M1 MPRI : Automates et Applications

Lecture 3
Logic for trees

kn@lri.fr

January 23, 2023

Outline

Previously

université PARIS-SACLAY

We have introduced a new computation model, *tree automata*.

Recall that these can be:

- Deterministic or non-deterministic, as in the word case
- Bottom-up or top-down

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

What is a high-level language for trees?

université PARIS-SACLAY

The notion of regular expression for words is standard and pervasive: operating system shells, programming languages, languages theory. For trees, the situation is a bit different. We study two formalisms:

1. Defining languages as sets that verify a property given in a particular logic
2. Concrete languages from the W3C

Outline

Problems with automata

université PARIS-SACLAY

By definition, regular tree languages are those that can be recognized by a BU TA. We have also seen on examples how to encode some properties as structural properties on trees (e.g. the set of true statements of the prop. logic).

The problem is that TA are very low-level. If a property must hold at for arbitrarily deep paths, one must:

- describe explicitely how to go from the leaves to such interesting nodes
- change state to indicate that we have seen such an interesting node
- continue up to the root

For uninteresting subtrees, must be ignored explicitely.

Can we do it differently?

université PARIS-SACLAY

Recal the example of right combs on $\Sigma = \{f^2, a\}$. We would like to write:

> "the set of trees which are either a or f-labeled nodes which have an a left subtree and a tree of the set as a right subtree"

without bothering with loops, refusing nodes (sink states).

First-order logic

université PARIS-SACLAY

Definition

A first-order logical formula is a finite production of the following grammar:

$$
\begin{array}{rcll}
\phi & ::= & U(x) & \textit{unary predicate} \\
 & | & B(x,y) & \textit{binary predicate} \\
 & | & \neg\phi & \textit{negation} \\
 & | & \phi \vee \phi & \textit{disjunction} \\
 & | & \exists x.\phi & \textit{existential quantification}
\end{array}
$$

Notations

université PARIS-SACLAY

We can use some syntactic sugar to define well known operations:

$\phi_1 \wedge \phi_2$	$\equiv$	$\neg(\neg\phi_1 \vee \neg\phi_2)$	conjunction
$\phi_1 \Rightarrow \phi_2$	$\equiv$	$\phi_1 \vee \neg\phi_2$	implication
$\phi_1 \Leftrightarrow \phi_2$	$\equiv$	$(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$	equivalence
$\phi_1 \oplus \phi_2$	$\equiv$	$\neg(\phi_1 \Leftrightarrow \phi_2)$	X-or
$\forall x\phi$	$\equiv$	$\neg\exists x.\neg\phi$	universal quantification

Predicates

université PARIS-SACLAY

For a *fixed* ranked alphabet Σ, we define the following predicates:

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

Semantics of predicates

université PARIS-SACLAY

Definition

Let Σ be a ranked and $t \in \mathcal{T}(\Sigma)$. Let $\pi, \pi_1, \pi_2 \in dom(t)$, The truth value of predicates is defined as:

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

Predicate $x \preceq y$ is called *document order* and it means that x is visited before y during a depth-first, left-to-right traversal of the tree.

Henceforth, when we say first-order logic, we mean first-order logic with this fixed set of predicates (for a given Σ).

Semantics of formulæ

université PARIS-SACLAY

The semantics of formulæ is given by the judgement $t, \gamma \vdash \phi$ which means that for a tree t and a valuation γ, the formula ϕ is true. A valuation is a mapping from free variables of the formula to $\mathrm{dom}(t)$.

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

Example

Let's revisit the right combs on $\Sigma = \{f^2, a\}$, we can verify that for all such tree, the following formula holds:

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

Importantly, there is no explicit recursion. We define a local property, and we state that this property holds for all nodes.

Logical characterization of languages

université PARIS-SACLAY

Definition

We can define the language L_ϕ of a formula ϕ as:

$$L_\phi = \{t \mid \exists \gamma \text{ such that } t, \gamma \vdash \phi\}$$

Are we limited to languages?

université PARIS-SACLAY

Using logical formulæ allows us to rephrase some problems:

- A closed formula defines a language (as in the example)
- A formula with a free variable x correspond to a *predicate* or a *query*

Indeed, given a tree t :

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

represents the set of paths in the tree for wich the formula holds. This can be generalized to an arbitrary number of variables. That allows us to *select* paths of interest in the tree.

Boolean closures

université PARIS-SACLAY

These are just the consequence of the semantics:

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

Decision problems

université PARIS-SACLAY

- For a fixed tree, is a formula ϕ true ? Polynomial.
- The language of a formula is empty $\equiv$ the formula is satisfiable?
  $\Rightarrow$ non-elementary...

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

In some ways this language is too powerful...

. . . but surprisingly

université PARIS-SACLAY

First order logic cannot express all regular tree languages!

Ex: The set of trees on $\Sigma = \{f^2, a, b\}$ with an even number of a.

Easily recognized by $\mathcal{A} = (\{q_0, q_1\}, \Sigma, \delta, \{q_0\})$:

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

q_0 represent a subtree with even number of a et q_1 odd numbers of a (note: this language is not TD deterministic).

FO on trees?

To summarize:

- compact: one can express properties on path without explicit recursion
- some properties cannot be expressed (FO cannot count, even modulo two)
- crazy complexity

©XKCD

université PARIS-SACLAY

Definition (Monadic Second-order Logic)

A formula of the MSO is a finite production of the following grammar:

$$\begin{array}{rcll} \phi & ::= & \ldots & \textit{formula of FO} \\ & | & x \in X & \textit{set membershiip} \\ & | & \exists X.\phi & \textit{second-order quantification} \end{array}$$

X denotes *sets of paths*. We have the syntactic sugar:

$\forall X.\phi \equiv \neg\exists X.\neg\phi$.

Semantics of MSO

We naturally express the semantics of MSO with a judgement

$$t, \gamma, \Gamma \vdash \phi$$

which means that given t, γ (from free variables to paths) and Γ (from free Variables to sets of paths), formula ϕ is true.

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

What can we express?

université PARIS-SACLAY

We can characterize the *descendants* of a path. The descendants of a path x is the set of paths Y such that:

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

where k is the (fixed) maximal arity of a symol in Σ.

What can we express?

université PARIS-SACLAY

We can characterize the set of trees with an even number of a (over $\Sigma = \{f^2, a, b\}$):

What can we express?

université PARIS-SACLAY

We can characterize the set of trees with an even number of a (over $\Sigma = \{f^2, a, b\}$):

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

MSO vs Automata

université PARIS-SACLAY

Theorem (Thatcher, Wright 68)

The set of regular tree languages is exactly the set of MSO-definable languages

Proof (very rough sketch):

- put the formula in a particular canonical form
- give automata for each basic formula ($x \in X$, $\mathrm{lab}_a(x)$, ...)
- build inductively automata for logical connective by connecting sub-automata (like for Thompson and regex, but there is a massive explosion due to quantification).

MSO vs Automates

université PARIS-SACLAY

We can also go backward:

- label each transition of an automaton with a formula, initialy only the formula $\mathrm{lab}_a(ax)$
- use an algorithm similar to state elimination (merge transitions then states and repeat)

MSO ?

université PARIS-SACLAY

On a donc un langage :

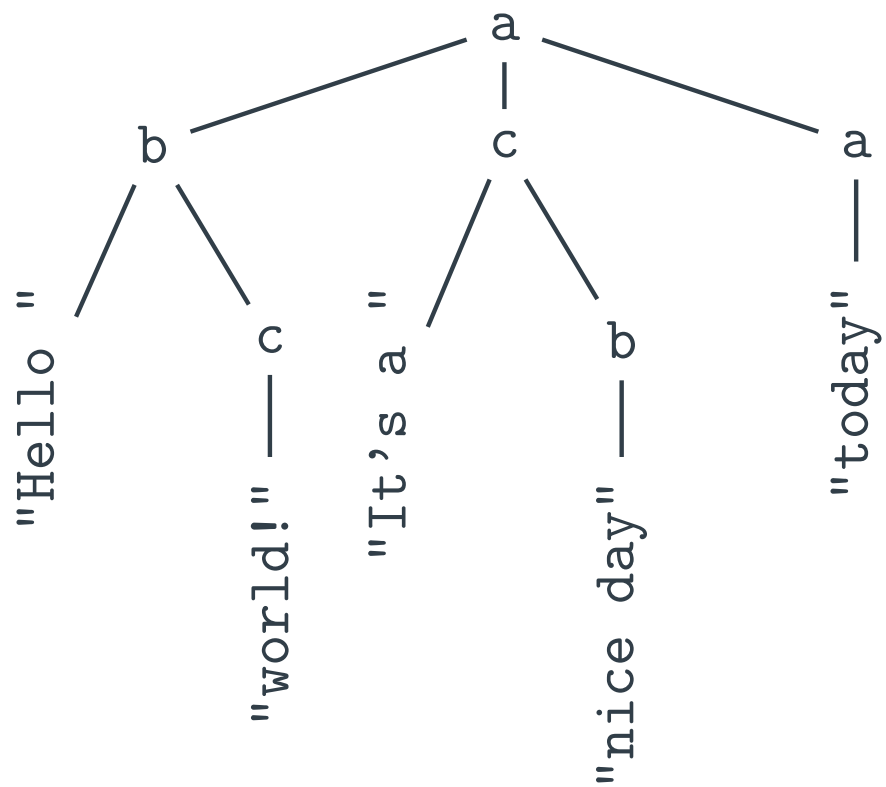
- compact: can express complex properties without explicit recursion
- exacly equivalent to regular tree languages
- still, non-elementary complexity

Outline

XML, a standard for documents

XML is a W3C standard that defines a format for storing and exchanging tree structured data. Essentially, it defines a *markup* with `<opening>` tags or `</closing>`, that must be balanced:

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



XML, a standard for documents (2)

université PARIS-SACLAY

For our overview:

- we ignore many intricacies of the standard (character encodings, attributes, namespaces, . . .)
- we assume all text path are replaced by leaves \$.
- we will also assumes that everything applies to HTML

Unranked trees

université PARIS-SACLAY

Documents represent *unranked* trees

- a path may have a finite, arbitrary number of children
- the labels do not indicate the number of children

Still, we would like to make use of automata or MSO

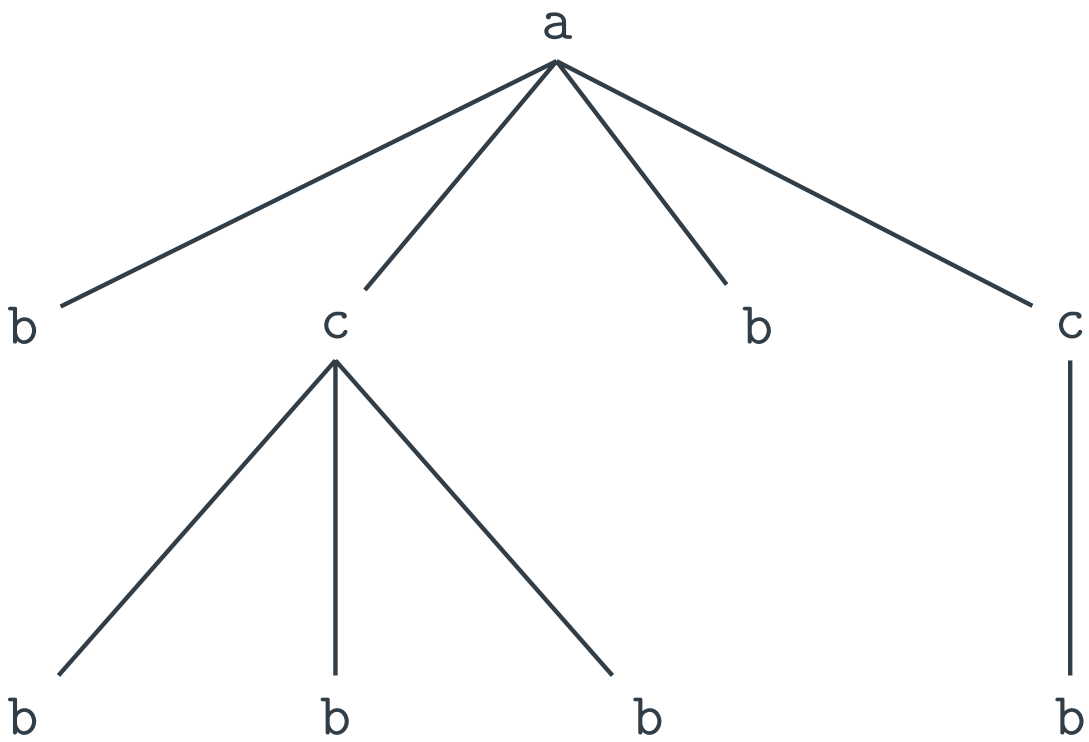
n-ary trees to binary trees

université PARIS-SACLAY

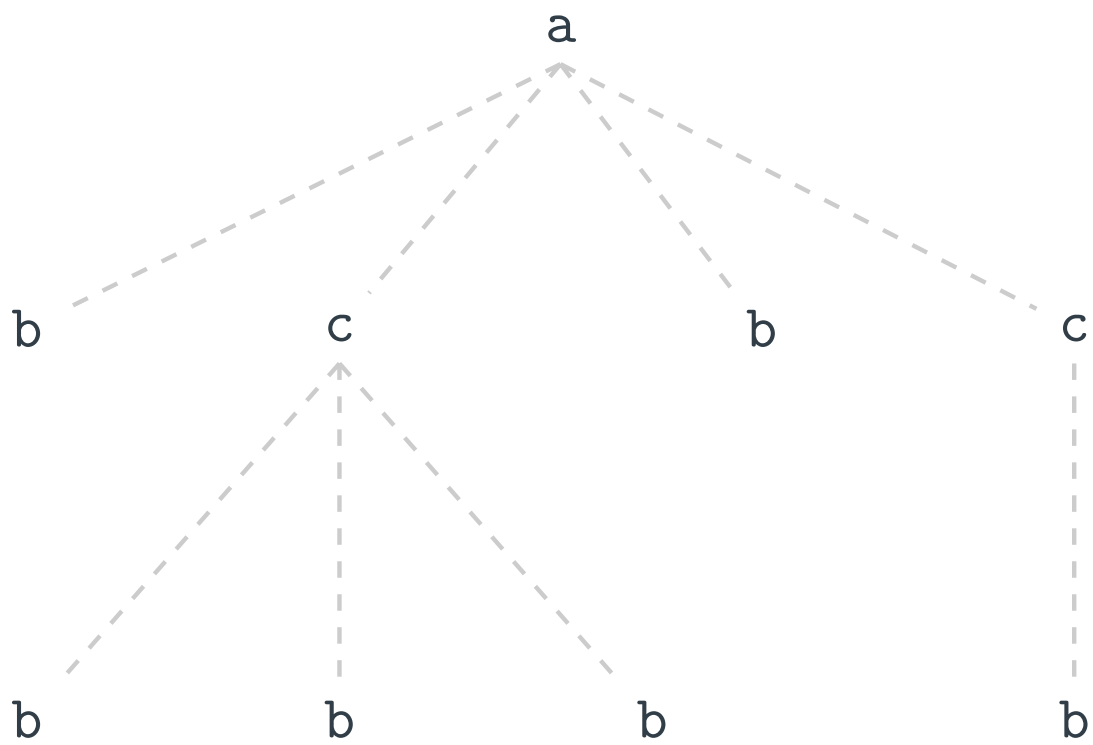
We use a trick to encode any unranked tree into a binary tree:

- the first child of a path (in the n-ary tree) is the first child in the binary tree
- the right sibling of a path (in the n-ary tree)is the second child in the binary tree

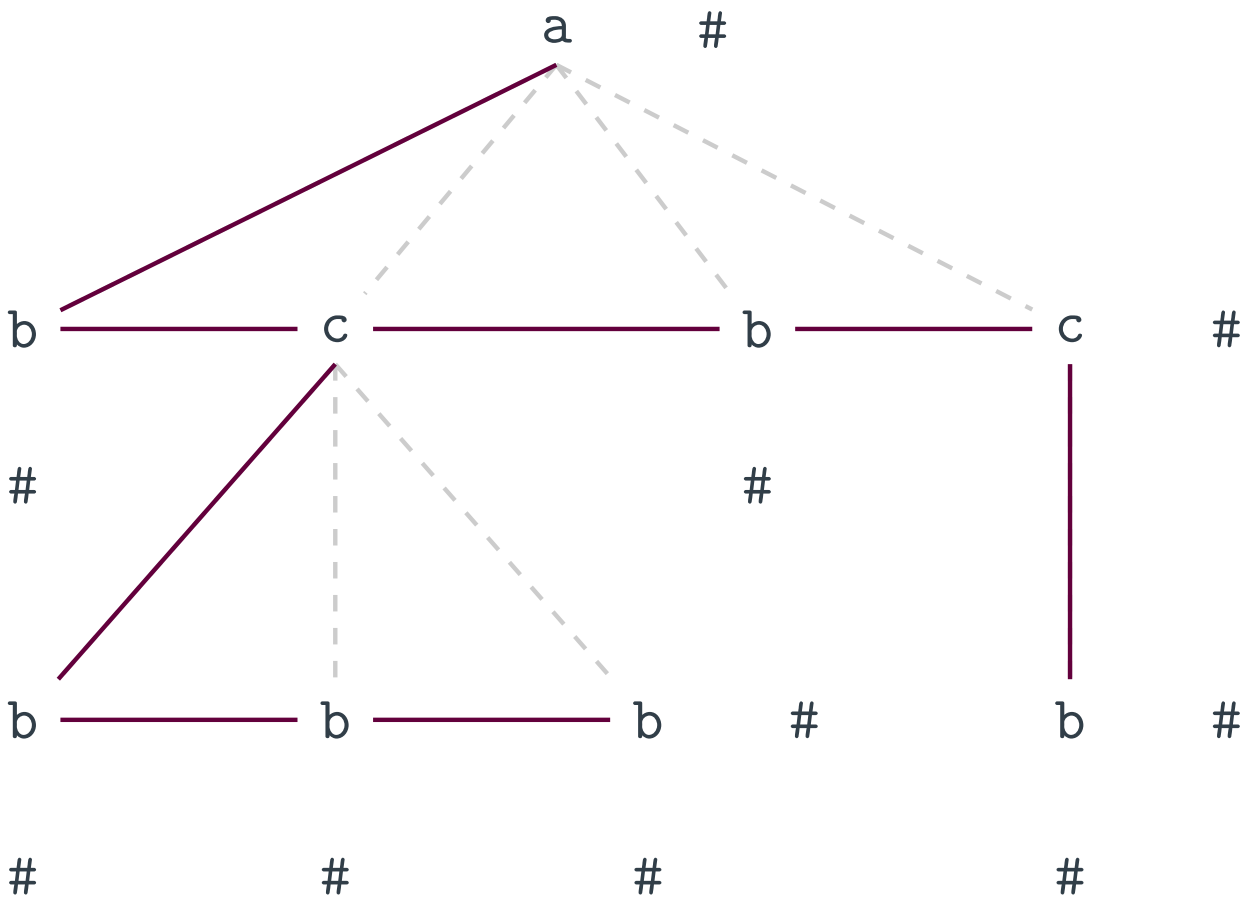
Example



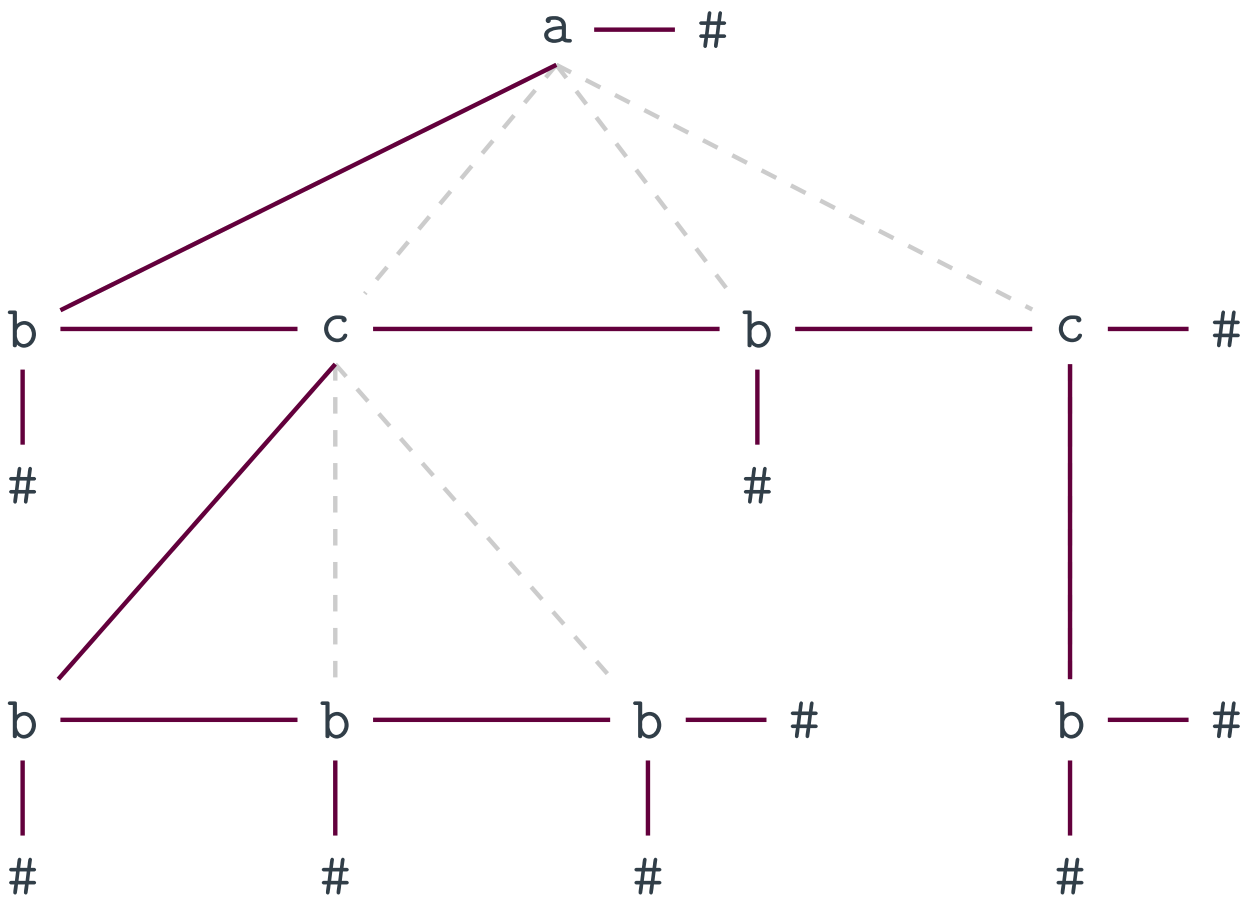
Example



Example



Example



Interpretation as a datastructure

université PARIS-SACLAY

This encoding correponds to the usual representation of a tree in memory (for instance in C):

- a node stores the label and a pointer to the linked list of children
- the first element of the list is directly accessible, the others require a traversal
- the `NULL` pointer corresponds to the # leaves

Structure de données (2)

université PARIS-SACLAY

```
struct node;
struct list {
    struct list *next; //right edge
    struct node *node;
};
struct node {
    char * label;
    struct list *children; //down edge
};
```

DTD

université PARIS-SACLAY

Document Type Definitions (DTD) allows one to define schemas, that is tree languages:

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

For each tag, we give its content as a regular expression over its children. A tag determines uniquely its content (we cannot have two alternative definitions for the same tag) $\Rightarrow$ TD déterministic

DTD (2)

université PARIS-SACLAY

The fact that DTD are TD deterministic means that we can validate a document in *streaming*, that is with memory bounded by the height of the document.

Why is it a good thing?

DTD (2)

université PARIS-SACLAY

The fact that DTD are TD deterministic means that we can validate a document in *streaming*, that is with memory bounded by the height of the document.

Why is it a good thing?

We can validate the document while reading the file.

What limitations ?

DTD (2)

université PARIS-SACLAY

The fact that DTD are TD deterministic means that we can validate a document in *streaming*, that is with memory bounded by the height of the document.

Why is it a good thing?

We can validate the document while reading the file.

What limitations ?

We cannot specify documents where there is a dependency between children.

XPath

université PARIS-SACLAY

The XPath language is a query language for trees. It allows one to select nodes. In its simplest form, (navigational Core XPath), queries have the form:

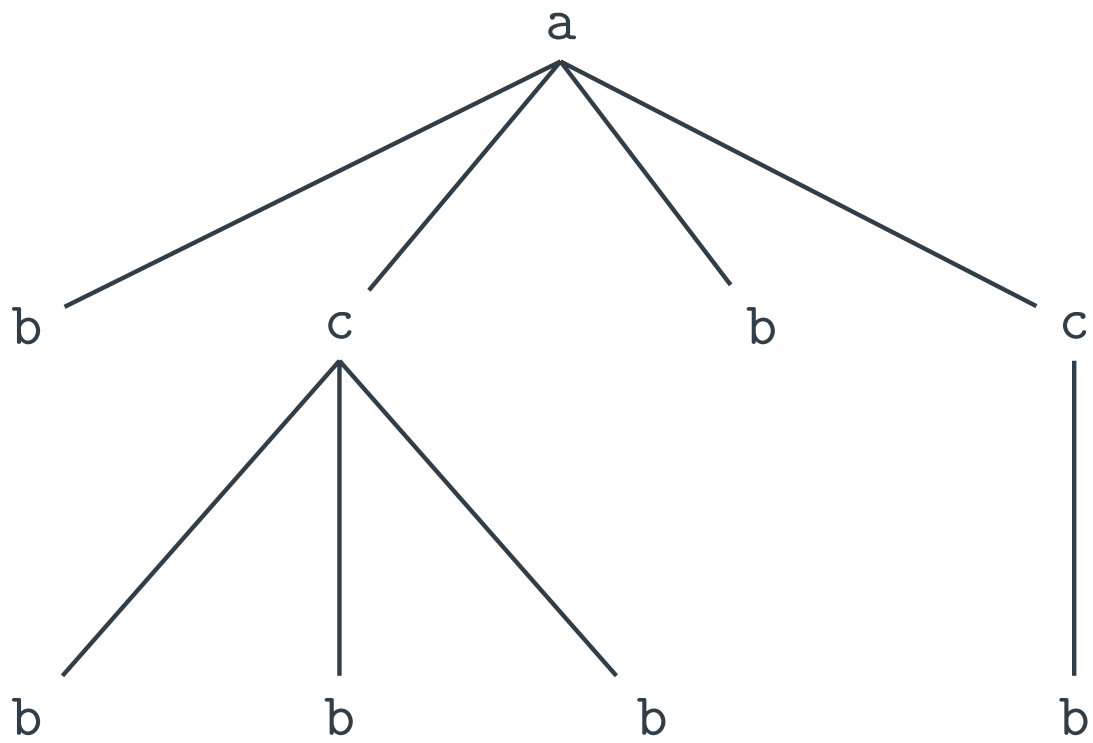
$$path ::= axe_1 :: test_1[\ pred_1]\ /\ldots/\ axe_n :: test_n[\ pred_n]$$

$$\begin{array}{lcl} axe & ::= & \texttt{child} \mid \texttt{descendant} \mid \texttt{parent} \mid \texttt{ancestor} \mid \ldots \\ test & ::= & tag \mid * \\ pred & ::= & path \mid pred_1\ \texttt{or}\ pred_2 \mid pred_1\ \texttt{and}\ pred_2 \mid \texttt{not}(pred_1) \end{array}$$

Exemple

université PARIS-SACLAY

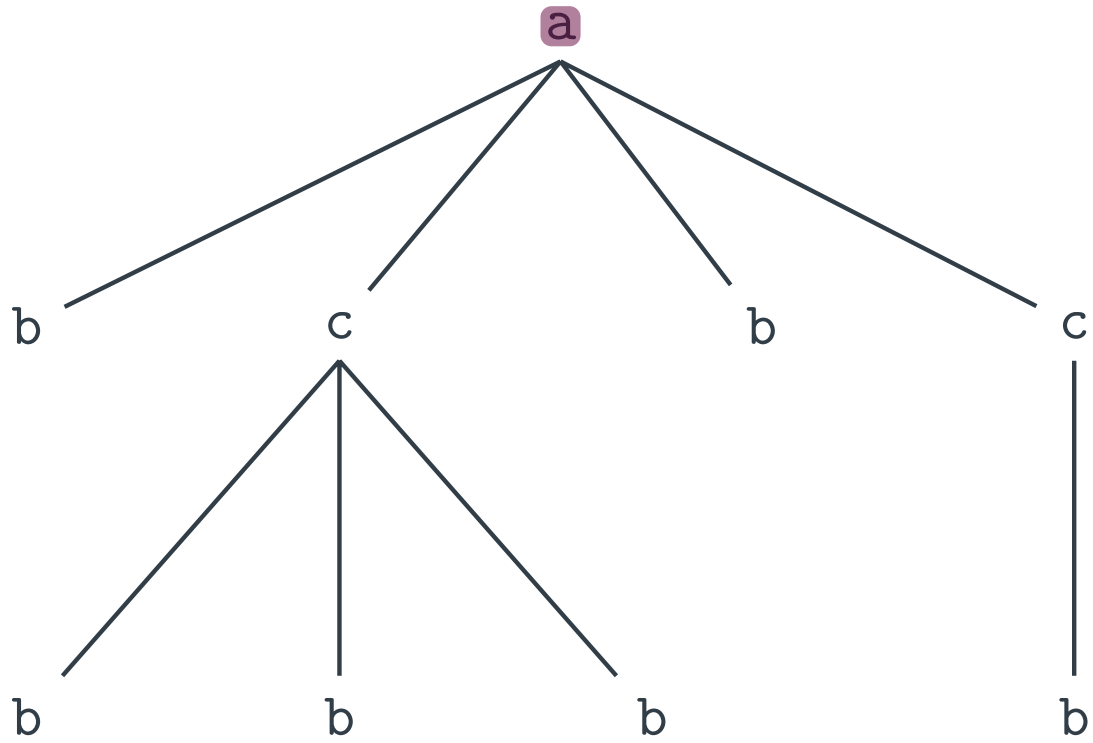
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

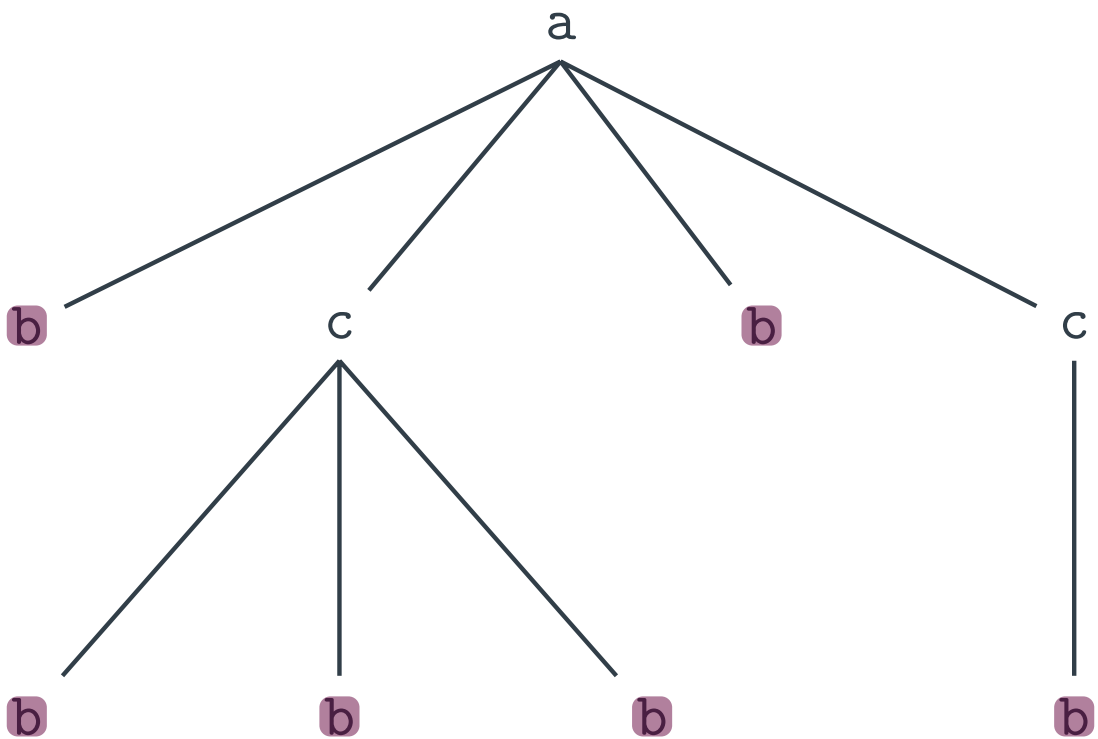
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

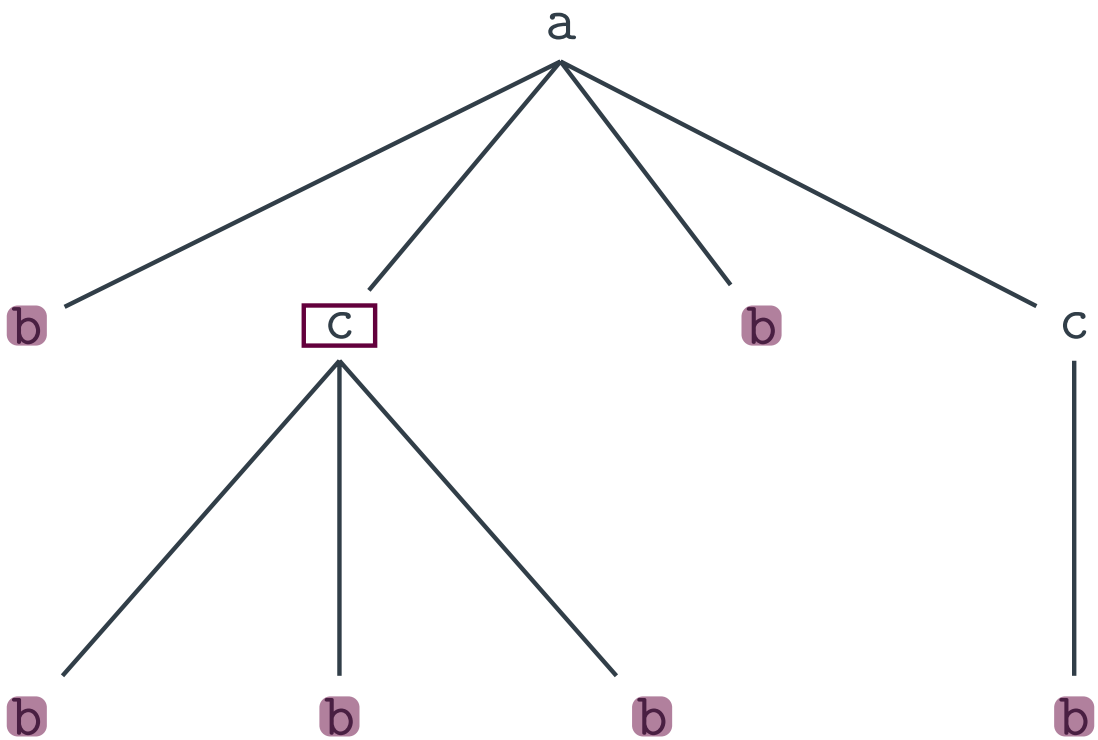
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

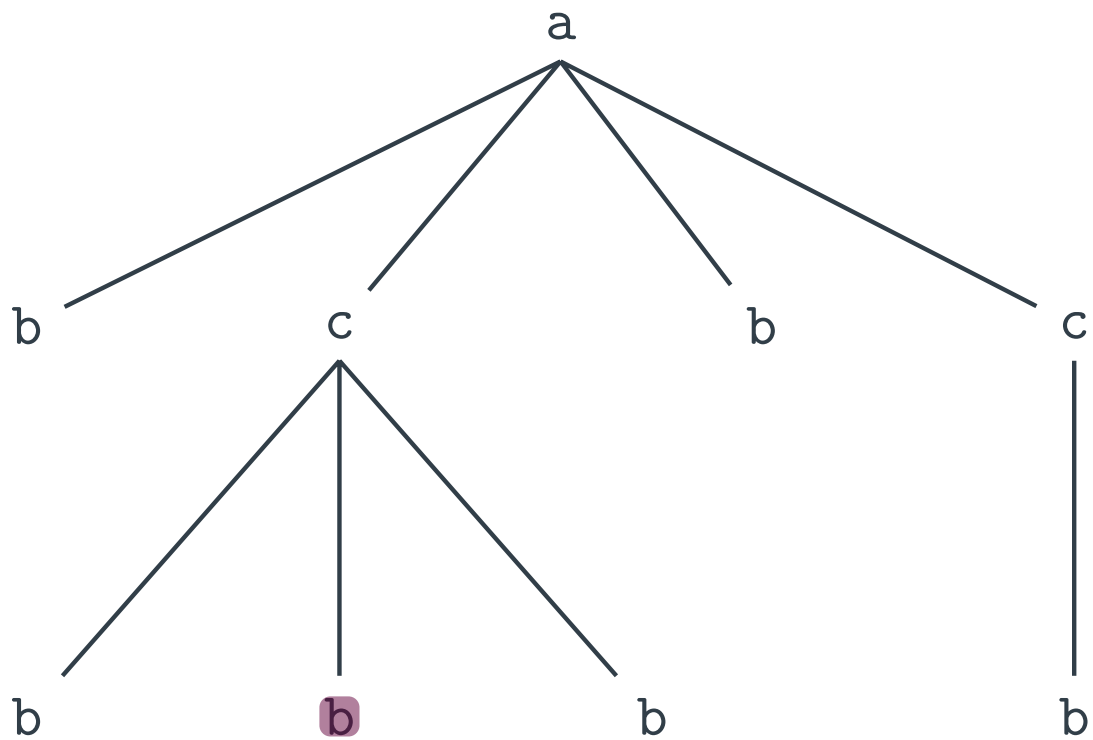
/descendant::b[parent::c]/parent::*/child::*



Exemple

université PARIS-SACLAY

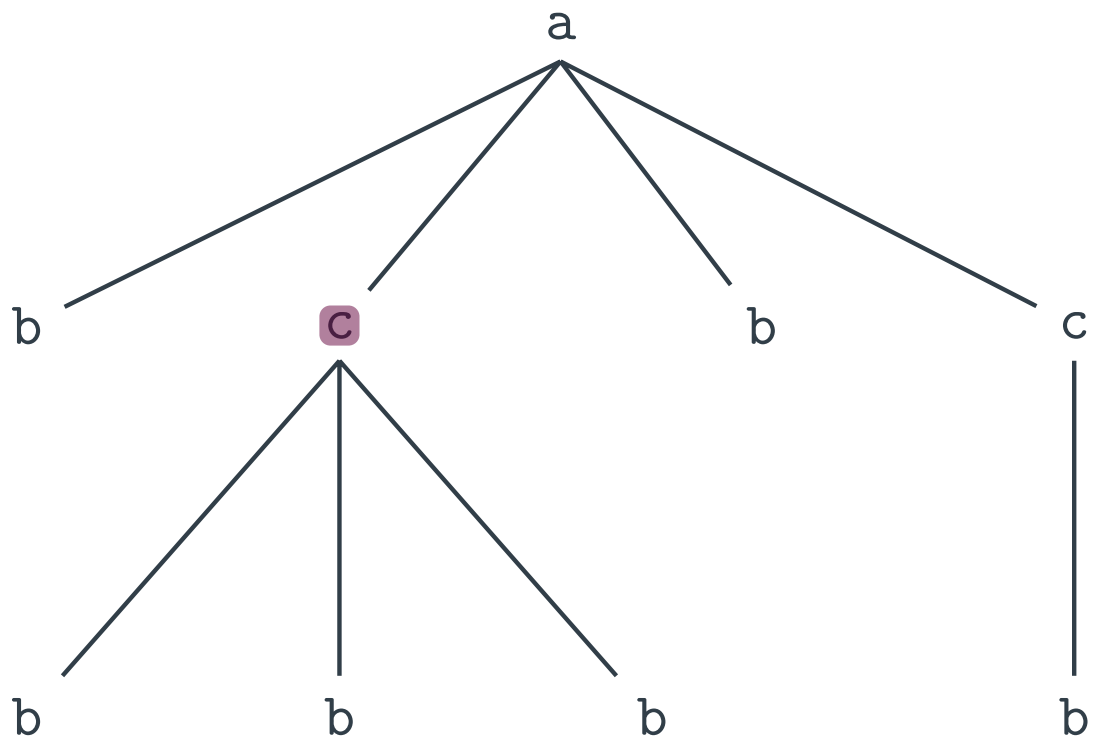
`/descendant::b[parent::c]/parent::*/child::*`



Exemple

université PARIS-SACLAY

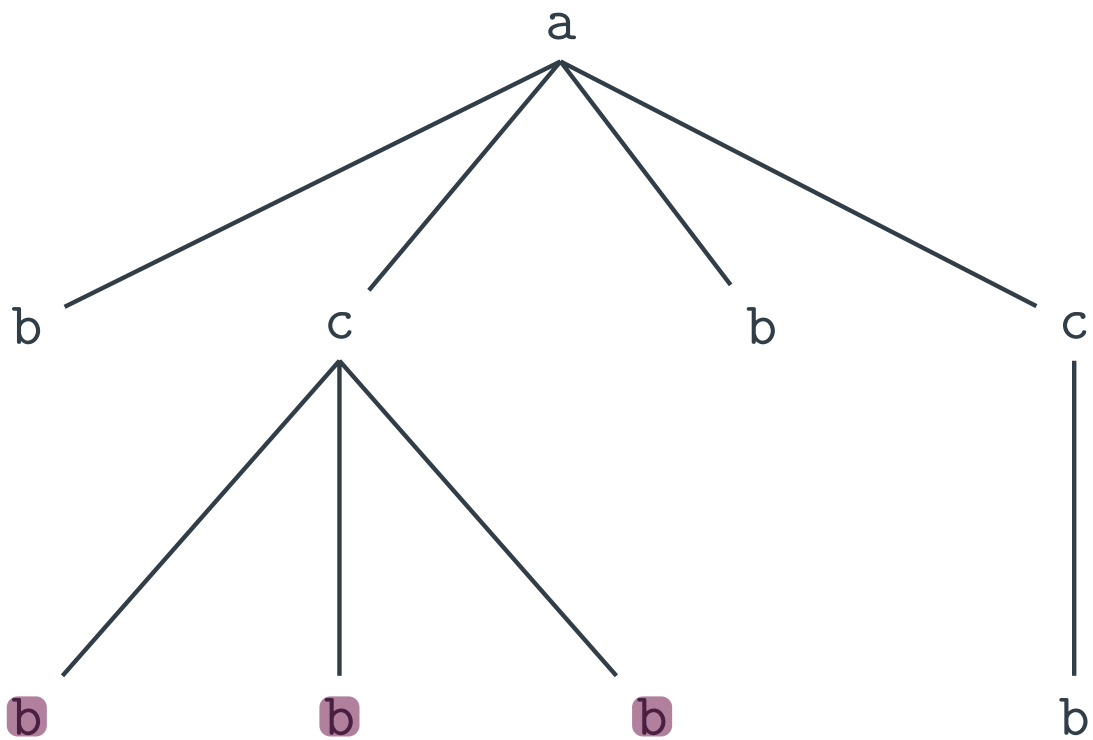
/descendant::b[parent::c]/parent::*/child::*



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.