Devoir 1 : Automates et XML

À rendre au plus tard le 18 mars 2010.
Tout retard sera pénalisé.

Mars 2010

						21
22	23	24	25	26	27	28
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Une version électronique (PDF) peut m'être envoyée par mail à ⟨schmitz@lsv.ens-cachan.fr⟩, les versions papiers doivent m'être rendues physiquement le 18 mars ou mises dans mon casier au LSV.

Ce devoir s'intéresse aux langages reconnaissables d'arbres d'arité non bornée, et à leurs applications dans le domaine des documents XML. Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Langages reconnaissables d'arbres d'arité non bornée

Arbres d'arité non bornée Un arbre ordonné étiqueté sur un alphabet Σ, d'arité non bornée, est défini comme une fonction t de $\mathbb{N}^*$ (c'est-à-dire des séquences d'entiers naturels) dans Σ, telle que son domaine $\mathrm{dom}(t)$ soit

- fermé par préfixe : si u et v sont deux séquences de $\mathbb{N}^*$, u est préfixe de v et $v \in \mathrm{dom}(t)$, alors $u \in \mathrm{dom}(t)$; et
- fermé par frère aîné : si $i < j$ sont deux entiers de $\mathbb{N}$, u une séquence de $\mathbb{N}^*$ et $uj \in \mathrm{dom}(t)$, alors $ui \in \mathrm{dom}(t)$.

On note $T(\Sigma)$ l'ensemble des arbres d'arité non bornée sur Σ. Les notions de hauteur ou de sous-arbre se généralisent immédiatement aux arbres d'arité non bornée.

Un arbre de racine étiquetée par a et de sous-arbres immédiats $t_1, \ldots, t_n$ est noté $a(t_1 \cdots t_n)$. La séquence $t_1 \cdots t_n$ d'arbres de $T(\Sigma)$ est aussi appelée une *haie*. Formellement, les ensembles $H(\Sigma)$ des haies et $T(\Sigma)$ des arbres d'arité non bornée sur Σ peuvent être définis inductivement par

- une séquence de $(T(\Sigma))^*$, y compris la séquence vide ε, est une haie ;
- si h est une haie et a un symbole de Σ, alors $a(h)$ est un arbre d'arité non bornée.

Exemple 1. On définit sur $\Sigma = \{a, b\}$ l'ensemble L_1 des arbres de hauteur 1, de racines étiquetées par a, de feuilles étiquetées par b, et qui contiennent un nombre impair de feuilles :

$$L_1 = \{t_n \mid n \geq 0\} = \{a(\ (b())^{2n+1}\ ) \mid n \geq 0\}$$
$$t_n\colon\ \varepsilon \mapsto a$$
$$m \mapsto b \qquad \forall m \leq 2n+1\ .$$

Le langage L_1 contient par exemple les arbres $t_0 = a(b())$ et $t_2 = a(b()b()b()b()b())$.

Exercice 1 (Automates de haies)**.** Un *automate de haies* non déterministe (NHA) sur Σ est un tuple $\mathcal{A} = \langle Q, \Sigma, \delta, F\rangle$ où Q est l'ensemble fini d'états, $F \subseteq Q$ l'ensemble des états acceptants, et $\delta \subseteq \mathrm{Rec}(Q^*) \times \Sigma \times Q$ est la relation finie de transition : chaque règle de δ est de la forme (R, a, q) avec R un langage reconnaissable de mots sur l'alphabet Q, a un symbole de Σ, et q un état de Q.

Un *calcul* de l'automate $\mathcal{A}$ sur l'arbre t de $T(\Sigma)$ est un arbre ρ de $T(Q)$ tel que $\mathrm{dom}(t) = \mathrm{dom}(\rho)$, et pour toute position u de $\mathrm{dom}(t)$ avec n successeurs (c'est-à-dire $un \in \mathrm{dom}(t)$ mais $u(n+1) \notin \mathrm{dom}(t)$), il existe une règle $(R, t(u), \rho(u))$ dans δ telle que $\rho(u1)\cdots\rho(un) \in R$. Le calcul est *acceptant* si $\rho(\varepsilon) \in F$. Le langage $L(\mathcal{A})$ de $\mathcal{A}$ est l'ensemble des arbres de $T(\Sigma)$ pour lesquels il existe un calcul acceptant. Un langage $L \subseteq T(\Sigma)$ est *reconnaissable* sur Σ s'il existe un NHA $\mathcal{A}$ tel que $L = L(\mathcal{A})$, et on écrira alors $L \in \mathrm{Rec}(T(\Sigma))$.

Alternativement, on peut voir un calcul comme une réécriture de termes. Soit $\mathcal{X} = \{x_1, x_2, \ldots\}$ un ensemble dénombrable de variables ; une transition (R, a, q) de δ est vue comme un ensemble de règles de réécriture linéaires entre termes de $T(\Sigma, Q, \mathcal{X})$:

$$a(q_1(x_1)\cdots q_n(x_n)) \to_{\mathcal{A}} q(a(x_1\cdots x_n)) \text{ si } q_1\cdots q_n \in R$$

Q étant un alphabet de symboles unaires et $\mathcal{X}$ de symboles constants. Les états $q, q_1, \ldots, q_n$ mis en jeu sont ceux du calcul ρ, construit inductivement des feuilles vers la racine. Un arbre t est alors reconnu si $t \to^*_{\mathcal{A}} q(t)$ avec $q \in F$.

La représentation concrète des langages reconnaissables dans la relation de transition d'un NHA peut utiliser différents formalismes, par exemple

- des expressions rationnelles, et on écrira alors NHA(RE),
- des automates de mots non déterministes, et on écrira NHA(NFA),
- des automates de mots non déterministes à double sens (boustrophédons), et on écrira NHA(2NFA), ou encore
- des automates de mots déterministes, pour lesquels on écrira NHA(DFA).

La *taille* $|\mathcal{A}|$ d'un NHA $\mathcal{A} = \langle Q, \Sigma, \delta, F\rangle$ est définie comme

$$|Q| + \sum_{(R,a,q)\in\delta} |R| + 2,$$

où $|R|$ dénote la taille de la représentation concrète du langage R, par exemple celle de l'expression rationnelle pour un NHA(RE).

Nous considérons les propriétés élémentaires des NHA et de leurs langages.

[1] 1. Donner un NHA(RE) pour le langage L_1 de l'exemple 1.

[1] 2. Un NHA $\mathcal{A}$ est *complet* s'il existe un calcul de $\mathcal{A}$ pour tout arbre t de $T(\Sigma)$. Montrer que pour tout NHA, il existe un NHA complet équivalent. Quelle est la complexité de cette construction sur un NHA(RE) ? Sur un NHA(DFA) ?

[2] 3. Un NHA est *normalisé* s'il n'existe pas deux règles distinctes (R, a, q) et (R', a, q) dans δ. Montrer que pour tout NHA, il existe un NHA normalisé équivalent. Quelle est la complexité de cette construction sur un NHA(RE) ? Sur un NHA(DFA) ?

[2] 4. Montrer que $\mathrm{Rec}(T(\Sigma))$ est fermé par union et intersection.

Exercice 2 (Déterminisation)**.** Un automate à haies $\mathcal{A} = \langle Q, \Sigma, \delta, F\rangle$ est *déterministe* (noté DHA) si, pour toutes règles (R, a, q) et (R', a, q'), $R \cap R' = \emptyset$ ou $q = q'$; δ devient alors une fonction partielle de $Q^* \times \Sigma$ dans Q, correspondant à un automate de haies déterministe ascendant.

[1] 1. Soient $\mathcal{A} = \langle Q, \Sigma, \delta, F\rangle$ un NHA, $a \in \Sigma$ et $q \in Q$. Montrer que

$$S_{a,q} = \{s_1 \cdots s_n \in (2^Q)^* \mid \exists (R, a, q) \in \delta, \exists q_1 \in s_1, \ldots, q_n \in s_n, q_1 \cdots q_n \in R\} .$$

est un langage reconnaissable de mots sur $(2^Q)^*$.

[4] 2. Montrer que, pour tout NHA $\mathcal{A}$, il existe un DHA $\mathcal{A}'$ équivalent.

[1] 3. Montrer que $\mathrm{Rec}(T(\Sigma))$ est fermé par complément.

Exercice 3 (Langage de flux balisé)**.** Soit $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$ une copie disjointe de Σ. Le *flux balisé* $\mathrm{flux}(t)$ d'un arbre $t \in T(\Sigma)$ est un mot de $(\Sigma \uplus \overline{\Sigma})^*$ défini inductivement par

$$\begin{aligned} \mathrm{flux}(a(h)) &= a \cdot \mathrm{flux}(h) \cdot \overline{a} && \forall a \in \Sigma \text{ et } h \in H(\Sigma) \\ \mathrm{flux}(t_1 \cdots t_n) &= \mathrm{flux}(t_1) \cdots \mathrm{flux}(t_n) && \forall n \in \mathbb{N} \text{ et } t_1, \ldots, t_n \in T(\Sigma) . \end{aligned}$$

Par exemple, $\mathrm{flux}(a(b()b()b())) = a\, b\overline{b}\, b\overline{b}\, b\overline{b}\, \overline{a}$. On étend naturellement cette définition aux langages d'arbres par $\mathrm{flux}(L) = \{\mathrm{flux}(t) \mid t \in L\}$.

[3] Montrer que si L est un langage de $\mathrm{Rec}(T(\Sigma))$, alors $\mathrm{flux}(L)$ est un langage algébrique sur $\Sigma \uplus \overline{\Sigma}$.

2 XPath et automates cheminants

Exercice 4 (Automates cheminants)**.** Soit t un arbre de $T(\Sigma)$ et u une position de $\mathrm{dom}(t)$. Le *type* $\theta(u)$ de u est un couple de l'ensemble $\mathrm{Type} = \{\mathsf{i}, \mathsf{f}\} \times \{\mathsf{u}, \mathsf{g}, \mathsf{d}, \mathsf{r}, \mathsf{a}\}$ défini

par $\theta(u) = (\nu, \tau)$ tel que

$\nu = \mathsf{i}$ si u est un nœud *interne* : $\exists j \in \mathbb{N}, uj \in \mathrm{dom}(t)$,
$\nu = \mathsf{f}$ sinon, c'est-à-dire si u est un nœud *feuille* : $\nexists j \in \mathbb{N}, uj \in \mathrm{dom}(t)$;
$\tau = \mathsf{u}$ si u est à la fois un nœud *fils gauche* et *fils droit* : $\exists v \in \mathbb{N}^*, u = v0$ et $v1 \notin \mathrm{dom}(t)$,
$\tau = \mathsf{g}$ si u est un nœud *fils gauche* : $\exists v \in \mathbb{N}^*, u = v0$ et $v1 \in \mathrm{dom}(t)$,
$\tau = \mathsf{d}$ si u est un nœud *fils droit* : $\exists v \in \mathbb{N}^*, n > 0, u = vn$ et $v(n+1) \notin \mathrm{dom}(t)$,
$\tau = \mathsf{r}$ si u est la *racine* de t : $u = \varepsilon$, et
$\tau = \mathsf{a}$ dans les autres cas.

Un *automate cheminant* (TWA) sur Σ est un tuple $\mathcal{A} = \langle Q, \Sigma, \delta, I, F \rangle$ composé d'un ensemble fini d'états Q, d'une relation finie de transitions $\delta \subseteq Q \times \mathrm{Type} \times \Sigma \times D \times Q$, où $D = \{\uparrow, \swarrow, \leftarrow, \rightarrow, \circlearrowleft\}$ est un ensemble de directions, $I \subseteq Q$ est l'ensemble d'états initiaux, et $F \subseteq Q$ d'états finaux.

Soit t un arbre de $T(\Sigma)$ et $\mathcal{A}$ un automate cheminant sur Σ. Une *configuration* est une paire (u, q) avec u une position de $\mathrm{dom}(t)$ et q un état de Q. Une transition $(q, u) \to_{t,\mathcal{A}} (q', v)$ entre deux configurations est possible dans $\mathcal{A}$ s'il existe une transition $(q, \theta(u), t(u), d, q')$ dans δ avec

si $d = \uparrow$ alors u est le père de v : $\exists i \in \mathbb{N}, u = vi$,
si $d = \swarrow$ alors v est le fils gauche de u : $v = u0$,
si $d = \leftarrow$ alors v est le frère gauche de u : $\exists w \in \mathbb{N}^*, i \in \mathbb{N}, u = w(i+1)$ et $v = wi$,
si $d = \rightarrow$ alors v est le frère droit de u : $\exists w \in \mathbb{N}^*$, $i \in \mathbb{N}$, $u = wi$ et $v = w(i+1)$,
si $d = \circlearrowleft$ alors $u = v$.

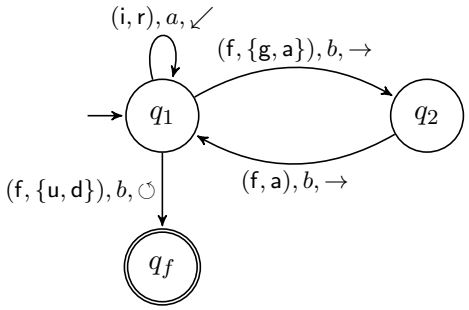
Un *calcul* de $\mathcal{A}$ est une séquence de transitions $(q, u) \to^*_{t,\mathcal{A}} (q', v)$. Un calcul est *acceptant* si $q \in I$ et $q' \in F$. La *relation* sur $\mathrm{dom}(t)$ définie par $\mathcal{A}$ est

$$R_t(\mathcal{A}) = \{(u, v) \in \mathrm{dom}(t)^2 \mid \exists q \in I, q' \in F, (q, u) \to^*_{t,\mathcal{A}} (q', v)\} .$$

Le *langage* de $\mathcal{A}$ est l'ensemble des arbres ayant un calcul acceptant depuis la racine :

$$L(\mathcal{A}) = \{t \in T(\Sigma) \mid R_t(\mathcal{A})(\varepsilon) \neq \emptyset\} .$$

Par exemple, l'automate suivant reconnaît le langage L_1 de l'exemple 1 :



[1] 1. Donner un automate cheminant pour l'ensemble des arbres de $T(\{a, b\})$ dont toutes les feuilles sont étiquetées par a.

[2] 2. Soit R un langage rationnel de mots sur Σ. Donner un automate d'arbre cheminant qui accepte un arbre de $T(\Sigma)$ s'il existe au moins une branche étiquetée par un mot de R.

[1] 3. Montrer que si $L \subseteq T(\Sigma)$ est reconnu par un automate cheminant $\mathcal{A}$, alors il existe un automate cheminant $\mathcal{A}'$ tel que

$$L(\mathcal{A}') = L = \{t \in T(\Sigma) \mid (\varepsilon, \varepsilon) \in R_t(\mathcal{A}')\} .$$

[2] 4. Montrer que les langages d'arbres reconnus par des automates cheminants sont fermés par union et intersection.

Exercice 5 (Reconnaissabilité)**.** On souhaite montrer que les langages reconnus par automates cheminants sont tous dans $\mathrm{Rec}(T(\Sigma))$, c'est-à-dire pour un TWA $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ construire un NHA $\mathcal{A}' = \langle Q', \Sigma, \delta', F'\rangle$ tel que $L(\mathcal{A}) = L(\mathcal{A}')$.

Soit t un arbre de $T(\Sigma)$ et u une position de $\mathrm{dom}(t)$. Une configuration (q, v) est *u-bornée* si u est un préfixe de v. Un calcul de $\mathcal{A}$ sur t de la forme $(q, v) \to^*_{t,\mathcal{A}} (q', w)$ est u-borné si toutes ses configurations intermédiaires, les configurations extrêmes (q, v) et (q', w) inclues, sont u-bornées ; on note alors $(q, v) \xrightarrow{\geq u}{}^*_{t,\mathcal{A}} (q', w)$. Un calcul u-borné ne visite jamais ni le père, ni les frères de u.

On définit comme ensemble d'états $Q' = \mathrm{Type} \times \Sigma \times (2^{Q^2})$. L'invariant que l'on souhaite faire respecter par δ' est que $((\nu, \tau), a, P)$ apparaisse comme état $\rho(u)$ dans un calcul ρ de $\mathcal{A}'$ sur t si et seulement s'il existe un calcul u-borné $(q, u) \xrightarrow{\geq u}{}^*_{t,\mathcal{A}} (q', u)$ dans $\mathcal{A}$ avec (q, q') dans P.

[3] 1. En guise d'échauffement, on s'intéresse aux calculs de $\mathcal{A}$ qui restent sur place sur une feuille. On définit ainsi pour $(\nu, \tau) \in \mathrm{Type}$ et $a \in \Sigma$ l'ensemble

$$E(\nu, \tau, a) = \{(q_0, q_n) \in Q^2 \mid \exists t \in T(\Sigma), u \in \mathrm{dom}(t), q_1, \ldots, q_{n-1} \in Q, \theta(u) = (\nu, \tau), \\ t(u) = a \text{ et } (q_0, u) \to_{t,\mathcal{A}} (q_1, u) \cdots (q_{n-1}, u) \to_{t,\mathcal{A}} (q_n, u)\}$$

des paires (q_0, q_n), $n \geq 0$, pour lesquelles il existe un calcul de $\mathcal{A}$ qui reste sur place (notez que c'est un calcul u-borné particulier).

Exprimez de manière générale $E(\nu, \tau, a)$ à l'aide de δ. En déduire le sous-ensemble des règles de δ' qui reconnaît les feuilles des arbres de $L(\mathcal{A})$, et montrer qu'il vérifie bien l'invariant proposé.

[5] 2. Soit $p = ((\mathrm{i}, \tau), a, P)$ un état de Q'. On veut construire les règles de δ' de la forme (R_p, a, p). Pour cela, définissez un 2NFA $\mathcal{A}_p = \langle Q'', Q', \delta'', I_p, F_p\rangle$, avec $\delta'' \subseteq Q'' \times Q' \times \{\leftarrow, \rightarrow\} \times Q''$, qui définit un langage $L(\mathcal{A}_p) = R_p \in \mathrm{Rec}(Q'^*)$ qui vérifie bien l'invariant proposé.

[1] 3. Conclure.

Exercice 6 (XPath)**.** Une *expression XPath simple* φ sur Σ est un terme défini par la syntaxe abstraite

$$\varphi ::= a \mid d \mid d^* \mid \varphi/\varphi \mid \varphi + \varphi$$

où d est une direction dans $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ et a un symbole de Σ. L'interprétation $[\![\varphi]\!]_t$ d'une expression XPath simple sur un arbre $t \in T(\Sigma)$ est une relation sur $\mathrm{dom}(t)$ définie inductivement par

$$\begin{aligned}
[\![a]\!]_t &= \{(u,u) \mid u \in \mathrm{dom}(t), t(u) = a\} \\
[\![\uparrow]\!]_t &= \{(ui,u) \mid ui \in \mathrm{dom}(t), i \in \mathbb{N}\} \\
[\![\downarrow]\!]_t &= \{(u,ui) \mid ui \in \mathrm{dom}(t), i \in \mathbb{N}\} \\
[\![\leftarrow]\!]_t &= \{(u(i+1),ui) \mid u(i+1) \in \mathrm{dom}(t), i \in \mathbb{N}\} \\
[\![\rightarrow]\!]_t &= \{(ui,u(i+1)) \mid u(i+1) \in \mathrm{dom}(t), i \in \mathbb{N}\} \\
[\![\uparrow^*]\!]_t &= \{(uv,u) \mid uv \in \mathrm{dom}(t), v \in \mathbb{N}^*\} \\
[\![\downarrow^*]\!]_t &= \{(u,uv) \mid uv \in \mathrm{dom}(t), v \in \mathbb{N}^*\} \\
[\![\leftarrow^*]\!]_t &= \{(uj,ui) \mid uj \in \mathrm{dom}(t), i \leq j \in \mathbb{N}\} \\
[\![\rightarrow^*]\!]_t &= \{(ui,uj) \mid uj \in \mathrm{dom}(t), i \leq j \in \mathbb{N}\} \\
[\![\varphi/\psi]\!]_t &= [\![\varphi]\!]_t \cdot [\![\psi]\!]_t = \{(u,w) \mid \exists v, (u,v) \in [\![\varphi]\!]_t, (v,w) \in [\![\psi]\!]_t\} \\
[\![\varphi + \psi]\!]_t &= [\![\varphi]\!]_t \cup [\![\psi]\!]_t
\end{aligned}$$

[4] Montrer que, pour toute expression XPath simple φ, on peut construire un TWA $\mathcal{A}$ tel que $[\![\varphi]\!]_t = R_t(\mathcal{A})$ pour tout t de $T(\Sigma)$.

A Compléments

Cette section rassemble pour les curieux quelques informations sur les documents XML et les expressions XPath.

A.1 Documents XML

Le format de documents XML (pour *eXtensible Markup Language*) offre un standard de représentation pour des données structurées. Son importance actuelle dans le développement d'applications découle de l'intérêt de disposer d'un format unifié dans lequel échanger des données informatiques et développer des dialectes spécialisés. La plupart des données échangées sur le Web sont écrites dans un dialecte XML, à commencer par les pages écrites en XHTML.

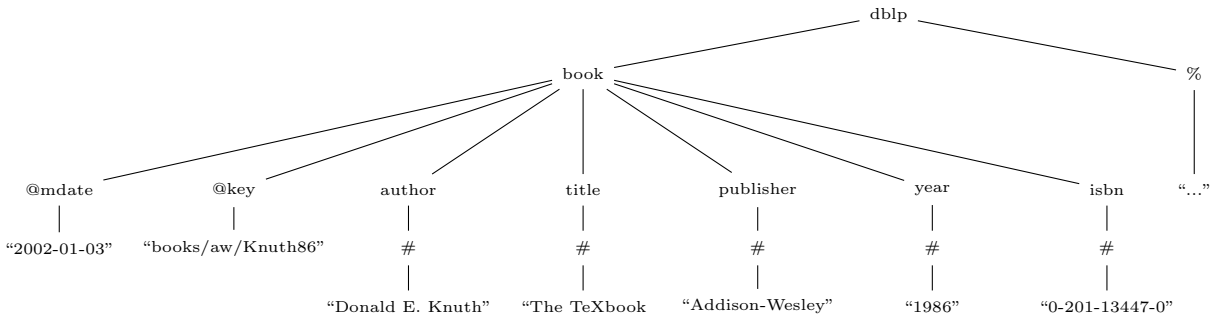
Pour les curieux, un document XML typique a la forme suivante :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
  <book mdate="2002-01-03" key="books/aw/Knuth86">
    <author>Donald E. Knuth</author>
    <title>The TeXbook</title>
    <publisher>Addison-Wesley</publisher>
    <year>1986</year>
    <isbn>0-201-13447-0</isbn>
  </book>
  <!-- ... -->
</dblp>
```

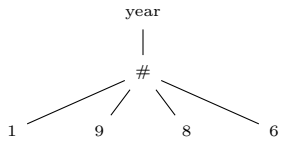
Outre un en-tête qui décrit en particulier le dialecte adopté (défini dans `dblp.dtd`), le document est constitué d'*éléments* entourés par une balise ouvrante (comme `<book>`) et une fermante (comme `</book>`) bien balancées. Ces éléments peuvent contenir des *attributs* (comme `mdate`), d'autres éléments, directement du *texte* (comme `Donald E. Knuth`), ou encore des *commentaires* (comme `<!-- ... -->`).

Un document XML peut être vu comme un arbre d'arité non bornée, où l'on peut de plus mettre les attributs des éléments et le contenu textuel dans des nœuds fils des éléments.

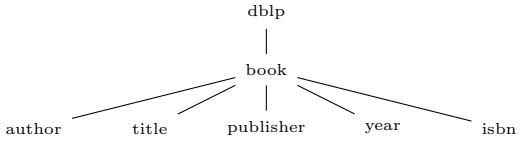
L'alphabet $\Sigma = \Sigma_e \uplus \Sigma_@ \uplus \{\#, \%\} \uplus \Sigma_{\text{text}}$ distingue alors entre
- étiquettes des éléments dans Σ_e (comme `book`),
- celles des attributs dans $\Sigma_@$ (comme `@mdate`),
- l'étiquette des nœuds textuels (`#`),
- celle des nœuds commentaires (`%`), et
- les étiquettes des contenus textuels Σ_{text} (comme `2002-01-03` ou `Donald E. Knuth` – cet alphabet est ici l'alphabet latin `ISO-8859-1` déclaré en en-tête).



L'arbre ci-dessus ignore les contenus textuels vides, comme les espaces et retours à la ligne entre `<dblp>` et `<book>`. Pour simplifier, on a aussi représenté les contenus textuels comme des chaînes de caractères, au lieu d'arbres, comme par exemple (on pourrait aussi écrire les chaînes verticalement, mais puisque nous avons des arbres d'arité non bornée, autant en profiter) :



On peut donc travailler sur des représentations qui tiennent compte des attributs et du texte dans les documents. Cependant, les modélisations à base d'automates de haies se contentent souvent des éléments. Pour l'exemple de document, cela signifie qu'on le verra comme un arbre de $T(\Sigma_e)$:



A.2 XPath

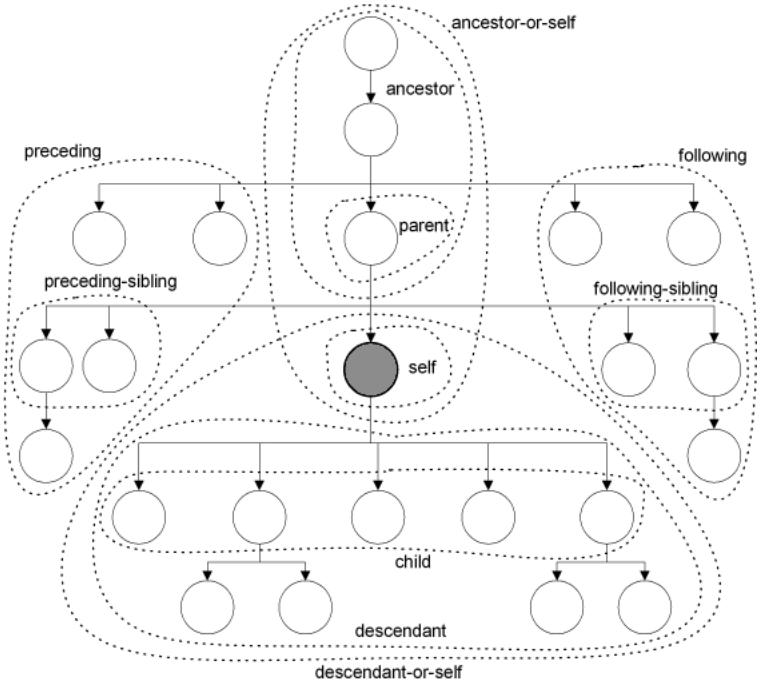
XPath est un langage qui permet d'écrire des sortes d'expressions rationnelles sur les documents XML. La recommandation W3C correspondante est `http://www.w3.org/TR/xpath`.

Fragment navigationnel Comme son nom l'indique, XPath langage raisonne surtout sur les chemins. Les chemins XPath fonctionnent un peu comme l'adressage dans l'arborescence des fichiers UNIX, avec "`.`" pour désigner l'élément courant, "`/`" pour désigner la racine ou pour trouver les éléments fils, et "`..`" pour remonter au père ; le chemin "`a//b`" trouve tous les éléments `b` descendants de l'élément `a`. Enfin, on peut utiliser le wildcard `*` : `a/*` désigne tous les fils de `a`, et `b//*` désigne tous les descendants de `b`.

Par exemple, si on travaille sur le document XML du début du devoir, l'expression XPath suivante

```
/dblp/*/title
```

va sélectionner tous les fils `title` de n'importe quel nœud dont le père est un nœud `dblp` à la racine du document. Ce chemin est absolu (il commence par la racine "`/`"), mais en général une requête XPath va sélectionner des nœuds en suivant un chemin relativement à un *nœud courant*.



Les chemins que nous venons de voir sont en fait des notations abrégées pour différents *axes* de recherche : "`..`" correspond à l'axe `parent::`, "`//`" à l'axe `descendant::`. Des généralisations de ces axes existent, comme `descendant-or-self::`, `ancestor::`, `ancestor-or-self::`, et ainsi de suite.

Enfin, on peut tester l'existence d'un frère avant ou après l'élément atteint par `preceding-sibling::` et `following-sibling::` respectivement. Ainsi, le chemin `preceding-sibling::a/b` trouve un élément `b` fils d'un élément `a` qui précède immédiatement le nœud courant.

Les expressions de l'exercice 6 ne modélisent que ce fragment navigationnel de XPath, mais ce langage va plus loin.