Automata theory and its applications

Lecture 21-22: Automata for XML

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences

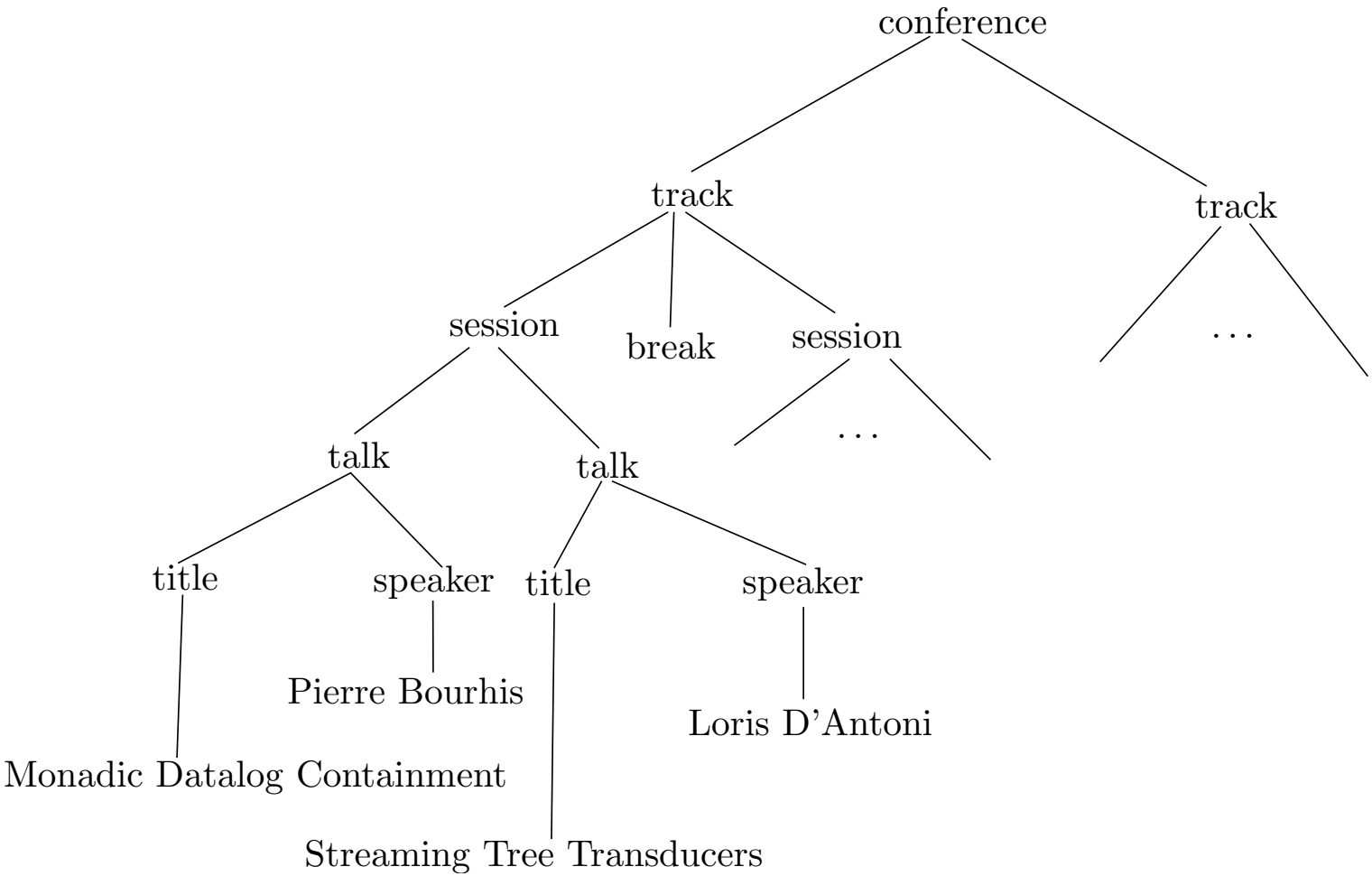
May 9, 2013

Outline

XML documents: An example

```
⟨ conference ⟩
  ⟨ track ⟩
    ⟨ session ⟩
      ⟨ talk ⟩
        ⟨ title ⟩
          Monadic Datalog Containment
        ⟨ /title ⟩
        ⟨ speaker ⟩
          Pierre Bourhis
        ⟨ /speaker ⟩
      ⟨ /talk ⟩
      ⟨ talk ⟩
        ⟨ title ⟩
          Streaming Tree Transducers
        ⟨ /title ⟩
        ⟨ speaker ⟩
          Loris D'Antoni
        ⟨ /speaker ⟩
      ⟨ /talk ⟩
    ⟨ /session ⟩
    ⟨ break ⟩
    ⟨ /break ⟩
    ⟨ session ⟩
    ...
    ⟨ /session ⟩
  ⟨ /track ⟩
  ⟨ track ⟩
    ...
  ⟨ /track ⟩
⟨ /conference ⟩
```

XML documents: An example



XML documents: An example

The schema:

<!DOCTYPE CONFERENCE [
<!ELEMENT conference (track$^+$ | (session, break?)$^+$)>
<!ELEMENT track (session, break?)$^+$ >
<!ELEMENT session (talk$^+$)>
<!ELEMENT talk (title, speaker)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT speaker (#PCDATA)>
] >

More formally

conference $\longrightarrow$ track$^+$ + session (break + ε)$^+$,
track $\longrightarrow$ session (break+ ε)$^+$
session $\longrightarrow$ talk$^+$
talk $\longrightarrow$ title speaker
title $\longrightarrow$ DATA
speaker $\longrightarrow$ DATA

XML documents: An example

Navigation and query: XPath

/conference//talk/title

Select all the titles of talks in a conference

/conference//track[/break]

Select all the tracks containing at least one break

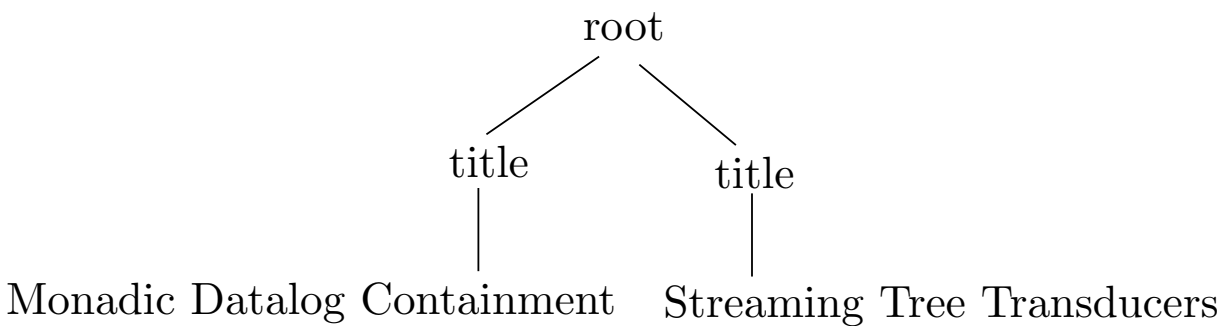
/conference//track[/break]//talk/title

Select the titles of all talks in a track containing at least one break

XML documents: An example

Transformation: XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/conference">
    <root>
      <xsl:for-each select="//title">
        <title>
          <xsl:value-of select="."/>
        </title>
      </xsl:for-each>
    </root>
</xsl:stylesheet>
```



Outline

Outline

Document Type Definition (DTD)

A *DTD* D is a tuple (Σ, s, δ), where

- $s \in \Sigma$ is the start symbol,
- and δ is a mapping assigning to each symbol a regular expression over Σ.

Suppose $\delta(a) = r$, then r is also called the *content model*.
We also write $\delta(a) = r$ as $a \to r$.

DTD seen as a DFHA (Deterministic Finite Hedge Automata):
$\mathcal{A}_D = (Q_D, \Sigma, \delta_D, F)$, where

- $Q_D = \Sigma$, $F = \{s\}$,
- $\delta_D = \{a(\delta(a)) \to a \mid a \in \Sigma\}$.

Proposition. DTD < NFHA.

Proof.

Consider the language $L = \{a(b(c,d)), a'(b(d,c))\}$.
Suppose L is defined by a DTD, then $a \to r$ and $b \to r'$ s.t. $b \in L(r)$ and $dc \in L(r')$.
It follows that $a(b(d,c)) \in L$, a contradiction. □

Deterministic DTD

The intuition: A rule $a \to r$ is deterministic

For a given word $w \in L(r)$, we can uniquely match each symbol in w to an occurrence of the same symbol in r by a single left to right pass without looking ahead in the input.

Marking of regular expressions

Let r be an regular expression over Σ.
The marking of r (denoted by r'):
The regular expression over the alphabet $\Sigma' = \{a_i \mid a \in \Sigma, i \in \mathbb{N}\}$ obtained by replacing the i-th occurrence of a by a_i.

Example: $(a+b)^*(ab)^* \Rightarrow (a_1+b_1)^*(a_2b_2)^*$.

Deterministic DTD

The intuition: A rule $a \to r$ is deterministic

For a given word $w \in L(r)$, we can uniquely match each symbol in w to an occurrence of the same symbol in r by a single left to right pass without looking ahead in the input.

Deterministic regular expressions (DRE)

A regular expression r over an alphabet Σ is *deterministic*

if for all words $u, v, w \in \Sigma'$ and all symbols $x_i, y_j \in \Sigma'$ with $ux_iv, uy_jw \in L(r')$, then $x_i \neq y_j$ implies $x \neq y$.

Example: $ab + ac$ is not deterministic since $a_1b_1, a_2c_1 \in L(a_1b_1 + a_2c_1)$, but $a_1 \neq a_2$. On the other hand, $a(b + c)$ is deterministic.

Remark (DRE < RE). $(a + b)^*b(a + b)$ cannot be defined by DREs (A. Brüggemann-Klein and D. Wood. One-unambiguous regular languages. Information and Computation, 142(2):182-206, 1998).

A rule $a \to r$ is deterministic if r is deterministic.

A DTD is deterministic if all its rules are deterministic.

From DRE to DFA

Several notations:

Let r be a regular expression, r' be the marking of r, and $a_i \in \Sigma'$.

- $\mathrm{sym}(r')$: the set of symbols from Σ' occurring in r',
- $\mathrm{first}(r') = \{b_j \in \Sigma' \mid \exists u.\ b_j u \in L(r')\}$,
- $\mathrm{last}(r') = \{b_j \in \Sigma' \mid \exists u.\ u b_j \in L(r')\}$,
- $\mathrm{follow}(r', a_i) = \{b_j \in \Sigma' \mid \exists u, v.\ u a_i b_j v \in L(r')\}$.

Example: Let $r' = (a_1 + b_1)^*(a_2 b_2)^+$,

- $\mathrm{sym}(r') = \{a_1, b_1, a_2, b_2\}$,
- $\mathrm{first}(r') = \{a_1, b_1, a_2\}$, $\mathrm{last}(r') = \{b_2\}$,
- $\mathrm{follow}(r', a_1) = \mathrm{follow}(r', b_1) = \{a_1, b_1, a_2\}$,
- $\mathrm{follow}(r', a_2) = \{b_2\}$, $\mathrm{follow}(r', b_2) = \{a_2\}$.

From DRE to DFA

Proposition. Let r be a regular expression and r' be the marking of r. Then $\forall w = x_1 \dots x_n \in (\Sigma')^+$, $w \in L(r')$ iff w satisfies the following three conditions,

$x_1 \in \text{first}(r')$, $x_n \in \text{last}(r')$, $\forall i : 1 \leq i < n$, $x_{i+1} \in \text{follow}(r', x_i)$.

Observation.

r' can be defined inductively by the following rules,

$r' = \varepsilon \mid a_i (a_i \in \Sigma') \mid r'_1 + r'_2 \mid r'_1 r'_2 \mid (r'_1)^*$ *s.t.* $\text{sym}(r'_1) \cap \text{sym}(r'_2) = \varnothing$.

From DRE to DFA

Proposition. Let r be a regular expression and r' be the marking of r. Then $\forall w = x_1 \dots x_n \in (\Sigma')^+$, $w \in L(r')$ iff w satisfies the following three conditions,

$x_1 \in \text{first}(r')$, $x_n \in \text{last}(r')$, $\forall i : 1 \leq i < n$, $x_{i+1} \in \text{follow}(r', x_i)$.

Proof.

"Only if" direction is trivial.
"If" direction: Induction on the structure of r'.

- $r' = \varepsilon$ or $r' = a_i$: trivial,
- $r' = r'_1 + r'_2$:
  $\text{first}(r') = \text{first}(r'_1) \cup \text{first}(r'_2)$, similarly for $\text{last}(r')$,
  $\text{follow}(r', a_i) = \text{follow}(r'_1, a_i)$ or $\text{follow}(r', a_i) = \text{follow}(r'_2, a_i)$ (since $\text{sym}(r'_1) \cap \text{sym}(r'_2) = \varnothing$),
  Suppose $x_1 \dots x_n$ satisfies the three conditions for r'.
  Because $\text{sym}(r'_1) \cap \text{sym}(r'_2) = \varnothing$, we deduce that $x_1 \dots x_n$ satisfies the three conditions for either r'_1 or r'_2.
  By induction hypothesis, $x_1 \dots x_n \in L(r'_1) \cup L(r'_2) = L(r')$.

From DRE to DFA

Proposition. Let r be a regular expression and r' be the marking of r. Then $\forall w = x_1 \dots x_n \in (\Sigma')^+$, $w \in L(r')$ iff w satisfies the following three conditions,

$x_1 \in \text{first}(r')$, $x_n \in \text{last}(r')$, $\forall i : 1 \leq i < n$, $x_{i+1} \in \text{follow}(r', x_i)$.

Proof.

"Only if" direction is trivial.

"If" direction: Induction on the structure of r'.

- $r' = \varepsilon$ or $r' = a_i$: trivial,
- $r' = r'_1 r'_2$:

$$\text{first}(r') = \begin{cases} \text{first}(r'_1) \cup \text{first}(r'_2) & \text{if } \varepsilon \in L(r'_1) \\ \text{first}(r'_1) & \text{otherwise} \end{cases}, \text{ similarly for last}(r'),$$

$$\text{follow}(r', a_i) = \begin{cases} \text{follow}(r'_1, a_i) & \text{if } a_i \in \text{sym}(r'_1) \setminus \text{last}(r'_1) \\ \text{follow}(r'_1, a_i) \cup \text{first}(r'_2) & \text{if } a_i \in \text{last}(r'_1) \\ \text{follow}(r'_1, a_i) & \text{if } a_i \in \text{sym}(r'_2) \end{cases}$$

We exemplify the proof by considering the case $x_1 \in \text{first}(r'_1), x_n \in \text{last}(r'_2)$. Then $\exists i < n.\ x_i \in \text{last}(r'_1)$ and $x_{i+1} \in \text{first}(r'_2)$, it follows from IH that $x_1 \dots x_i \in L(r'_1)$, $x_{i+1} \dots x_n \in L(r'_2)$, so $w \in L(r'_1)L(r'_2)$.

From DRE to DFA

Proposition. Let r be a regular expression and r' be the marking of r. Then $\forall w = x_1 \dots x_n \in (\Sigma')^+$, $w \in L(r')$ iff w satisfies the following three conditions,

$$x_1 \in \text{first}(r'),\ x_n \in \text{last}(r'),\ \forall i : 1 \leq i < n,\ x_{i+1} \in \text{follow}(r', x_i).$$

Proof.

"Only if" direction is trivial.
"If" direction: Induction on the structure of r'.

- $r' = \varepsilon$ or $r' = a_i$: trivial,
- $r' = (r'_1)^*$: Similar to the case $r' = r'_1 r'_2$.

□

From DRE to DFA

Proposition. Let r be a regular expression and r' be the marking of r. Then $\forall w = x_1 \dots x_n \in (\Sigma')^+$, $w \in L(r')$ iff w satisfies the following three conditions,

$$x_1 \in \text{first}(r'),\ x_n \in \text{last}(r'),\ \forall i : 1 \leq i < n,\ x_{i+1} \in \text{follow}(r', x_i).$$

Corollary 1. Let r be a regular expression and r' be the marking of r. Then $uav, u'av' \in L(r')$ implies $uav' \in L(r')$.

Corollary 2. Let r be a regular expression and r' be the marking of r. Then $L(r)$ is defined by the Glushkov automaton $\mathcal{A}_r = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = \text{sym}(r') \cup \{\varepsilon\}$, $q_0 = \varepsilon$,
- if $\varepsilon \in L(r')$, then $F = \text{last}(r') \cup \{\varepsilon\}$, otherwise $F = \text{last}(r')$,
- for every $x, y \in \text{sym}(r'), a \in \Sigma$,
  $(\varepsilon, a, x) \in \delta$ iff $\exists i \in \mathbb{N}.\ x = a_i$ and $x \in \text{first}(r')$,
  $(x, a, y) \in \delta$ iff $\exists i \in \mathbb{N}.\ y = a_i$ and $y \in \text{follow}(r', x)$.

From DRE to DFA

Corollary 2. Let r be a regular expression and r' be the marking of r. Then $L(r)$ is defined by the Glushkov automaton $\mathcal{A}_r = (Q, \Sigma, \delta, q_0, F)$, where

- $Q = \mathrm{sym}(r') \cup \{\varepsilon\}$, $q_0 = \varepsilon$,
- if $\varepsilon \in L(r')$, then $F = \mathrm{last}(r') \cup \{\varepsilon\}$, otherwise $F = \mathrm{last}(r')$,
- for every $x, y \in \mathrm{sym}(r'), a \in \Sigma$,
  $(\varepsilon, a, x) \in \delta$ iff $\exists i \in \mathbb{N}.\ x = a_i$ and $x \in \mathrm{first}(r')$,
  $(x, a, y) \in \delta$ iff $\exists i \in \mathbb{N}.\ y = a_i$ and $y \in \mathrm{follow}(r', x)$.

Theorem. Let r be a regular expression. Then r is a DRE iff $\mathcal{A}_r$ is a DFA.

Lemma. Let r be a regular expression. Then r is deterministic iff the following two conditions hold,

1. $\forall a_i, b_j \in \mathrm{first}(r')$, $a_i \neq b_j$ implies $a \neq b$,
2. $\forall a_i \in \mathrm{sym}(r')$ and $\forall b_j, c_k \in \mathrm{follow}(r', a_i)$, $b_j \neq c_k$ implies $b \neq c$.

Decision problem for (deterministic) DTDs

Theorem. The membership problem, the emptiness problem for DTDs are solvable in polynomial time.

Proof.

The membership and emptiness problem for NFHA(NFA) can be solved in polynomial time. □

Decision problem for (deterministic) DTDs

Theorem. The complexity of the inclusion problem:

- PSPACE-complete for DTDs,
- in polynomial time for deterministic DTDs.

Proof.

Let $D_1 = (\Sigma, s, \delta_1)$ and $D_2 = (\Sigma, s, \delta_2)$ be two (deterministic) DTDs.
W.l.o.g. assume that for every $a \in \Sigma$, $\exists$ a tree $t = (D', L)$ s.t.

$t \vDash D$, $L(\varepsilon) = s$ and $L(x) = a$ for some node x in t.

Otherwise, redundant symbols can be identified and removed in polynomial time.
Then $L(D_1) \subseteq L(D_2)$ iff

$$\forall a \in \Sigma, \textit{ let } a \to r_1 \in \delta_1 \textit{ and } a \to r_2 \in \delta_2, \textit{ then } L(r_1) \subseteq L(r_2).$$

For DTD:
Language inclusions for REs (NFAs) is PSPACE-complete.
For deterministic DTD:
Language inclusions for DREs (DFAs) is in PTIME. □

Outline

Extended DTD

Motivation:

The expressive power of DTD is strictly less than that of NFHA.

DTD only define the local languages:
$C(a(t_1,\ldots,t_n)), C'(a(t'_1,\ldots,t'_n)) \in L$ *implies* $C(a(t'_1,\ldots,t'_n)) \in L$.

Extended DTD:

$\forall$ alphabet Σ, let $\hat{\Sigma}$ be another alphabet s.t. $\exists$ a surjective mapping $\pi : \hat{\Sigma} \to \Sigma$.

For every $a \in \Sigma$, the letters $b \in \hat{\Sigma}$ s.t. $\pi(b) = a$ are called the types of a.

An *extended DTD (EDTD)* a pair (D, π), where D is a DTD over $\hat{\Sigma}$. A tree $t \in U_\Sigma$ satisfies an EDTD D if

there is an assignment of types to the labels of t s.t.
the resulting tree over $\hat{\Sigma}$ satisfies D (as a DTD over $\hat{\Sigma}$).

Example: $\{a(b(c,d)), a'(b(d,c))\}$ is defined by the following EDTD

$a \to b^{(1)}, a' \to b^{(2)}, b^{(1)} \to cd,\ b^{(2)} \to dc, c \to \varepsilon, d \to \varepsilon.$

Extended DTD: Expressibility and complexity

Theorem. Fix the root label, say s, of unranked trees, then EDTD $\equiv$ NFHA.

Proof.

Let (D, π) (where $D = (\hat{\Sigma}, \hat{s}, \hat{\delta})$) be an EDTD over the alphabet Σ.
Then $\pi(L(D))$ is defined by $\mathcal{A}'_D = (\hat{\Sigma}, \Sigma, \delta, \{\hat{s}\})$ s.t. $\forall b \to r \in \hat{\delta}$, $\pi(b)(r) \to b \in \delta$.

On the other hand, Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a NFHA. W.l.o.g. assume $F = \{q_f\}$.
Suppose all the horizontal lang. of $\mathcal{A}$ are given by regular expressions (over Q).
Then $L(\mathcal{A})$ can be defined by the EDTD $D = (Q \times \Sigma, (q_f, s), \hat{\delta})$ with
$\pi((q, a)) = a$, where

> *for every* $a(r) \to q \in \delta$, *we have* $(q, a) \to r' \in \hat{\delta}$,
> *where* r' *is obtained from* r *by replacing each* q' *with*
> $(q', b_1) + (q', b_2) + \cdots + (q', b_m)$ *(Suppose* $\Sigma = \{b_1, \ldots, b_m\}$*).* □

Theorem. The membership and emptiness problem of EDTD can be solved in polynomial time.

Theorem. The inclusion problem for EDTD is EXPTIME-complete.

Remark. Both results follow from those of NFHA(NFA).

Outline

Outline

Conditional XPath (CXPath)

Syntax:

Node formulas: $\alpha := a \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle\beta\rangle$,
Path formulas: $\beta := [\alpha] \mid \text{axis} \mid \text{axis}^* \mid (\text{axis}[\alpha])^* \mid \beta_1/\beta_2 \mid \beta_1 \vee \beta_2$.
where $\text{axis} \in \{\downarrow, \rightarrow, \uparrow, \leftarrow\}$.

Semantics:

Given a tree $t = (D, L)$, the CXPath formulas are interpreted as follows.

$$[\![a]\!]_t = \{x \in D \mid L(x) = a\}$$
$$[\![\neg\alpha]\!]_t = D \setminus [\![\alpha]\!]_t$$
$$[\![\alpha_1 \vee \alpha_2]\!]_t = [\![\alpha_1]\!]_t \cup [\![\alpha_2]\!]_t$$
$$[\![\langle\beta\rangle]\!]_t = \{x \in D \mid \exists x' \in D.(x, x') \in [\![\beta]\!]_t\}$$

$$[\![[\alpha]]\!]_t = \{(x, x) \mid x \in [\![\alpha]\!]_t\}$$
$$[\![\text{axis}]\!]_t = \{(x, x') \mid (x, x') \in \text{axis}\}$$
$$[\![\text{axis}^*]\!]_t = ([\![\text{axis}]\!]_t)^*$$
$$[\![(\text{axis}[\alpha])^*]\!]_t = ([\![(\text{axis}/[\alpha])]\!]_t)^*$$
$$[\![\beta_1/\beta_2]\!]_t = [\![\beta_1]\!]_t \circ [\![\beta_2]\!]_t$$
$$[\![\beta_1 \vee \beta_2]\!]_t = [\![\beta_1]\!]_t \cup [\![\beta_2]\!]_t$$

Abbreviations:

- $root = \langle\neg \uparrow true\rangle$, $lmSibling = \langle\neg \leftarrow true\rangle$,
- $\text{axis}^+ = \text{axis}/\text{axis}^*$, $/\beta = [root]/\beta$, $//\beta = [root]/ \downarrow^* /\beta$,
- $\beta_1//\beta_2 = \beta_1/ \downarrow^* /\beta_2$, ...

Conditional XPath ≡ FO

First-order logic over (ordered) unranked trees,

Syntax:

$$\varphi := P_a(x) \mid x = y \mid x < y \mid x \prec y \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x\varphi_1$$

Semantics:

Given a tree $t = (D, L)$ and $I : Var \to D$,

- $(t, I) \vDash x < y$ iff $I(x)$ is a strict prefix of $I(y)$ ($D \subseteq \mathbb{N}^*$ is a tree domain),
- $(t, I) \vDash x \prec y$ iff $I(x) = ui$ and $I(y) = uj$ s.t. $i < j$.

Abbreviations:

- $root(x) := \neg\exists y(y < x)$,
- $child(x, y) = x < y \wedge \neg\exists z(x < z \wedge z < y)$,
- $leftMost(x) = \neg\exists y(y \prec x)$,
- $nextSibling(x, y) = x \prec y \wedge \neg\exists z(x \prec z \wedge z \prec y)$.

Example:

$\exists$ a vertical path from x to some descendant of x, say y, labeled by b s.t. all the intermediate nodes (except x, y) are labeled by a.

$$\varphi(x) = \exists y(P_b(y) \wedge x < y \wedge \forall z(x < z \wedge z < y \to P_a(x))))$$

Conditional XPath ≡ FO

Theorem. Node expressions of CXPath ≡ FO formulas with one free variable.

Proof sketch.

Easy direction: From CXPath to FO,
$Tr : \alpha \to Tr(\alpha)(x), \beta \to Tr(\beta)(x, y)$,

- $Tr(a) = P_a(x)$, $Tr(\neg\alpha) = \neg Tr(\alpha)(x)$, $Tr(\alpha_1 \vee \alpha_2) = Tr(\alpha_1) \vee Tr(\alpha_2)$,
- $Tr(\langle\beta\rangle) = \exists y Tr(\beta)(x, y)$,
- $Tr([\alpha]) = Tr(\alpha)(x) \wedge x = y$, $Tr(\text{axis}) = \text{axis}(x, y)$,
- $Tr(\downarrow^*) = x \leq y$, $Tr(\rightarrow^*) = x \preceq y$, $Tr(\uparrow^*) = y \leq x$, $Tr(\leftarrow^*) = y \preceq x$,
- $Tr(\beta_1 \vee \beta_2) = Tr(\beta_1) \vee Tr(\beta_2)$,
- $Tr(\beta_1/\beta_2) = \exists z(Tr(\beta_1)[y \leftarrow z] \wedge Tr(\beta_2)[x \leftarrow z])$,
- $Tr((\downarrow[\alpha])^*) = x = y \vee (x < y \wedge \forall z(x < z \leq y \to Tr(\alpha)[x \leftarrow z]))$

The hard direction: From FO to CXPath
Reference: Maarten Marx, Conditional XPath, PODS 2004. □

Outline

Query automata

A *query automaton* (QA) $\mathcal{Q}$ is a tuple $(Q, \Sigma, F, \delta, Q_o)$, where

- (Q, Σ, F, δ) is a NFHA,
- $Q_o \subseteq Q$ (the set of selecting states).

Runs and accepting runs of $\mathcal{Q}$ over a tree t:

Runs and accepting runs of the NFHA (Q, Σ, F, δ) over t.

Unary query defined by QA

Let $\mathcal{Q} = (Q, \Sigma, F, \delta, Q_o)$ be a QA.
Let ρ be a run of $\mathcal{Q}$ over t, define $O_\rho(t) = \{x \in D \mid \rho(x) \in Q_o\}$.
The unary query defined by $\mathcal{Q}$ is defined as follows:

- Existential semantics: $\mathcal{Q}^{\exists}(t) = \bigcup_{\rho\text{: accepting run}} O_\rho(t)$,
- Universal semantics: $\mathcal{Q}^{\forall}(t) = \bigcap_{\rho\text{: accepting run}} O_\rho(t)$.

NFHA ≡ MSO sentences: A Supplement

MSO over unranked trees:

$$\varphi := P_a(x) \mid X(x) \mid x = y \mid x < y \mid x \prec y \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x\varphi_1 \mid \exists X\varphi_1$$

Abbreviations:

- $firstChild(x, y) = child(x, y) \wedge \neg\exists z(child(x, z) \wedge z < y)$,
- $lastChild(x, y) = child(x, y) \wedge \neg\exists z(child(x, z) \wedge y < z)$.

MSO syntax: Another equivalent definition

$$\varphi := \begin{array}{l} P_a(x) \mid X(x) \mid x = y \mid lastChild(x, y) \mid \\ nextSibling(x, y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x\varphi_1 \mid \exists X\varphi_1 \end{array}$$

The argument for the equivalence:

- $x < y =$
  $\forall X((X(x) \wedge \forall z_1 \forall z_2(X(z_1) \wedge nextSibling(z_1, z_2) \rightarrow X(z_2))) \rightarrow X(y))$
- $child(x, y) = \exists z(lastChild(x, z) \wedge (y = z \vee y < z))$
- $x < y = \forall X((X(x) \wedge \forall z_1 \forall z_2(X(z_1) \wedge child(z_1, z_2) \rightarrow X(z_2))) \rightarrow X(y))$

NFHA ≡ MSO sentences: A Supplement

Theorem. NFHA ≡ MSO sentences

Proof.

From NFHA to MSO sentences:

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a NFHA(NFA). Let $Q = \{q_1, \ldots, q_n\}$.

Suppose $\mathcal{A}$ is normalised:

$\forall (a, q) \in \Sigma$, let $R_{a,q}$ defined by $\mathcal{B}_{a,q} = (Q_{a,q}, Q, \delta_{a,q}, p_{(a,q),0}, F_{a,q})$.

Define a MSO formula $\varphi_{\mathcal{A}}$ as follows:

$\varphi_{\mathcal{A}} := \exists q_1 \ldots q_n \exists (p_{(a,q),1}, \ldots, p_{(a,q),n_{(a,q)}})_{(a,q)} \varphi_{acc} \wedge \varphi_{leaf} \wedge \varphi_{leftMost} \wedge \varphi_{trans}$,

$\varphi_{acc} := \exists x (root(x) \rightarrow \bigvee_{q \in F} q(x))$, $\varphi_{leaf}(x) := \forall x (leaf(x) \rightarrow \bigvee_{q : a(R_{a,q}) \rightarrow q, \varepsilon \in R_{a,q}} q(x))$,

$\varphi_{leftMost} :=$

$$\forall x \left(\begin{array}{c} (leftMost(x) \wedge \neg root(x)) \rightarrow \\ \exists y \left(child(y, x) \wedge \bigvee_{(a,q)} \left(P_a(y) \wedge q(y) \wedge \bigvee_{(p_{(a,q),0}, q', p) \in \delta_{a,q}} q'(x) \wedge p(x) \right) \right) \end{array} \right)$$

$\varphi_{trans} := \varphi_{vert} \wedge \varphi_{hor}$,

$\varphi_{vert} := \forall x \forall y (lastChild(x, y) \rightarrow \bigvee_{(a,q)} (P_a(x) \wedge q(x) \wedge \bigvee_{p \in F_{a,q}} p(y)))$,

$\varphi_{hor} = \forall x \forall y (nextSibling(x, y) \rightarrow$

$\exists z (child(z, y) \wedge \bigvee_{(a,q)} (P_a(z) \wedge q(z) \wedge \bigvee_{(p, q', p') \in \delta_{a,q}} (p(x) \wedge q'(y) \wedge p'(y)))).$ □

NFHA ≡ MSO sentences: A Supplement

Theorem. NFHA ≡ MSO sentences

Proof.

From MSO sentences to NFHA.

Normal form for MSO formulas.

$$\varphi := \begin{array}{l} Sing(X) \mid X \subseteq Y \mid X \subseteq P_a \mid P_a \subseteq X \mid lastChild(X,Y) \mid \\ nextSibling(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1 \end{array}$$

From MSO formulas in normal form to NFHA:

Similar to the construction from MSO to BUTA over ranked trees.

For instance, for $\varphi := lastChild(X,Y)$, $\mathcal{A}_\varphi = (Q, \Sigma \times \{0,1\}^2, \delta, F)$ is defined as follows:

- $Q = \{q_0, q_1, q_2\}, F = \{q_2\}$,
- δ includes the following rules for every $a \in \Sigma$:
  - $(a,0,0)((q_0)^*) \to q_0$,
  - $(a,0,0)((q_0)^* q_2 (q_0)^*) \to q_2$,
  - $(a,0,1)((q_0)^*) \to q_1$,
  - $(a,1,0)((q_0)^* q_1) \to q_2$.

QA ≡ MSO unary queries

MSO unary queries: MSO formulas with one free first-order variable.

Let $\varphi(x)$ be a MSO unary query and $t = (D, L)$ be a tree, then

the evaluation result of φ over t (denoted $[\![\varphi]\!]_t$) is $\{v \in D \mid t \vDash \varphi(v)\}$.

Theorem. QA with existential semantics ≡ MSO unary queries.

Proof.

From MSO to QA:

Let $\varphi(x)$ be a MSO unary query.
Define $\psi(X) = \forall x(X(x) \to \varphi(x))$.
From the equivalence of MSO ≡ NFHA,
an equivalent NFHA $\mathcal{A}_\psi = (Q, \Sigma \times \{0,1\}, \delta, F)$ can be constructed.
Construct $\mathcal{Q}_\psi = (Q \times \{0,1\}, \Sigma, \delta', F \times \{0,1\}, Q \times \{1\})$ as follows:

For every $(a,i)(r) \to q \in \delta$, let $a(r') \to (q,i) \in \delta'$ (where $i = 0,1$), where r' is obtained from r by replacing every q' with $(q',0) + (q',1)$.

Claim. $[\![\varphi]\!]_t = \bigcup\{U \mid \mathcal{A}_\psi \text{ accepts } (t,U)\} = \mathcal{Q}^{\exists}_\psi(t)$. □

QA ≡ MSO unary queries

MSO unary queries: MSO formulas with one free first-order variable.

Let $\varphi(x)$ be a MSO unary query and $t = (D, L)$ be a tree, then

the evaluation result of φ over t (denoted $[\![\varphi]\!]_t$) is $\{v \in D \mid t \vDash \varphi(v)\}$.

Theorem. QA with existential semantics ≡ MSO unary queries.

Proof.

From QA to MSO:

Let $\mathcal{Q} = (Q, \Sigma, \delta, F, Q_o)$ be a QA. Let $Q = \{q_1, \ldots, q_n\}$.
Suppose (Q, Σ, δ, F) is normalised:

$\forall (a, q) \in \Sigma$, let $R_{a,q}$ defined by $\mathcal{B}_{a,q} = (Q_{a,q}, Q, \delta_{a,q}, p_{(a,q),0}, F_{a,q})$.

Define a MSO formula $\varphi_{\mathcal{Q}}(x)$ as follows:

$\varphi_{\mathcal{Q}}(x) := \exists q_1 \ldots q_n \exists (p_{(a,q),1}, \ldots, p_{(a,q),n_{(a,q)}})_{a,q} (\varphi_{run} \wedge \bigvee_{q \in Q_o} q(x))$, where

$\varphi_{run} := \varphi_{acc} \wedge \varphi_{leaf} \wedge \varphi_{leftMost} \wedge \varphi_{trans}$ as in the poof of NFHA ≡ MSO.

Claim. $[\![\varphi_{\mathcal{Q}}]\!]_t = \bigcup_{\rho\text{: accepting run}} O_\rho = \mathcal{Q}^{\exists}(t)$. □

QA ≡ MSO unary queries

MSO unary queries: MSO formulas with one free first-order variable.

Let $\varphi(x)$ be a MSO unary query and $t = (D, L)$ be a tree, then

the evaluation result of φ over t (denoted $[\![\varphi]\!]_t$) is $\{v \in D \mid t \vDash \varphi(v)\}$.

Theorem. QA with existential semantics ≡ MSO unary queries.

Corollary. QA with existential semantics ≡ QA with universal semantics.

Proof.

Let $\mathcal{Q} = (Q, \Sigma, \delta, F, Q_o)$ be a QA.

From existential-QA ≡ MSO unary queries,

$\exists$ a QA $\mathcal{Q}' = (Q', \Sigma, \delta', F', Q'_o)$ s.t. $(\mathcal{Q}')^{\exists}(t) = [\![\neg\varphi_{\mathcal{Q}}]\!]_t$.

W.l.o.g. assume that the NFHA $(Q', \Sigma, \delta', F')$ is complete ($\exists$ a run for every tree).

Define $\mathcal{Q}'' = (Q', \Sigma, \delta', F', Q'/Q'_o)$. Then for every tree $t = (D, L)$,

$$\begin{aligned}(\mathcal{Q}'')^{\forall}(t) &= \{x \in D \mid \forall \text{ accepting run } \rho', \rho'(x) \notin Q'_o\} \\ &= D \setminus (\mathcal{Q}')^{\exists}(t) = D \setminus [\![\neg\varphi_{\mathcal{Q}}]\!]_t = [\![\varphi_{\mathcal{Q}}]\!]_t\end{aligned}.$$

□

Single-run query automaton

Single-run query automaton: Query automaton $\mathcal{Q}$ s.t.

> *for every* t *and every accepting run* ρ_1 *and* ρ_2 *of* $\mathcal{Q}$ *over* t,
> $S_{\rho_1}(t) = S_{\rho_2}(t)$.

Theorem. For every query automaton, there is an equivalent single-run query automaton.

Reference: Markus Frick, Martin Grohe, Christoph Koch, Query evaluation on compressed trees, LICS 2003.

Theorem. The nonemptiness of single-run query automata can be solved in polynomial time.

Outline

The closure of XPath formulas

Theorem. For every XPath node formula φ, an equivalent single-run query automaton $\mathcal{A}_\varphi$ can be constructed in exponential time.

Corollary. The satisfiability of XPath node formulas can be solved in exponential time (in fact, EXPTIME-complete).

Let φ be a XPath formula, then the closure of φ, denoted by $cl(\varphi)$, is defined as follows.

- Node formulas:
  - $\mathrm{cl}(a) = \{a, \neg a\}$, $\mathrm{cl}(\neg\alpha) = \{\neg\alpha\} \cup \mathrm{cl}(\alpha)$,
  - $\mathrm{cl}(\alpha_1 \vee \alpha_2) = \{\alpha_1 \vee \alpha_2, \neg(\alpha_1 \vee \alpha_2)\} \cup \mathrm{cl}(\alpha_1) \cup \mathrm{cl}(\alpha_2)$,
  - $\mathrm{cl}(\langle\beta\rangle) = \{\langle\beta\rangle, \neg\langle\beta\rangle\} \cup \mathrm{cl}(\beta)$.
- Path formulas:
  - $\mathrm{cl}([\alpha]) = \{[\alpha]\} \cup \mathrm{cl}(\alpha), \mathrm{cl}(\mathrm{axis}) = \{\mathrm{axis}\}$, $\mathrm{cl}(\mathrm{axis}^*) = \{\mathrm{axis}^*\}$,
  - $\mathrm{cl}((\mathrm{axis}[\alpha])^*) = \{(\mathrm{axis}[\alpha])^*\} \cup \mathrm{cl}(\alpha)$,
  - $\mathrm{cl}(\beta_1/\beta_2) = \{\beta_1/\beta_2\} \cup \mathrm{cl}(\beta_1) \cup \mathrm{cl}(\beta_2)$,
  - $\mathrm{cl}(\beta_1 \vee \beta_2) = \{\beta_1 \vee \beta_2\} \cup \mathrm{cl}(\beta_1) \cup \mathrm{cl}(\beta_2)$.

An illustration of the intuition

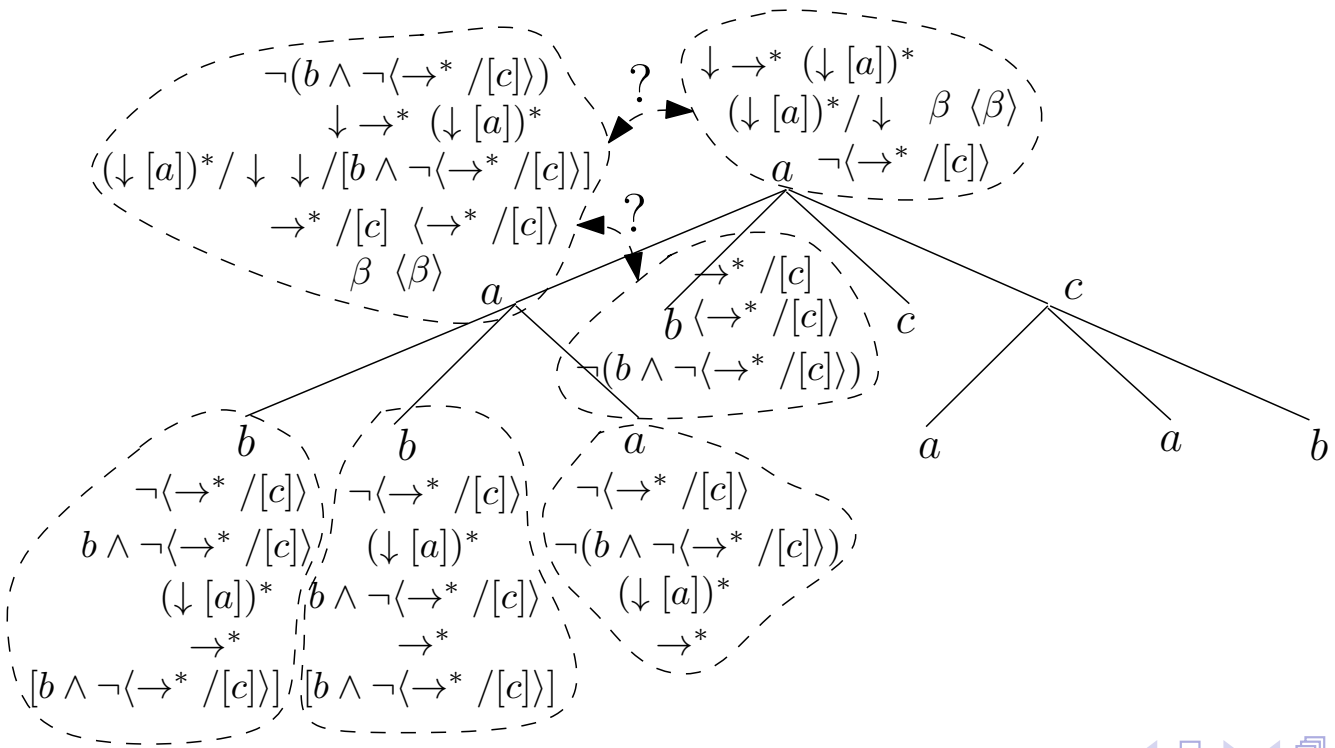
For a XPath node formula φ,

Construct a QA to attach each node x of a tree t with all the formulas $\alpha \in \mathrm{cl}(\varphi)$ s.t. $x \in [\![\alpha]\!]_t$ and $\beta \in \mathrm{cl}(\varphi)$ s.t. $\exists x'.(x, x') \in [\![\beta]\!]_t$.

$\varphi = \langle\beta\rangle = \langle(\downarrow [a])^*/ \downarrow /[b \wedge \neg\langle\rightarrow^* /[c]\rangle]\rangle$

$cl(\varphi)$

$a, b, c, \neg a, \neg b, \neg c, \langle\rightarrow^* /[c]\rangle, \neg\langle\rightarrow^* /[c]\rangle$
$b \wedge \neg\langle\rightarrow^* /[c]\rangle, \neg(b \wedge \neg\langle\rightarrow^* /[c]\rangle), \varphi, \neg\varphi$
$\downarrow, (\downarrow [a])^*, (\downarrow [a])^*/ \downarrow, \rightarrow^*, [c], \rightarrow^* /[c], [b \wedge \neg\langle\rightarrow^* /[c]\rangle], \beta$



Elementary set of formulas

Negation normal form (NNF) of CXPath formulas:

All the negations are before a or $\langle\beta\rangle$ formulas.

Remark. All the XPath formulas can be transformed into equivalent NNFs, by using $\wedge$.

Let φ be a XPath formula in NNF. A set $\Phi \subseteq cl(\varphi)$ is said to be *elementary* iff the following conditions hold,

- there is $a \in \Sigma$ s.t. $a \in \Phi$ and $\forall b \in \Sigma(b \neq a \rightarrow \neg b \in \Phi)$,
- for every $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \mathrm{cl}(\varphi)$, $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \Phi$ iff $(\alpha_1 \in \Phi \underset{\text{and}}{\overset{\text{or}}{}} \alpha_2 \in \Phi)$,
- for every $\psi = a$ or $\langle\beta\rangle \in \mathrm{cl}(\varphi)$, $\psi \in \Phi$ iff $\neg\psi \notin \Phi$,

Elementary set of formulas

Negation normal form (NNF) of CXPath formulas:

All the negations are before a or $\langle\beta\rangle$ formulas.

Remark. All the XPath formulas can be transformed into equivalent NNFs, by using $\wedge$.

Let φ be a XPath formula in NNF. A set $\Phi \subseteq cl(\varphi)$ is said to be *elementary* iff the following conditions hold,

- there is $a \in \Sigma$ s.t. $a \in \Phi$ and $\forall b \in \Sigma(b \neq a \rightarrow \neg b \in \Phi)$,
- for every $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \mathrm{cl}(\varphi)$, $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \Phi$ iff $(\alpha_1 \in \Phi \underset{\text{and}}{\overset{\text{or}}{}} \alpha_2 \in \Phi)$,
- for every $\psi = a$ or $\langle\beta\rangle \in \mathrm{cl}(\varphi)$, $\psi \in \Phi$ iff $\neg\psi \notin \Phi$,
- for every $\langle\beta\rangle \in \mathrm{cl}(\varphi)$, $\langle\beta\rangle \in \Phi$ iff $\beta \in \Phi$,
- for every $[\alpha] \in \mathrm{cl}(\varphi)$, $[\alpha] \in \Phi$ iff $\alpha \in \Phi$,
- for every $\mathrm{axis}^*, (\mathrm{axis}[\alpha])^* \in \mathrm{cl}(\varphi)$, $\mathrm{axis}^*, (\mathrm{axis}[\alpha])^* \in \Phi$,

Elementary set of formulas

Negation normal form (NNF) of CXPath formulas:

All the negations are before a or $\langle\beta\rangle$ formulas.

Remark. All the XPath formulas can be transformed into equivalent NNFs, by using $\wedge$.

Let φ be a XPath formula in NNF. A set $\Phi \subseteq cl(\varphi)$ is said to be *elementary* iff the following conditions hold,

- there is $a \in \Sigma$ s.t. $a \in \Phi$ and $\forall b \in \Sigma(b \neq a \rightarrow \neg b \in \Phi)$,
- for every $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \mathrm{cl}(\varphi)$, $\alpha_1 \overset{\vee}{\underset{\wedge}{}} \alpha_2 \in \Phi$ iff $(\alpha_1 \in \Phi \underset{\text{and}}{\overset{\text{or}}{}} \alpha_2 \in \Phi)$,
- for every $\psi = a$ or $\langle\beta\rangle \in \mathrm{cl}(\varphi)$, $\psi \in \Phi$ iff $\neg\psi \notin \Phi$,
- for every $\langle\beta\rangle \in \mathrm{cl}(\varphi)$, $\langle\beta\rangle \in \Phi$ iff $\beta \in \Phi$,
- for every $[\alpha] \in \mathrm{cl}(\varphi)$, $[\alpha] \in \Phi$ iff $\alpha \in \Phi$,
- for every $\mathrm{axis}^*, (\mathrm{axis}[\alpha])^* \in \mathrm{cl}(\varphi)$, $\mathrm{axis}^*, (\mathrm{axis}[\alpha])^* \in \Phi$,
- for every $\beta_1 \vee \beta_2 \in \mathrm{cl}(\varphi)$, $\beta_1 \vee \beta_2 \in \Phi$ iff $\beta_1 \in \Phi$ or $\beta_2 \in \Phi$,
- for every $\beta_1/\beta_2 \in \mathrm{cl}(\varphi)$, if β_1 does not contain any occurrence of $\{\text{“}\downarrow\text{”}, \text{“}\rightarrow\text{”}, \text{“}\uparrow\text{”}, \text{“}\downarrow\text{”}\}$, then $\beta_1/\beta_2 \in \Phi$ iff $\beta_1 \in \Phi$ and $\beta_2 \in \Phi$,
- for every $\beta_1/\beta_2 \in \mathrm{cl}(\varphi)$, if $\beta_1 = \mathrm{axis}^*$ or $\beta_1 = (\mathrm{axis}[\alpha])^*$, then $\beta_2 \in \Phi$ implies $\beta_1/\beta_2 \in \Phi$.

The construction

Construct a query automaton $\mathcal{A}_\varphi = (Q, \Sigma, \delta, F, Q_o)$ as follows:

- Q: the set of elementary sets of formulas,
- F: the set of elementary sets of formulas Φ s.t.
  - Φ does not contain formulas of the form axis or axis$/\beta$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$,
  - if axis$^*/\beta \in \Phi$ or $(\text{axis}[\alpha])^*/\beta \in \Phi$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$, then $\beta \in \Phi$,
- Q_o: the set of elementary sets of formulas Φ s.t. $\varphi \in \Phi$,
- δ is the set of all the rules of the form $a(L_{a,\Phi}) \rightarrow \Phi$, where $a \in \Phi$ and $L_{a,\Phi} = L_{hor} \cap L_{up}(\Phi) \cap L_{down}(a, \Phi)$,
  - L_{hor} is the set of strings $\Phi_0 \dots \Phi_n$ satisfying the following conditions,
    - Φ_0 (resp. Φ_n) does not contain formulas of the form $\leftarrow/\beta$ (resp. $\rightarrow/\beta$);
    - if $(\leftarrow[\alpha])^*/\beta \in \Phi_0$ (resp. $(\rightarrow[\alpha])^*/\beta \in \Phi_n$), then $\beta \in \Phi_0$ (resp. $\beta \in \Phi_n$) (remark: $\leftarrow^*/\beta, \rightarrow^*/\beta : \alpha = true$);
    - $\forall i : 0 \le i < n$ (resp. $i : 0 < i \le n$), if $\rightarrow \in \text{cl}(\varphi)$ (resp. $\leftarrow \in \text{cl}(\varphi)$), then $\rightarrow \in \Phi_i$ (resp. $\leftarrow \in \Phi_i$);
    - $\forall i : 0 \le i < n$, $\rightarrow/\beta \in \Phi_i$ iff $\beta \in \Phi_{i+1}$;
    - $\forall i : 0 < i \le n$, $\leftarrow/\beta \in \Phi_i$ iff $\beta \in \Phi_{i-1}$;
    - $\forall i : 0 \le i < n$, if $(\rightarrow[\alpha])^*/\beta \in \text{cl}(\phi)$, then $(\rightarrow[\alpha])^*/\beta \in \Phi_i$ iff either $\beta \in \Phi_i$ or $\alpha \in \Phi_{i+1}$ and $(\rightarrow[\alpha])^*/\beta \in \Phi_{i+1}$ (remark: $\rightarrow^*/\beta$: $\alpha = true$);
    - $\forall i : 0 < i \le n$, if $(\leftarrow[\alpha])^*/\beta \in \text{cl}(\phi)$, then $(\leftarrow[\alpha])^*/\beta \in \Phi_i$ iff either $\beta \in \Phi_i$ or $\alpha \in \Phi_{i-1}$ and $(\leftarrow[\alpha])^*/\beta \in \Phi_{i-1}$ (remark: $\leftarrow^*/\beta$: $\alpha = true$).

The construction

Construct a query automaton $\mathcal{A}_\varphi = (Q, \Sigma, \delta, F, Q_o)$ as follows:

- Q: the set of elementary sets of formulas,
- F: the set of elementary sets of formulas Φ s.t.
  - Φ does not contain formulas of the form axis or axis$/\beta$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$,
  - if axis$^*/\beta \in \Phi$ or $(\text{axis}[\alpha])^*/\beta \in \Phi$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$, then $\beta \in \Phi$,
- Q_o: the set of elementary sets of formulas Φ s.t. $\varphi \in \Phi$,
- δ is the set of all the rules of the form $a(L_{a,\Phi}) \rightarrow \Phi$, where $a \in \Phi$ and $L_{a,\Phi} = L_{hor} \cap L_{up}(\Phi) \cap L_{down}(a, \Phi)$,
  - $L_{up}(\Phi)$ is the set of strings $\Phi_0 \ldots \Phi_n$ satisfying the following conditions,
    - if $\uparrow \in \text{cl}(\varphi)$, then $\forall i.\ \uparrow \in \Phi_i$,
    - for every $\uparrow /\beta \in \text{cl}(\varphi)$, then $\beta \in \Phi$ iff for every $i : 0 \leq i \leq n$, $\uparrow /\beta \in \Phi_i$,
    - for every $(\uparrow [\alpha])^*/\beta \in \text{cl}(\phi)$, $\forall i.\ (\uparrow [\alpha])^*/\beta \in \Phi_i$ iff either $\beta \in \Phi_i$ or $\alpha \in \Phi$ and $(\uparrow [\alpha])^*/\beta \in \Phi$ (remark $\uparrow^* /\beta : \alpha = true$).

The construction

Construct a query automaton $\mathcal{A}_\varphi = (Q, \Sigma, \delta, F, Q_o)$ as follows:

- Q: the set of elementary sets of formulas,
- F: the set of elementary sets of formulas Φ s.t.
  - Φ does not contain formulas of the form axis or axis$/\beta$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$,
  - if axis$^*/\beta \in \Phi$ or $(\text{axis}[\alpha])^*/\beta \in \Phi$ s.t. axis $\in \{\uparrow, \leftarrow, \rightarrow\}$, then $\beta \in \Phi$,
- Q_o: the set of elementary sets of formulas Φ s.t. $\varphi \in \Phi$,
- δ is the set of all the rules of the form $a(L_{a,\Phi}) \rightarrow \Phi$, where $a \in \Phi$ and $L_{a,\Phi} = L_{hor} \cap L_{up}(\Phi) \cap L_{down}(a, \Phi)$,
  - $L_{down}(a, \Phi)$:
    if $\downarrow \in \text{cl}(\varphi) \setminus \Phi$, then $L_{down}(a, \Phi) = \{\varepsilon\}$;
    if $\downarrow \in \Phi$, then $L_{down}(a, \Phi)$ is the set of strings $\Phi_0 \dots \Phi_n$ $(n > 0)$ s.t.
    - for every $\downarrow/\beta \in \text{cl}(\varphi)$, $\downarrow/\beta \in \Phi$ iff $\exists i : 0 \le i \le n$, $\beta \in \Phi_i$,
    - for every $(\downarrow[\alpha])^*/\beta \in \text{cl}(\phi)$, $(\downarrow[\alpha])^*/\beta \in \Phi$ iff either $\beta \in \Phi$ or $\exists i : 0 \le i \le n$ s.t. $\alpha \in \Phi_i$ and $(\downarrow[\alpha])^*/\beta \in \Phi_i$ (remark: $\downarrow^*/\beta : \alpha = true$)

    if $\downarrow \notin \text{cl}(\varphi)$, then $L_{down}(a, \Phi)$ is the union of $\{\varepsilon\}$ and the set of strings $\Phi_0 \dots \Phi_n$ $(n > 0)$ ...

References

- Anne Bruggemmann-Klein, Derick Wood, One-unambiguous regular languages, Information and computation, 140, 229-253, 1998.
- M. Frick, M. Grohe, C. Koch. Query evaluation on compressed trees, LICS 2003.
- Nadime Francis, Claire David, Leonid Libkin, A direct translation from XPath to Nondeterministic Automata, AMW 2011.

Next Lecture

Up to you !