma \vdash U(x) & \Leftrightarrow & U(\gamma(x)) \\
t, \gamma \vdash B(x, y) & \Leftrightarrow & B(\gamma(x), \gamma(y)) \\
t, \gamma \vdash \phi_1 \vee \phi_2 & \Leftrightarrow & t, \gamma \vdash \phi_1 \text{ ou bien } t, \gamma \vdash \phi_2 \\
t, \gamma \vdash \neg\phi & \Leftrightarrow & t, \gamma \nvdash \phi \\
t, \gamma \vdash \exists x.\phi & \Leftrightarrow & \exists \pi \in \mathrm{dom}(t),\ t, \gamma \cup \{x \mapsto \pi\} \vdash \phi
\end{array}
$$

Exemple

Si on reprend les arbres en forme de peignes sur $\Sigma = \{f^2, a\}$, on peut vérifier que pour tout arbre de ce langage la formule ci-dessous est vraie :

$$\forall x.(\mathsf{lab}_a(x) \ \vee \ (\mathsf{lab}_f(x) \ \wedge \ (\exists y.\mathsf{child}_1(x,y) \wedge \mathsf{lab}_a(y))))$$

On se rend compte qu'il n'y a pas ici de « récursion ». On définit une propriété « locale » sur un nœud et on dit qu'elle est vraie pour tous les nœuds de l'arbre.

Définition d'un langage

université PARIS-SACLAY

Définition

On peut définir un langage L_ϕ par :

$$L_\phi = \{t \mid \exists \gamma \text{ tel que } t, \gamma \vdash \phi\}$$

Seulement des langages ?

université PARIS-SACLAY

La spécification logique d'un langage permet de reformuler certains problèmes :

- Une formule close (comme l'exemple) défini un langage.
- Une formule avec une variable libre x définit *une requête*

En effet, étant donné un arbre t :

$$\{\pi \in \mathsf{dom}(t) \mid t, \{x \mapsto \pi\} \vdash \phi\}$$

représente l'ensemble des nœuds de l'arbre tels que la formule est vraie. On peut généraliser à des formules contenant un nombre arbitraire de variables libres.

Clôtures booléennes

Le formalisme rend trivial les propriétés de clôtures :

$$
\begin{aligned}
L_{\phi_1} \cap L_{\phi_2} &\Leftrightarrow L_{\phi_1 \wedge \phi_2} \\
L_{\phi_1} \cup L_{\phi_2} &\Leftrightarrow L_{\phi_1 \vee \phi_2} \\
\bar{L_\phi} &\Leftrightarrow L_{\neg \phi}
\end{aligned}
$$

Problèmes de décisions

université PARIS-SACLAY

- Étant donné un arbre, est-ce que la formule ϕ est vraie $\Rightarrow$ polynomial
- Le langage d'une formule est-il vide $\equiv$ la formule est-elle satisfiable ?

  $\Rightarrow$ non élémentaire...

$$\left.2^{2^{\cdots^{2^c}}}\right\} |\phi|$$

Ce langage est trop puissant...

...et pourtant

La logique du premier ordre sur les arbres ne capture *pas* l'ensemble des langages d'arbres reconnaissables !

Exemple : L'ensemble des arbres sur $\Sigma = \{f^2, a, b\}$ avec un nombre pair de a. On se donne l'automate $\mathcal{A} = (\{q_0, q_1\}, \Sigma, \delta, \{q_0\})$:

$$\delta : \begin{array}{rcl@{\qquad}rcl} a & \mapsto & \{q_1\} & f(q_0, q_0) & \mapsto & \{q_0\} \\ b & \mapsto & \{q_0\} & f(q_1, q_0) & \mapsto & \{q_1\} \\ f(q_1, q_1) & \mapsto & \{q_0\} & f(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$

q_0 représente les nombres pairs de a et q_1 les nombres impairs (note : ce langage n'est pas TD déterministe).

FO sur les arbres ?

On a donc un langage :

- compact : on peut exprimer des propriétés vraies sur tous les nœuds ou sur certains *sans* avoir recours à la récursion
- certaines propriété de compatage (représentables par des automates) ne sont pas exprimables
- a une complexité catastrophique pour le problème de satisfiabilité

©XKCD

université PARIS-SACLAY

Définition (Monadic Second-order Logic)

Une formule de MSO est une production finie de la grammaire :

$$\begin{array}{rcll} \phi & ::= & \ldots & \textit{formules de FO} \\ & | & x \in X & \textit{appartenance à un ensemble} \\ & | & \exists X.\phi & \textit{quantification du second ordre} \end{array}$$

X dénote des *ensemble de nœuds*. On a évidemment que $\forall X.\phi$ est une notation pour $\neg\exists X.\neg\phi$.

Sémantique de MSO

université PARIS-SACLAY

On exprime naturellement la sémantique de MSO par un jugement : La sémantique des formules est donnée par le jugement $t, \gamma, \Gamma \vdash \phi$ indiquant que pour l'arbre t et les valuation γ et Γ, la formule ϕ est vraie. Une Γ est une fonction des variables s'ensembles libres de la formule dans $\mathcal{P}_f(()\mathrm{dom}(t))$.

$$
\begin{aligned}
t, \gamma, \Gamma \vdash x \in X &\quad \Leftrightarrow \quad \gamma(x) \in \Gamma(X) \\
t, \gamma \vdash \exists X.\phi &\quad \Leftrightarrow \quad \exists P \subseteq \mathrm{dom}(t),\ t, \gamma, \Gamma \cup \{X \mapsto P\} \vdash \phi
\end{aligned}
$$

Que peut-on dire de plus ?

université PARIS-SACLAY

On peut caractériser les descendants d'un nœud. Les descendants de x sont les nœuds de l'ensemble Y tel que :

$$\forall y \in Y . y \in Y \Leftrightarrow x = y \vee \exists z . (z \in Y \wedge (\mathrm{child}_1(z, y) \vee \ldots \vee \mathrm{child}_k(z, y)))$$

On suppose un alphabet d'arité maximale k.

Que peut-on dire de plus ?

université PARIS-SACLAY

On peut caractériser les arbres ayant un nombre pair de a (sur $\Sigma = \{f^2, a, b\}$) :

Que peut-on dire de plus ?

université PARIS-SACLAY

On peut caractériser les arbres ayant un nombre pair de a (sur $\Sigma = \{f^2, a, b\}$) :

$$\begin{aligned}
&\exists P.\exists I.\forall x.(x \in P \oplus x \in I) \\
\wedge\ & \mathrm{lab}_a(x) \Rightarrow x \in I \\
\wedge\ & \mathrm{lab}_b(x) \Rightarrow x \in P \\
\wedge\ & \mathrm{lab}_f(x) \Rightarrow (\exists y.\exists z.\mathrm{child}_1(x,y) \wedge \mathrm{child}_2(x,z) \\
& ((y \in P) \wedge (z \in P)) \vee ((y \in I) \wedge (z \in I)))
\end{aligned}$$

MSO vs Automates

université PARIS-SACLAY

Théorème (Thatcher, Wright 68)

Les langages exprimables par une formule MSO sont exactement les langages reconnaissables par un automate d'arbre (ascendant).

Preuve (idée) :

- on met la formule dans une certaine forme canonique
- on donne des automates pour les formules de base ($x \in X$, $\mathrm{lab}_a(x)$, ...)
- on construit inductivement un automate pour les connecteurs logiques (les quantificateurs et les négations impliquent une explosion)

MSO vs Automates Preuve

université PARIS-SACLAY

On peut aussi construire une formule à partir d'un automate :

- on associe une sous-formule pour chaque transition
- elimination des états (comme pour les regexps)

MSO ?

On a donc un langage :

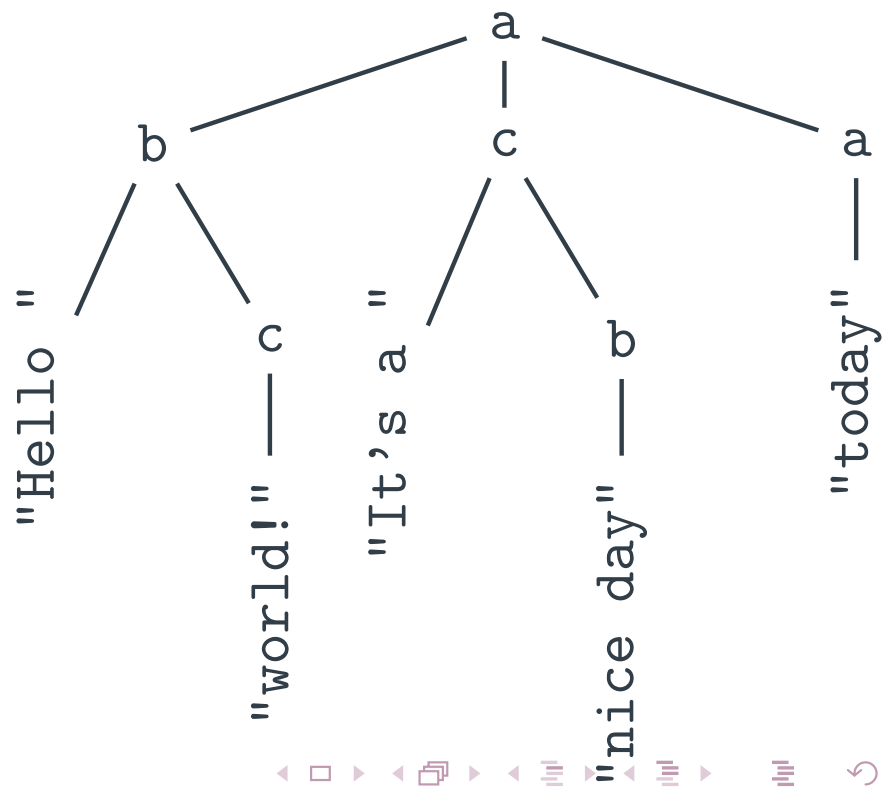
- compact : on peut exprimer des propriétés vraies sur tous les nœuds ou sur certains *sans* avoir recours à la récursion
- exactement équivalent aux automates d'arbres
- a une complexité catastrophique pour le problème de satisfiabilité (ça ne change pas)

Plan

Le format XML pour les documents

Le standard XML du W3C définit un format de document pour le stockage et l'échange de données. Essentiellement il s'agit des documents composés de *balises* `<ouvrantes>` ou `</fermantes>`, bien parenthésées. Exemple :

```
<a>
  <b>Hello <c>world!</c></b>
  <c>It's a <b>nice day</b></c>
  <a>today</a>
</a>
```



Le format XML pour les documents (2)

université PARIS-SACLAY

- on ignore de nombreux autre aspects du standard
- on suppose que tous les textes sont remplacés par un nœud \$.
- on va aussi considérer que le format HTML rentre dans ce cadre

Arbres d'arité non bornée

université PARIS-SACLAY

On est en présence d'arbres d'arité non bornée :

- un nœud peut avoir un nombre arbitraire de fils
- le nom des étiquettes ne determine pas le nombre de fils

Mais on veut tout de même pouvoir utiliser MSO ou nos automates !

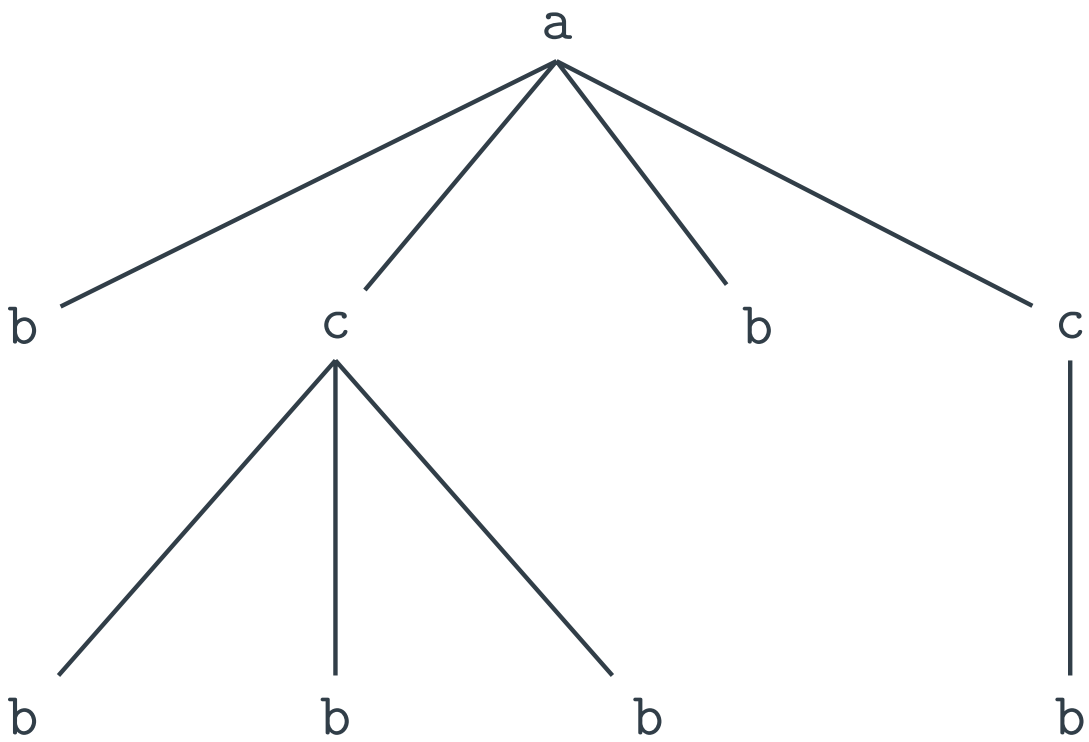
Arbres n-aires vers arbres binaires

université PARIS-SACLAY

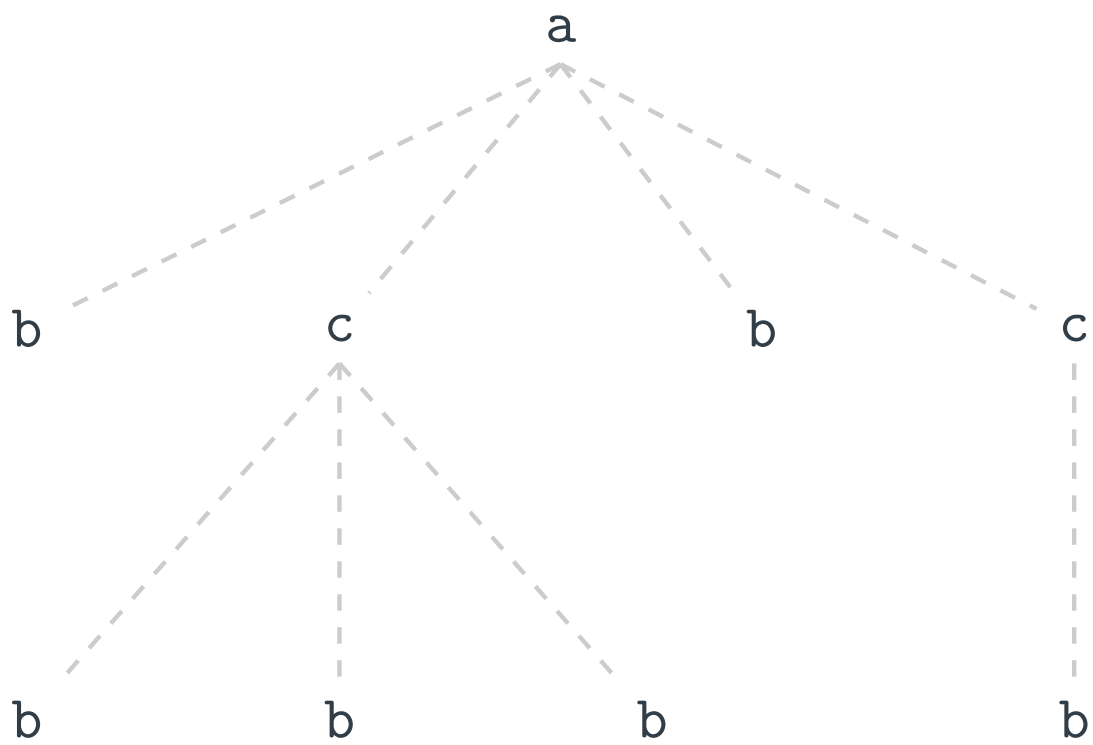
On utilise une astuce permettant d'encoder n'importe quel arbre n-aire dans un arbre binaire :

- Le premier fils d'un nœud (dans l'arbre n-aire) est le fils gauche
- Le frère à droite d'un nœud (dans l'arbre n-aire) est le fils droit

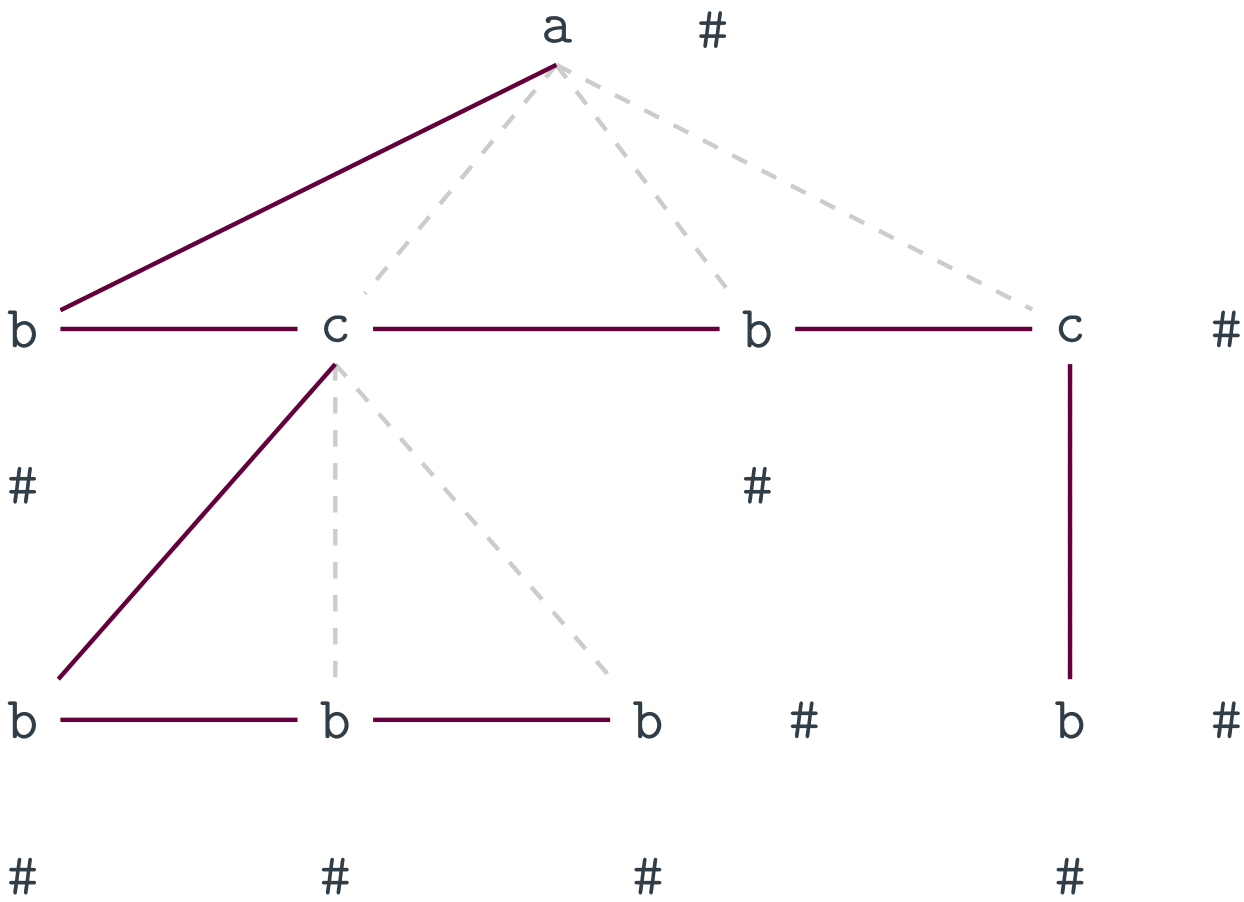
Exemple



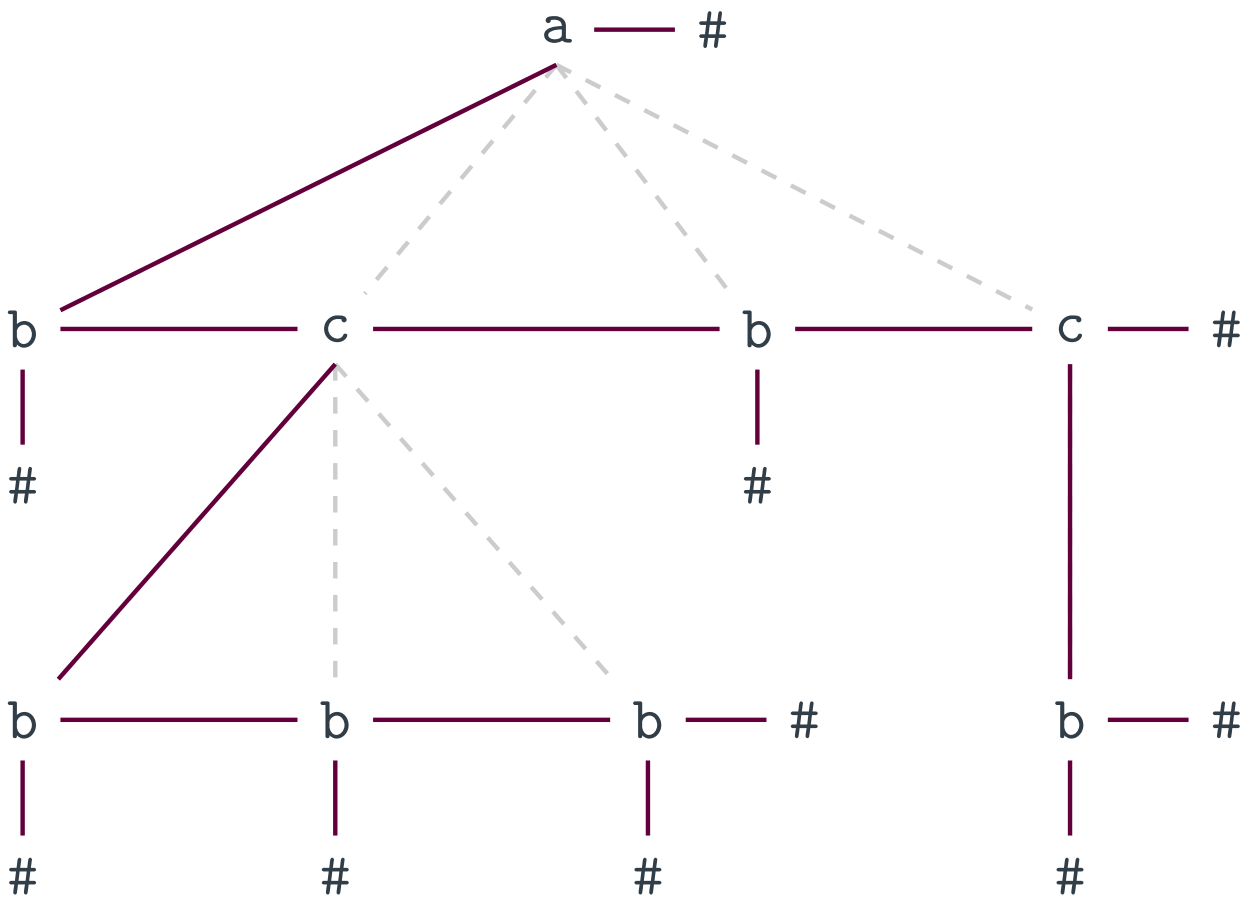
Exemple



Exemple



Exemple



Structure de données

université PARIS-SACLAY

Ce codage est identique à celui d'une structure de données classiques pour les arbres :

- Un nœud stocke une étiquette et un pointeur vers une liste chaînée de fils
- Le premier élément de la liste est accessible directement, pour les autres il faut parcourir la liste
- le pointeur `null` correspond au nœuds fictifs #

Structure de données (2)

université PARIS-SACLAY

```
struct node;
struct list {
    struct list *next; //arc vers la droite
    struct node *node;
};
struct node {
    char * label;
    struct list *children; //arc vers le bas
};
```

DTD

Les *document type definitions* (DTD) permettent de définir des schémas de documents c'est à dire des langages d'arbres :

```
<!ELEMENT a  ( (b|c)* ) >
<!ELEMENT b  ( EMPTY ) >
<!ELEMENT c  ( (b|c)* ) >
```

Pour chaque nom de balise on donne son contenu au moyen d'une expression régulière sur les fils. Une balise détermine le contenu de façon unique (on ne peut pas avoir deux définitions pour la même balise). $\Rightarrow$ TD déterministe

DTD (2)

Le fait que le langage soit TD déterministe implique qu'on peut le valider en *streaming* c'est à dire, en ne gardant qu'un espace mémoire proportionnel à la hauteur du document.

Pourquoi est-ce bien ?

DTD (2)

Le fait que le langage soit TD déterministe implique qu'on peut le valider en *streaming* c'est à dire, en ne gardant qu'un espace mémoire proportionnel à la hauteur du document.

Pourquoi est-ce bien ?

On peut valider le document pendant le chargement du fichier.

Quelles limitations ?

DTD (2)

Le fait que le langage soit TD déterministe implique qu'on peut le valider en *streaming* c'est à dire, en ne gardant qu'un espace mémoire proportionnel à la hauteur du document.

Pourquoi est-ce bien ?

On peut valider le document pendant le chargement du fichier.

Quelles limitations ?

On ne peut pas accepter des documents ayant une dépendence entre les fils (ou tout autre condition qui n'est pas top-down det).

XPath

université PARIS-SACLAY

Le langage XPath est un langage de requête (sélection de nœuds). Dans sa version la plus simple (navigational Core XPath), les requêtes sont de la forme :

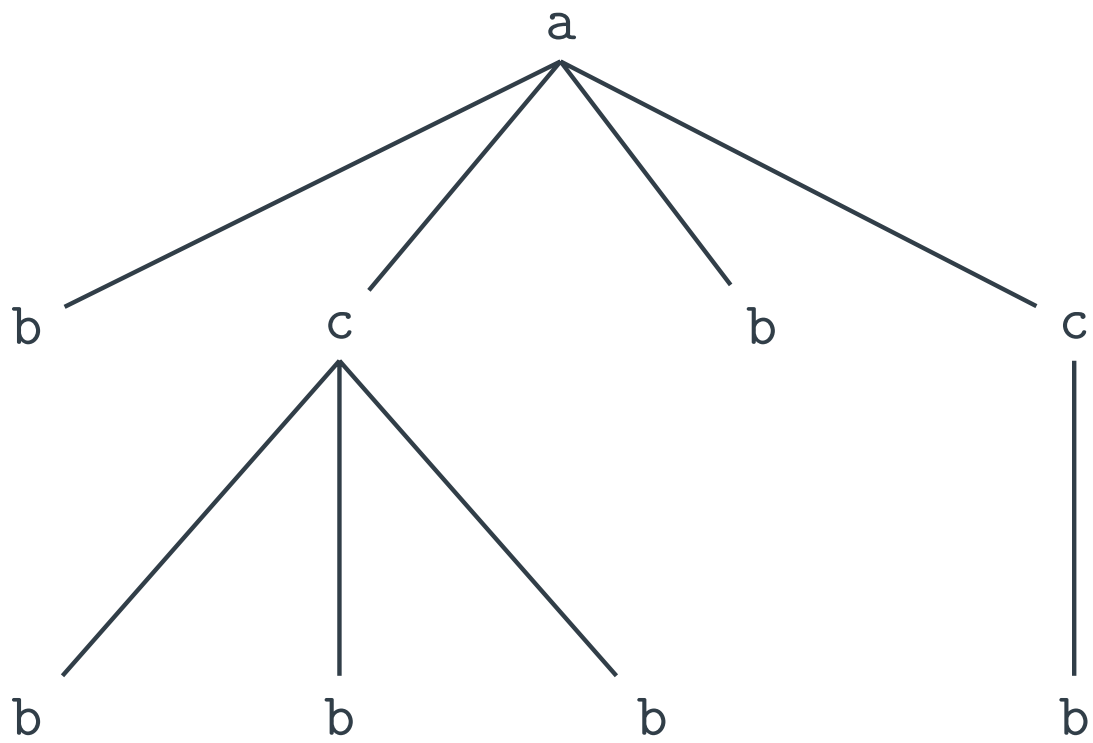
$$\mathit{chemin} ::= \mathit{axe}_1 :: \mathit{test}_1 [\ \mathit{pred}_1] \ / \ldots / \mathit{axe}_n :: \mathit{test}_n [\ \mathit{pred}_n]$$

$$\begin{array}{lcl} \mathit{axe} & ::= & \texttt{child} \mid \texttt{descendant} \mid \texttt{parent} \mid \texttt{ancestor} \mid \ldots \\ \mathit{test} & ::= & \mathit{nom\ de\ balise} \mid * \\ \mathit{pred} & ::= & \mathit{chemin} \mid \mathit{pred}_1 \ \texttt{or} \ \mathit{pred}_2 \mid \mathit{pred}_1 \ \texttt{and} \ \mathit{pred}_2 \mid \texttt{not}(\mathit{pred}_1) \end{array}$$

Exemple

université PARIS-SACLAY

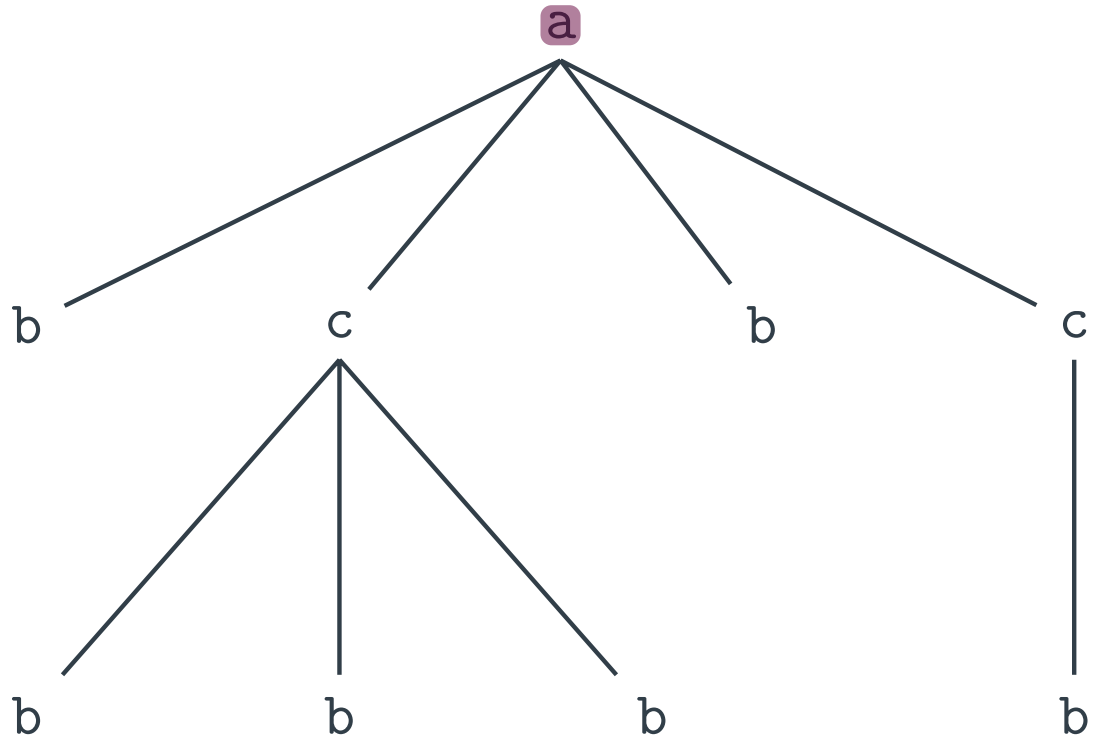
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

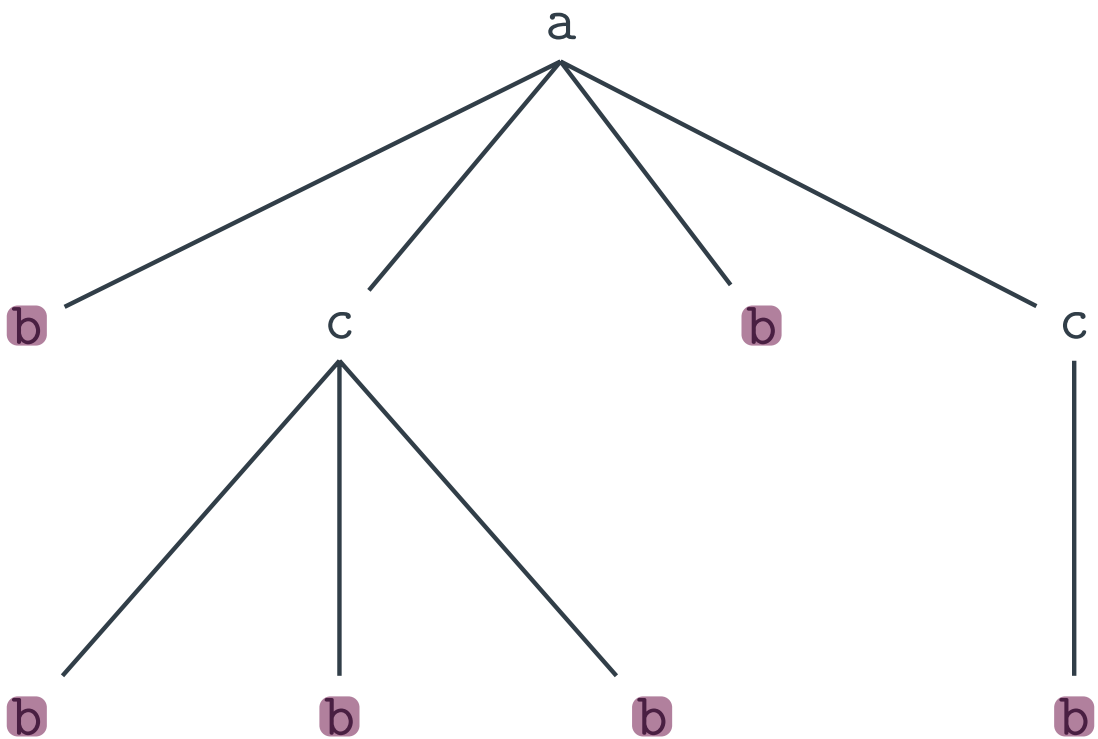
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

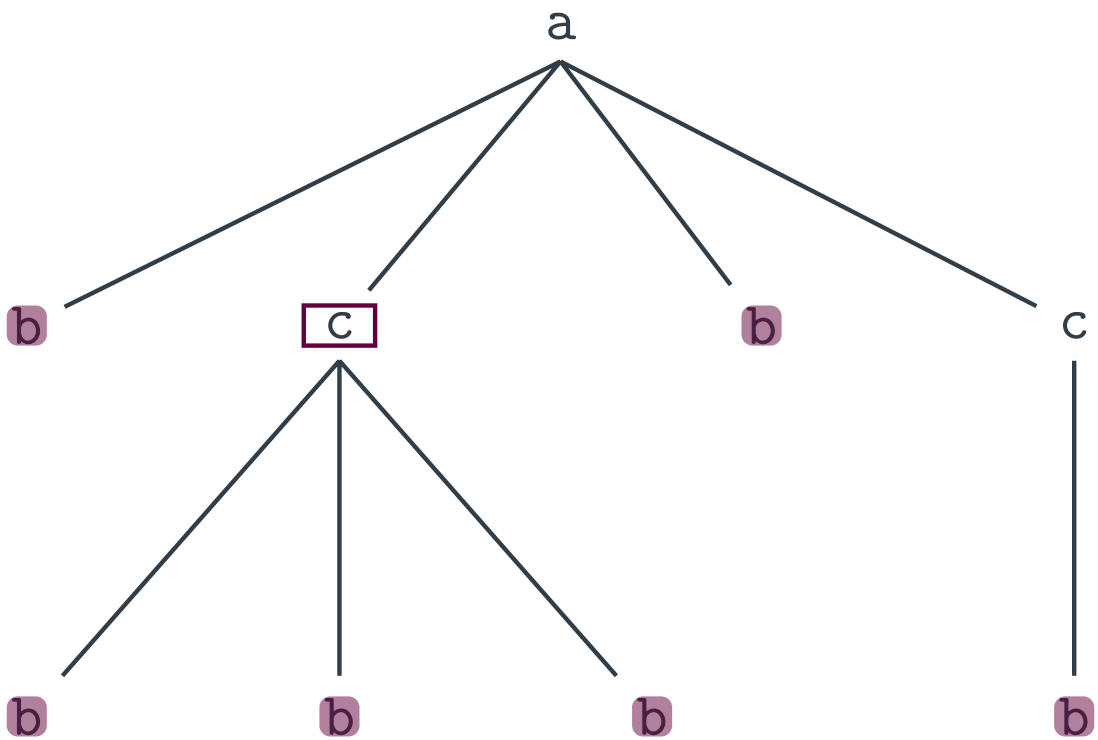
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

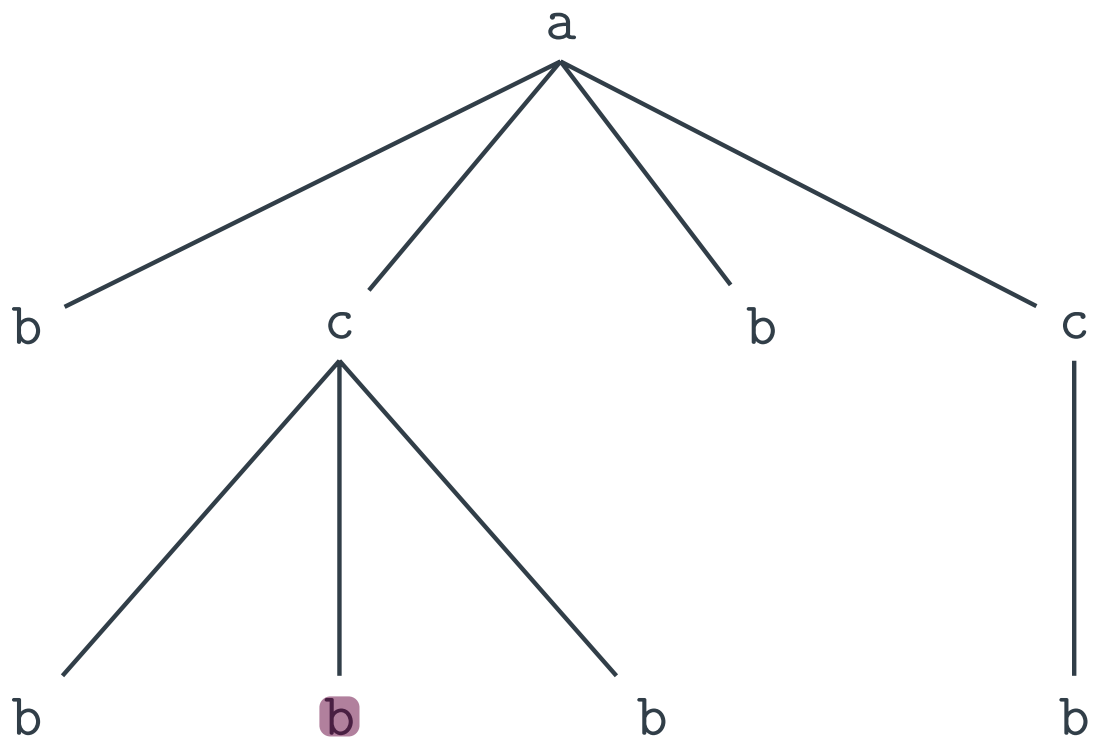
/descendant::b[parent::c]/parent::*/child::*



Exemple

université PARIS-SACLAY

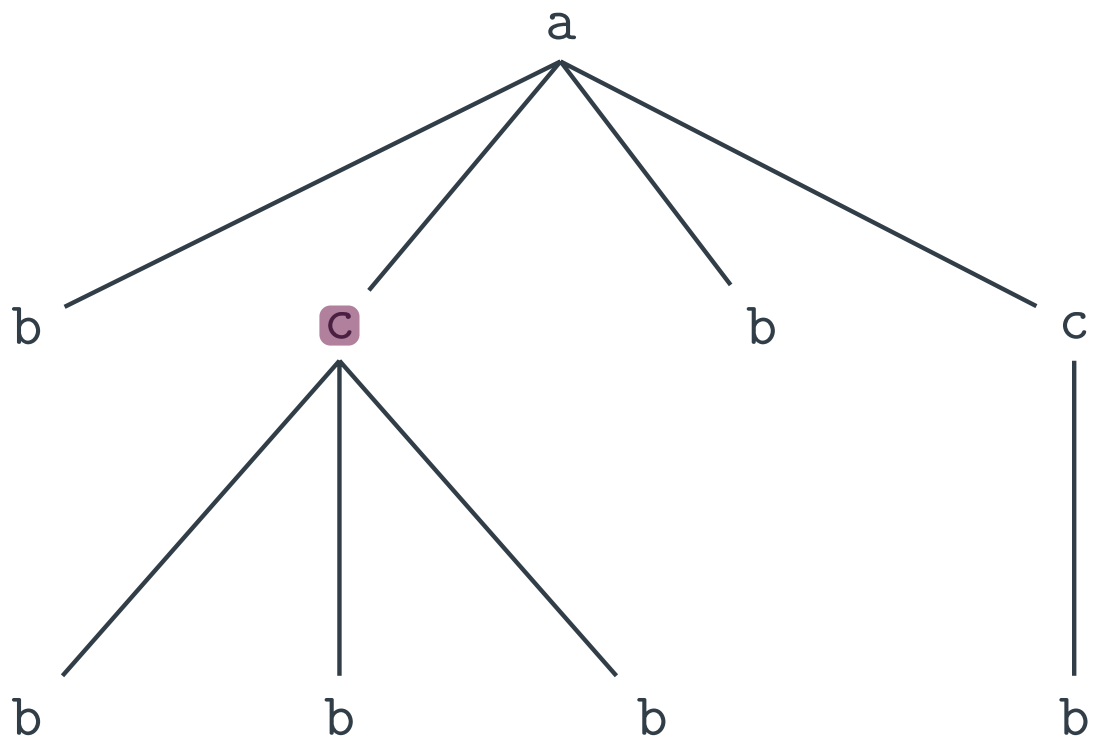
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

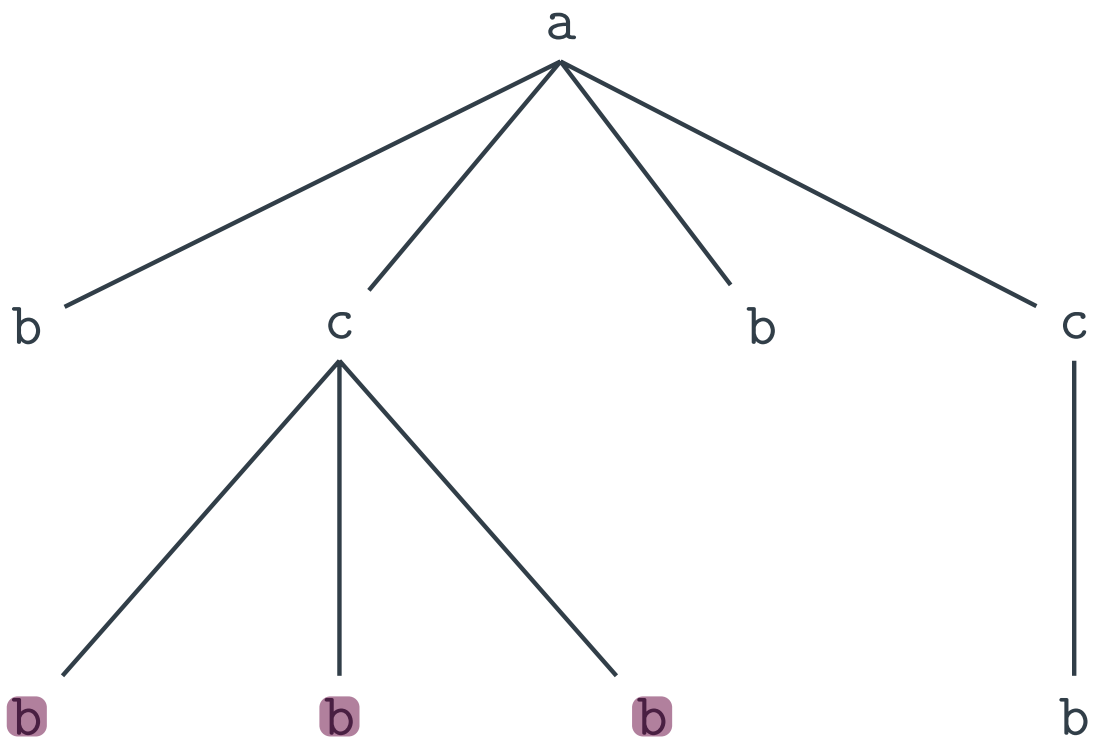
/descendant::b[parent::c]/parent::*/child::*



Exemple

université PARIS-SACLAY

/descendant::b[parent::c]/parent::*/child::*



Expressivité de XPath ?

université PARIS-SACLAY

Le fragment présenté est moins expressif que FO. Par exemple, on ne peut pas exprimer : Renvoyer tous les nœuds `a` descendants d'un `b` tel qu'il n'y a pas de `c` entre le `a` et le `b`.

Quels problèmes souhaite-t-on résoudre ?

université PARIS-SACLAY

1. Étant donné un document D, est-il valide par rapport à une DTD T ?

Quels problèmes souhaite-t-on résoudre ?

université PARIS-SACLAY

1. Étant donné un document D, est-il valide par rapport à une DTD T ? $O(|D| + |T|)$
2. Étant donné un document D, quels sont les nœuds renvoyés par une requête XPath Q ?

Quels problèmes souhaite-t-on résoudre ?

université PARIS-SACLAY

1. Étant donné un document D, est-il valide par rapport à une DTD T ? $O(|D| + |T|)$
2. Étant donné un document D, quels sont les nœuds renvoyés par une requête XPath Q ? $O(|D| \times |Q|)$
3. Étant donné une DTD T et une requête XPath Q, la requête est-elle satisfiable ?

Quels problèmes souhaite-t-on résoudre ?

université PARIS-SACLAY

1. Étant donné un document D, est-il valide par rapport à une DTD T ? $O(|D| + |T|)$
2. Étant donné un document D, quels sont les nœuds renvoyés par une requête XPath Q ? $O(|D| \times |Q|)$
3. Étant donné une DTD T et une requête XPath Q, la requête est-elle satisfiable ?EXPTIME

Proposer un algorithme pour (2) basé sur des automates sera le sujet du devoir.