Efficient Inclusion Checking for Deterministic Tree Automata and DTDs

Jérôme Champavère
Joint work with R. Gilleron, A. Lemay and J. Niehren

Université de Lille, INRIA, Mostrare

LATA 2008, Tarragona, España
March 14, 2008

Tree Automata Inclusion

- Tree automata A and B
- Instance: $L(A) \subseteq L(B)$?
- Non deterministic B: DEXPTIME-complete [Seidl 1994]
- Deterministic B
  - B is complete: $O(|A| * |B|)$
  - Not complete: ?

Deterministic Tree Automata Inclusion

Usual algorithm

- Ranked automata, arity n
- Decide $L(A) \cap L(B)^c = \emptyset$

Complexity

- Running time: $O(|A| * |\Sigma| * |B|^n)$
- Involves computation of B complement
- Linear reduction to binary automata: $O(|A| * |\Sigma| * |B|^2)$
- Our goal: $O(|A| * |B|)$, independently of $|\Sigma|$

Inclusion into DTDs

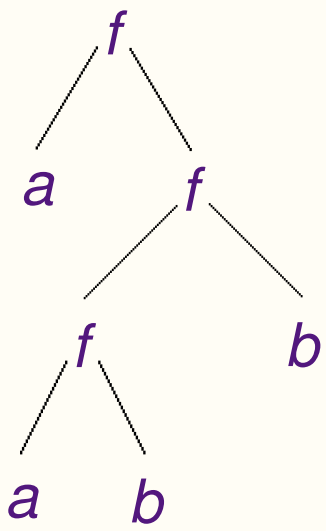
- $L(A)$: set of XML documents
  - Automata for unranked trees [TATA 2007]
  - Possibly non deterministic
- Representations for DTDs [Martens & Niehren 2007]
  - Hedge automata
  - Stepwise automata
- Algorithm for testing $L(A) \subseteq L(DTD)$

Outline

Tree Automata

Basic notions

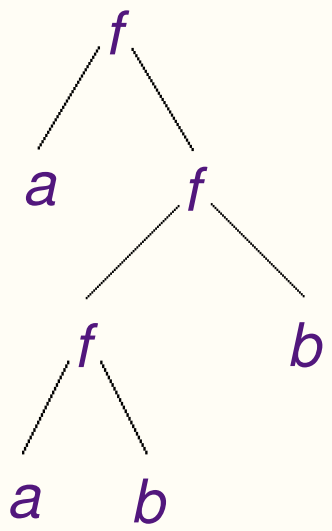
- Binary tree t:



Tree Automata

Basic notions

- Binary tree t:

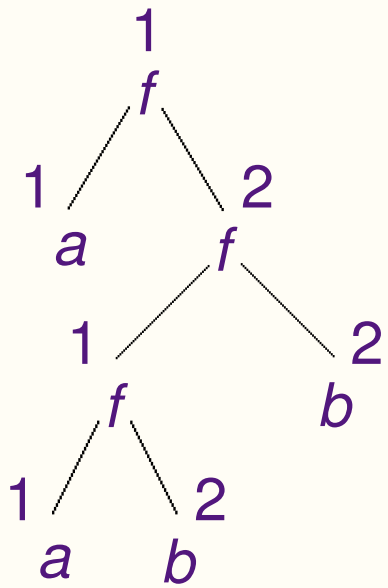


- Automaton A:
  - States: $\{1, 2\}$
  - Final states: $\{2\}$
  - Rules:$\{$
    $a \to 1,$
    $b \to 2,$
    $f(1, 2) \to 1,$
    $f(1, 2) \to 2 \quad \}$

Tree Automata

Basic notions

- Binary tree t:

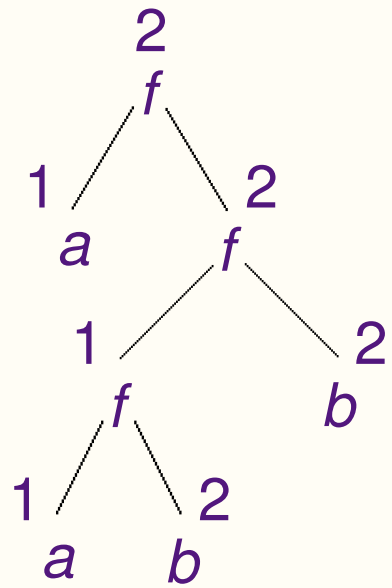


- Automaton A:
  - States: $\{1, 2\}$
  - Final states: $\{2\}$
  - Rules:$\{$
    $a \to 1,$
    $b \to 2,$
    $f(1, 2) \to 1,$
    $f(1, 2) \to 2$ $\}$

Tree Automata

Basic notions

- Binary tree t:



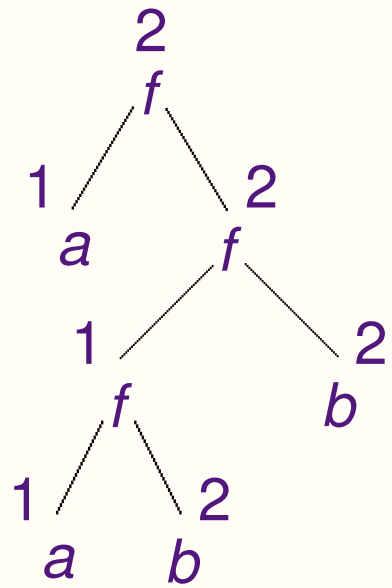
- $\text{eval}_A(t) = \{1, 2\}$

- Automaton A:
  - States: $\{1, 2\}$
  - Final states: $\{2\}$
  - Rules:$\{$
    $a \to 1,$
    $b \to 2,$
    $f(1, 2) \to 1,$
    $f(1, 2) \to 2$ $\}$

Tree Automata

Basic notions

- Binary tree t:



- $\text{eval}_A(t) = \{1, 2\}$

- Automaton A:
  - States: $\{1, 2\}$
  - Final states: $\{2\}$
  - Rules:$\{$
    $a \to 1,$
    $b \to 2,$
    $f(1, 2) \to 1,$
    $f(1, 2) \to 2$ $\}$

$$L(A) = \{t | \text{eval}_A(t) \cap \text{final}(A) \neq \emptyset\}$$

Emptiness/Accessibility

Emptiness

Input: A, Σ
Output: $L(A) = \emptyset$
Algorithm: does A have accessible final states?

Accessibility

$\mathrm{acc}(p)$ in A if $\exists t$ such that $p \in \mathrm{eval}_A(t)$.

Efficiency of Ground Datalog

Inferring a ground Datalog program for accessibility

$$\frac{a \to p \in \text{rules}(A)}{\text{acc}(p).} \qquad \frac{f(p_1, p_2) \to p \in \text{rules}(A)}{\text{acc}(p) :- \text{acc}(p_1), \text{acc}(p_2).}$$

Number of clauses: $O(|A|)$

Theorem (Dantsin et al. 2001)

The least fixed point $\text{lfp}(P)$ *of a ground Datalog program* P *can be computed in time* $O(|P|)$*, even for unbounded predicate arities.*

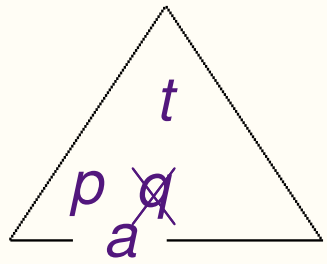
Guiding Ideas of Characterization

- Detection of inclusion failure $L(A) \not\subseteq L(B)$
- Find $t \in L(A)$ such that $t \notin L(B)$
- Accessibility on product $A \times B$
- Ground Datalog program $D_{A,B}$
- Conditions: B deterministic, A productive

Inclusion Failure

Constant rules

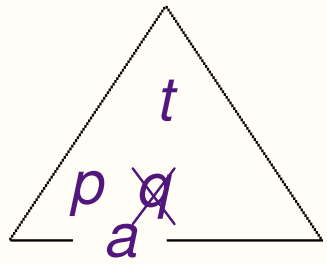
$$\frac{a \longrightarrow p \text{ in } A \qquad \nexists q.\ a \longrightarrow q \text{ in } B}{\texttt{fail.}}$$



Inclusion Failure

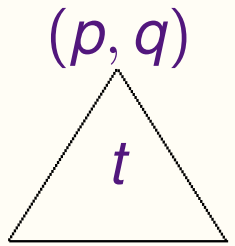
Constant rules

$$\frac{a \longrightarrow p \text{ in } A \quad \nexists q.\ a \longrightarrow q \text{ in } B}{\texttt{fail.}}$$



Final states

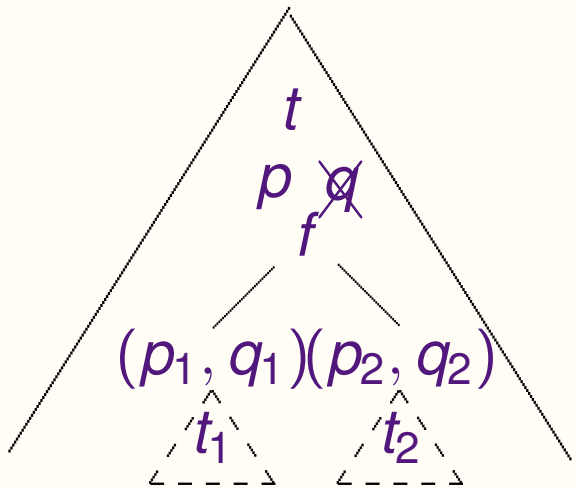
$$\frac{p \text{ final in } A \quad q \text{ not final in } B}{\texttt{fail} :\!-\ \text{acc}(p, q).}$$



Inclusion Failure (cont.)

Binary rules

$$\frac{f(p_1, p_2) \longrightarrow p \text{ in } A \qquad \nexists q.\ f(q_1, q_2) \longrightarrow q \text{ in } B}{\texttt{fail} :\!- \ \mathrm{acc}(p_1, q_1), \mathrm{acc}(p_2, q_2).}$$



Datalog Characterization

Proposition

For deterministic B and productive A, $L(A) \not\subseteq L(B)$ if and only if `fail` $\in \mathsf{lfp}(D_{A,B})$.

Datalog Characterization

Proposition

For deterministic B and productive A, $L(A) \not\subseteq L(B)$ if and only if $\texttt{fail} \in \mathrm{lfp}(D_{A,B})$.

Computing failures

- For constant rules and final states: $O(|A| * |B|)$
- For binary rules: $O(|A| * |B|^2)$

$$\frac{\begin{array}{rl} & f(p_1, p_2) \to p \text{ in } A \\ \nexists q. & f(q_1, q_2) \to q \text{ in } B \end{array}}{\texttt{fail} :- \mathrm{acc}(p_1, q_1), \mathrm{acc}(p_2, q_2).}$$

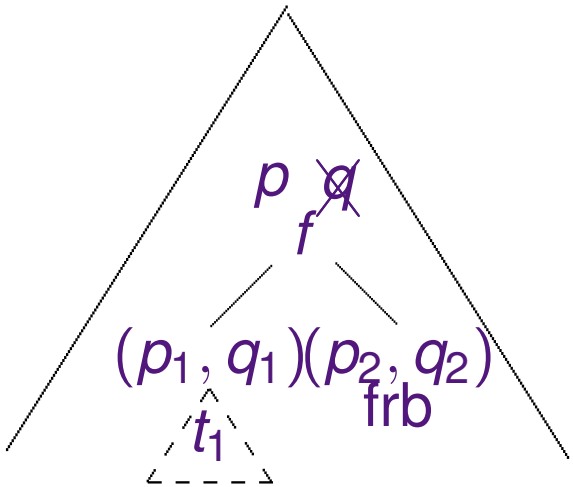
Forbidden States

$$\boxed{\mathrm{frb}(p,q) \Leftrightarrow (\mathrm{acc}(p,q) \Rightarrow L(A) \not\subseteq L(B))}$$

- Inferred Datalog clauses:

$$\frac{f(p_1,p_2) \to p \in \mathrm{rules}(A) \quad \nexists q.\ f(q_1,q_2) \to q \in \mathrm{rules}(B)}{\begin{array}{l}\mathrm{frb}(p_2,q_2) :- \mathrm{acc}(p_1,q_1).\\ \mathrm{frb}(p_1,q_1) :- \mathrm{acc}(p_2,q_2).\end{array}}$$

$$\frac{p \in \mathrm{states}(A) \quad q \in \mathrm{states}(B)}{\texttt{fail} :- \mathrm{acc}(p,q), \mathrm{frb}(p,q).}$$



- Number of clauses: $O(|A| * |B|^2)$

Grouping Forbidden States

For $Q \subseteq \text{states}(B)$:

$$\boxed{\text{frb}^c(p, Q) \Leftrightarrow \forall q \notin Q,\ \text{frb}(p, q)}$$

- Inferred Datalog clauses:

$$\frac{f(p_1, p_2) \to p \in \text{rules}(A) \qquad \left.\begin{array}{c} f(q_1, q_2^1) \to q^1 \in \text{rules}(B) \\ \vdots \\ f(q_1, q_2^n) \to q^n \in \text{rules}(B) \end{array}\right\} \text{all rules for } q_1}{\text{frb}^c(p_2, \{q_2^1, \ldots, q_2^n\}) :- \text{acc}(p_1, q_1).}$$

Grouping Forbidden States

For $Q \subseteq \text{states}(B)$:

$$\boxed{\text{frb}^c(p, Q) \Leftrightarrow \forall q \notin Q,\ \text{frb}(p, q)}$$

- Inferred Datalog clauses:

$$\frac{f(p_1, p_2) \to p \in \text{rules}(A) \qquad \left.\begin{array}{c} f(q_1, q_2^1) \to q^1 \in \text{rules}(B) \\ \vdots \\ f(q_1, q_2^n) \to q^n \in \text{rules}(B) \end{array}\right\} \text{all rules for } q_1}{\begin{array}{r} \text{frb}^c(p_2, \{q_2^1, \ldots, q_2^n\}) :- \text{acc}(p_1, q_1). \\ \text{acc}(p_1, q^1) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^1). \\ \vdots \\ \text{acc}(p_1, q^n) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^n). \end{array}}$$

Grouping Forbidden States

For $Q \subseteq \text{states}(B)$:

$$\boxed{\text{frb}^c(p, Q) \Leftrightarrow \forall q \notin Q,\ \text{frb}(p, q)}$$

- Inferred Datalog clauses:

$$f(p_1, p_2) \to p \in \text{rules}(A) \qquad \left.\begin{array}{c} f(q_1, q_2^1) \to q^1 \in \text{rules}(B) \\ \vdots \\ f(q_1, q_2^n) \to q^n \in \text{rules}(B) \end{array}\right\} \text{all rules for } q_1$$

$$\text{frb}^c(p_2, \{q_2^1, \ldots, q_2^n\}) :- \text{acc}(p_1, q_1). \qquad \text{1 clause of size } O(n)$$

$$\left.\begin{array}{l} \text{acc}(p_1, q^1) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^1). \\ \vdots \\ \text{acc}(p_1, q^n) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^n). \end{array}\right\} \quad n \text{ clauses of size } O(1)$$

Grouping Forbidden States

For $Q \subseteq \text{states}(B)$:

$$\boxed{\text{frb}^c(p, Q) \Leftrightarrow \forall q \notin Q,\ \text{frb}(p, q)}$$

- Inferred Datalog clauses:

$$f(p_1, p_2) \to p \in \text{rules}(A) \qquad \left.\begin{array}{c} f(q_1, q_2^1) \to q^1 \in \text{rules}(B) \\ \vdots \\ f(q_1, q_2^n) \to q^n \in \text{rules}(B) \end{array}\right\} \text{all rules for } q_1$$

$$\text{frb}^c(p_2, \{q_2^1, \ldots, q_2^n\}) :- \text{acc}(p_1, q_1). \qquad \text{1 clause of size } O(n)$$

$$\left.\begin{array}{l} \text{acc}(p_1, q^1) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^1). \\ \vdots \\ \text{acc}(p_1, q^n) :- \text{acc}(p_1, q_1), \text{acc}(p_2, q_2^n). \end{array}\right\} \quad n \text{ clauses of size } O(1)$$

- Size of the inferred Datalog program: $O(|A| * |B|)$

Testing Failure

Failure for binary rules

$$\frac{p \in \text{states}(A) \quad q \in \text{states}(B)}{\texttt{fail} :- \text{acc}(p, q), \text{frb}(p, q).}$$

Computing forbidden states

$$O(|A| * |B|) \left\{ \begin{array}{lcl} \text{frb}^c(p, Q_1) & \Rightarrow & \text{frb}(p, q_1),\ \forall q_1 \notin Q_1 \\ & \vdots & \\ \text{frb}^c(p, Q_m) & \Rightarrow & \text{frb}(p, q_m),\ \forall q_m \notin Q_m \end{array} \right\} O(|A| * |B|^2)$$

- Cannot enumerate frb predicates: need algorithmic solution

Efficient data structure

- Example: $\text{states}(B) = \{1, 2, 3, 4\}$

$$\left\{ \begin{array}{lcl} \text{frb}^c(p, \{1, 2\}) & \Rightarrow & \text{frb}(p, 3) \text{ and } \text{frb}(p, 4) \\ \text{frb}^c(p, \{2, 4\}) & \Rightarrow & \text{frb}(p, 1) \text{ and } \text{frb}(p, 3) \end{array} \right\}$$

- Array for state p:

q	1	2	3	4	$total$
$occurrence(q)$	1	2	0	1	2

- $\text{frb}(p, q)$ if $occurrence(q) < total$
- Overall time for testing failure: $O(|A| * |B|)$

Testing Inclusion

Theorem

$L(A) \subseteq L(B)$ can be tested in time $O(|A| * |B|)$ for binary tree automata A, deterministic B over Σ.

Outline

XML Trees

- Unranked trees
- Binary encoding *via* Currying
- Stepwise tree automata [Carme et al. 2004]

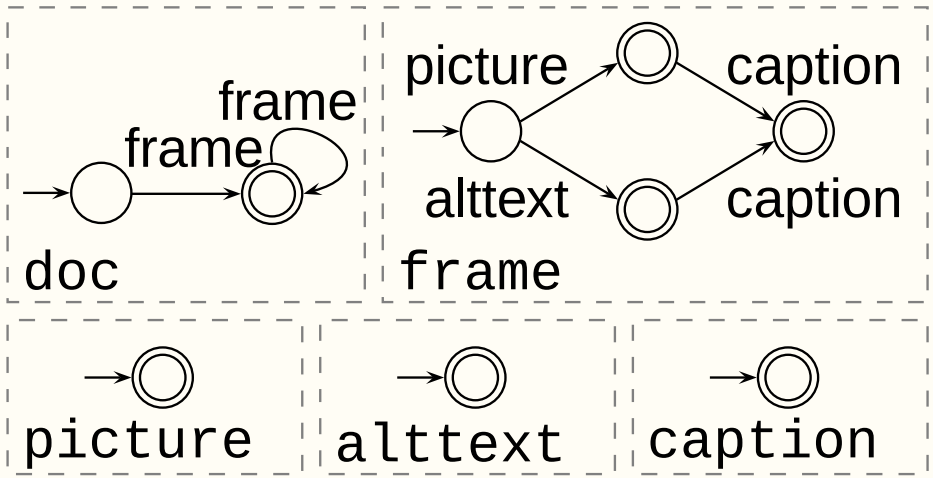
DTDs to Glushkov Automata

DTD

```
<!ELEMENT doc      (frame+)>
<!ELEMENT frame    (picture|alttext),caption?>
<!ELEMENT picture  (#PCDATA)>
<!ELEMENT alttext  (#PCDATA)>
<!ELEMENT caption  (#PCDATA)>
```

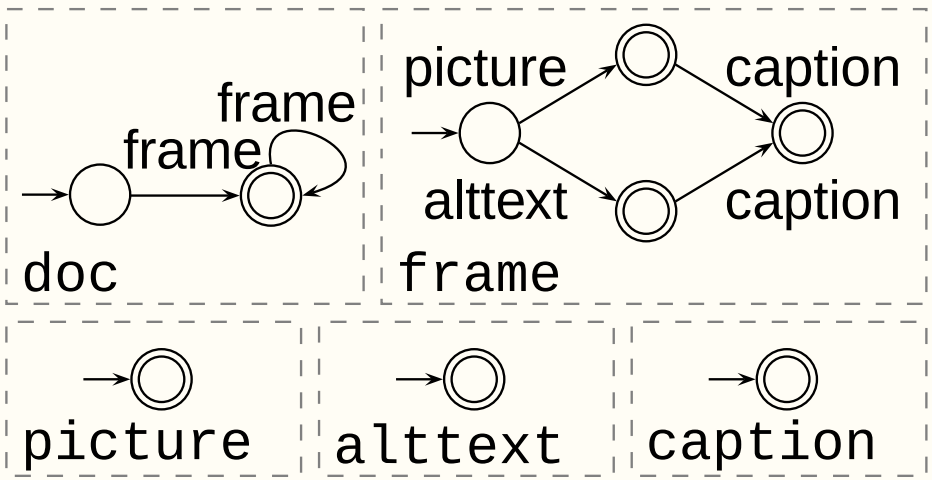
Glushkov automata

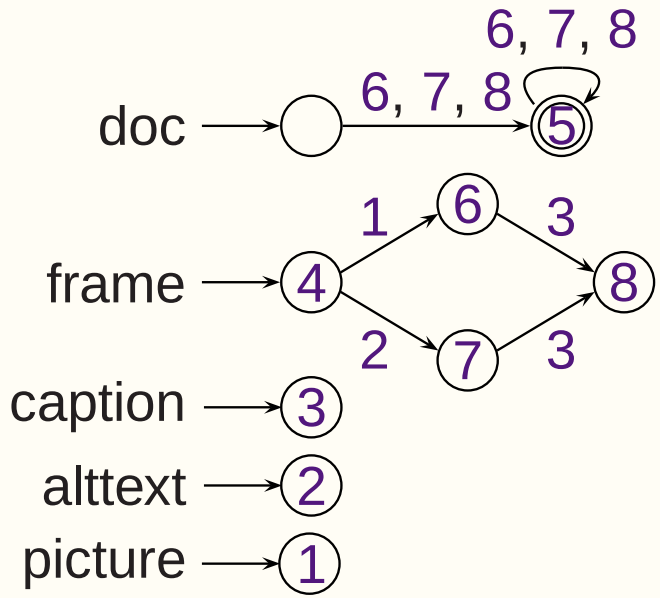
- One-unambiguous regular expressions (W3C)
- To deterministic word automata in $O(|\Sigma| * |DTD|)$ [Brüggemann-Klein & Wood 1998]



DTDs to Stepwise Tree Automata

Glushkov automata to stepwise tree automaton

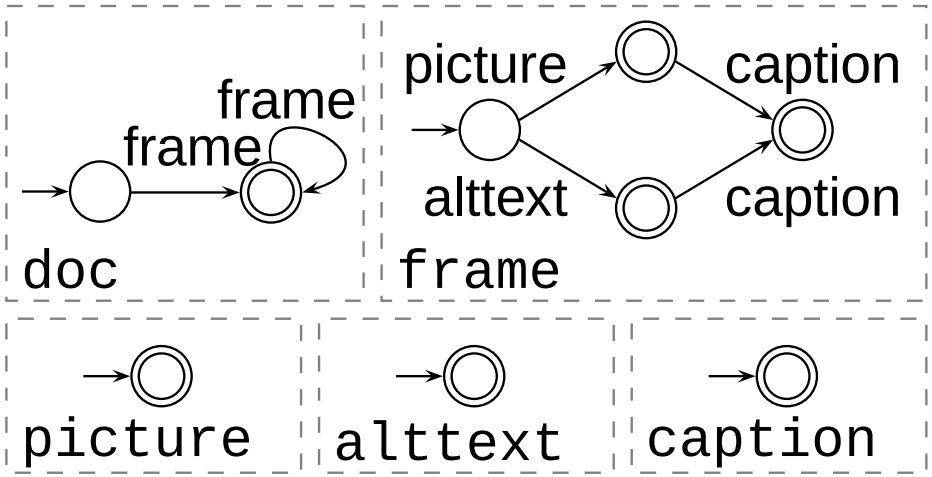




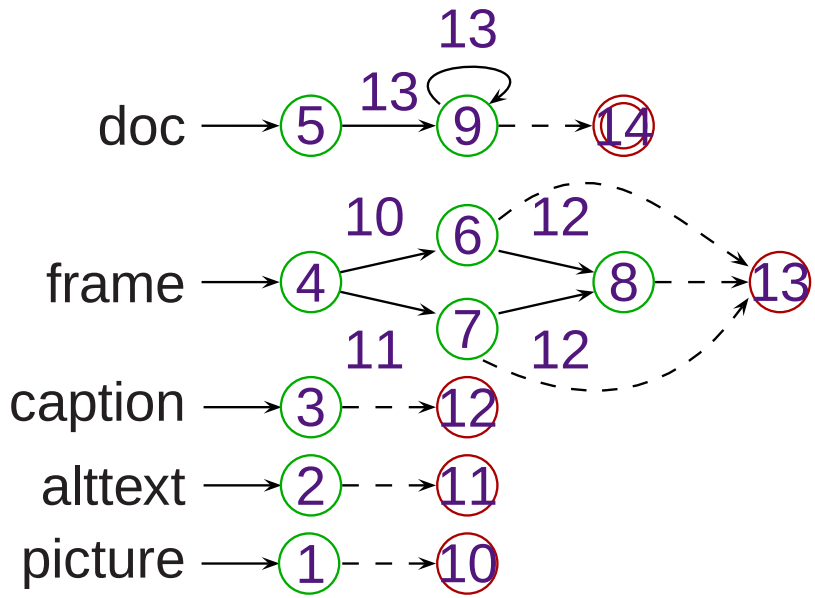
$O(|\Sigma| * |DTD|)$

$O(|\Sigma| * |DTD|^2)$

Factorized Tree Automata



$O(|\Sigma| * |DTD|)$



$O(|\Sigma| * |DTD|)$

Inclusion into Factorized Tree Automata

- More inference rules to consider
  - Epsilon-rules of factorized automaton
  - Distinct cases for each sort of states
  - Other stuff
- Complexity: $O(|A| * |\Sigma| * |DTD|)$

Summary

- Inclusion in deterministic binary tree automata in time $O(|A| * |B|)$
  - Characterization in terms of failure
  - Inferring a ground Datalog program
  - Efficient data structure
- Extension to inclusion in DTDs in time $O(|A| * |\Sigma| * |DTD|)$
  - Unranked tree automata to stepwise tree automata *via* Currying
  - DTDs to factorized tree automata
  - Adapt inclusion algorithm
  - Lower bound?