

A missing link in root-to-frontier tree pattern matching

Loek Cleophas, Kees Hemerik, Gerard Zwaan



Software Construction Group
Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
Eindhoven, The Netherlands
<http://www.win.tue.nl/soc>



FASTAR Research Group
<http://www.fastar.org>

Overview

- Introduction to Tree Pattern Matching
 - Problem Context & Solutions
 - Trees & Tree Automata (*TA*)
 - Tree Pattern Matching (*TPM*)
- Using Aho-Corasick Automata
 - Automaton Construction
 - *TPM* Algorithm
- Using Deterministic Root-to-Frontier *TA*
 - Automaton Construction
 - *TPM* Algorithm
- Conclusions & Future Work

Problem Context & Solutions

'Find all occurrences of a given pattern tree (pattern) in a given subject tree'

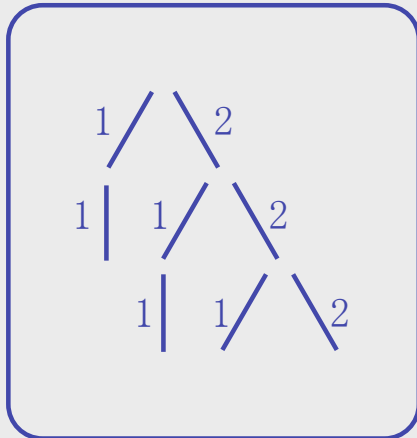
- Important applications: compilers, *XML* document validation, rewriting systems
- Most solutions use *TA* or string automata, frontier-to-root (*FR*) or root-to-frontier (*RF*) traversal
- *FR* algorithms based on deterministic *TA*
- *RF* algorithms based on nondeterministic *TA*
- *RF* algorithms based on deterministic string automata
- Relationship between string automata-based and *TA*-based algorithms unclear

A missing link in root-to-frontier tree pattern matching
PSC 2005, Prague, Czech Republic, August 29 – 31, 2005

Our contribution

- New algorithm
 - *RF* tree traversal
 - Using *DRFTA* for matching pattern stringpaths
- Missing link between string automata-based and *TA*-based algorithms, by using *TA* for stringpath matching
- Not necessarily efficient
 - $O(m \cdot n)$ worst case
 - Many algorithms with better worst case (Kosaraju; Dubiner, Galil, Magen; Cole, Hariharan, Indyke)
 - More elaborate, larger auxiliary data structures

Trees – I



An *ordered tree domain* is a finite non-empty set $D \subseteq \mathbb{N}_+^*$

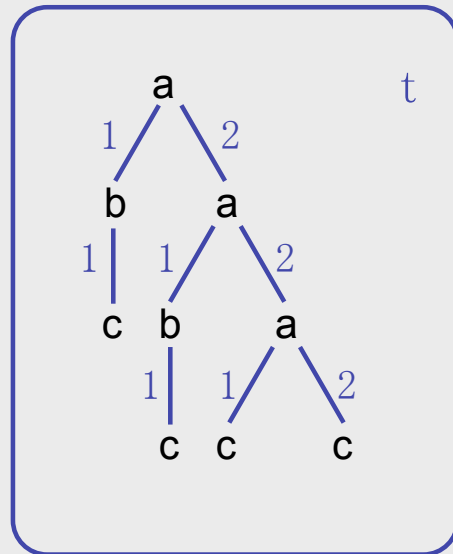
- which is prefix-closed and
- for which for all $n \in D$ and $i \in \mathbb{N}_+$,
 $n \cdot i \in D \Rightarrow \langle \forall j: j < i: n \cdot j \in D \rangle$.

Example: $\{ \varepsilon, 1, 1 \cdot 1, 2, 2 \cdot 1, 2 \cdot 1 \cdot 1, 2 \cdot 2, 2 \cdot 2 \cdot 1, 2 \cdot 2 \cdot 2 \}$

Let V be an alphabet, a *ranking function* is a function $r \in V \rightarrow \mathbb{N}$. Pair (V, r) is a *ranked alphabet*.

Example: $\{ (a, 2), (b, 1), (c, 0) \}$

Trees – II

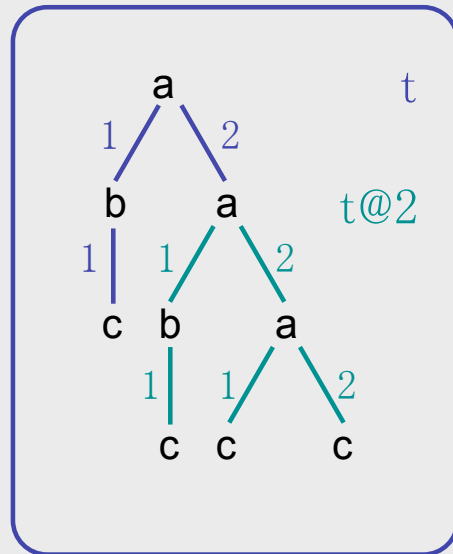


Let D be an ordered tree domain, (V, r) a ranked alphabet.

An *ordered ranked tree* t is a function $t \in D \rightarrow V$ for which for every node $n \in D$, $r(t(n))$ equals the number of $i \in \mathbb{N}_+$ such that $n \cdot i \in D$.

$t@n$ is t 's subtree starting at node n

Trees – II



Let D be an ordered tree domain, (V, r) a ranked alphabet.

An *ordered ranked tree* t is a function $t \in D \rightarrow V$ for which for every node $n \in D$, $r(t(n))$ equals the number of $i \in \mathbb{N}_+$ such that $n \cdot i \in D$.

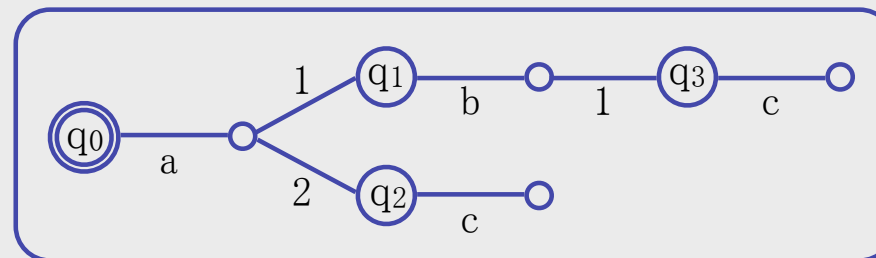
$t@n$ is t 's subtree starting at node n

Tree Automata – I

$Q = \{q_0, q_1, q_2, q_3\}$,
 (V, r) as before,
 $R_a = \{ (q_0, (q_1, q_2)) \}$,
 $R_b = \{ (q_1, (q_3)) \}$,
 $R_c = \{ (q_2, ()), (q_3, ()) \}$,
 $Q_{ra} = \{ q_0 \}$, and
 $Q_{la} = \{ q_2, q_3 \}$.

A tree automaton (TA) M is a 6-tuple $(Q, V, r, R, Q_{ra}, Q_{la})$ with

- Q a finite set, the state set,
- (V, r) a ranked alphabet,
- $R = \langle \text{Set } a: a \in V: R_a \rangle$ the set of transition relations, where $R_a \subseteq Q \times Q^{r(a)}$ for all $a \in V$,
- $Q_{ra} \subseteq Q$, the root accepting states, and
- $Q_{la} \subseteq Q$, the leaf accepting states, defined by $Q_{la} = \langle \text{Set } a, q: a \in V \wedge r(a) = 0 \wedge (q, ()) \in R_a: q \rangle$.

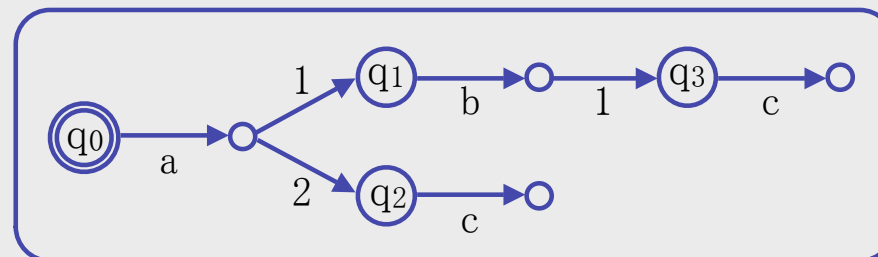


A missing link in root-to-frontier tree pattern matching
 PSC 2005, Prague, Czech Republic, August 29 – 31, 2005

Tree Automata – II

A *nondeterministic root-to-frontier tree automaton (NRFTA)* $M=(Q,V,r,R,Q_{ra},Q_{la})$ is a TA where $R_a \in Q \rightarrow \mathcal{P}(Q^{r(a)})$ for all $a \in V$, i.e. R_a is considered to be directed RF

R	a	b	c
q0	(q1,q2)		
q1		(q3)	
q2			()
q3			()



A *deterministic root-to-frontier tree automaton (DRFTA)* $M=(Q,V,r,R,Q_{ra},Q_{la})$ is an NRFTA where $R_a \in Q \rightarrow Q^{r(a)}$ for all $a \in V$ and $Q_{ra}=\{q_{ra}\}$ – i.e. there is a unique root accepting state (start state)

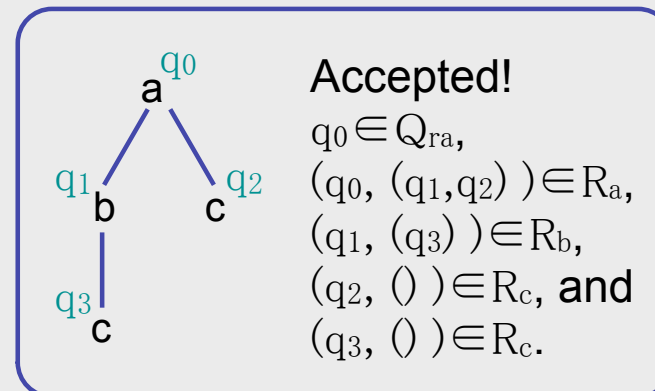
Tree Automata – III

Tree acceptance is defined using assignments of a state to each node of a subject tree:

subject tree accepted

\Leftrightarrow

set of state assignments respecting R and assigning root accepting state to root node is non-empty.



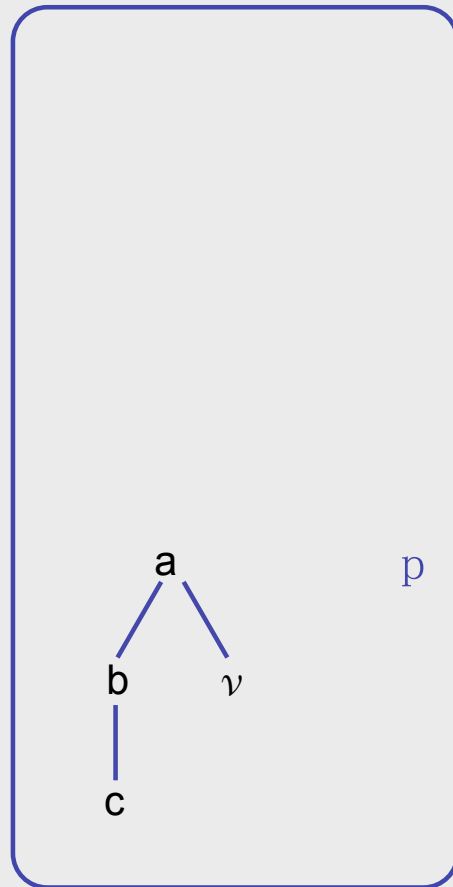
Tree Pattern Matching – I

We extend ranked alphabet (V, r) to (V', r') by adding symbol ν of rank 0, a variable or 'wildcard,' indicating a match of any tree from $\text{Tree}(V, r)$.

Function $\text{Match} \in \text{Tree}(V', r') \times \text{Tree}(V, r) \times D \rightarrow \text{Bool}$ is defined for every pattern $p \in \text{Tree}(V', r')$, subject $t \in \text{Tree}(V, r)$, node $n \in D_t$ by

$$\text{Match}(p, t, n) = \langle \exists s_1, \dots, s_k: s_1, \dots, s_k \in \text{Tree}(V, r): p[s_1, \dots, s_k] = t@n \rangle$$

where $p[s_1, \dots, s_k]$ is the tree obtained by substituting s_1, \dots, s_k respectively for the k instances of ν in p .



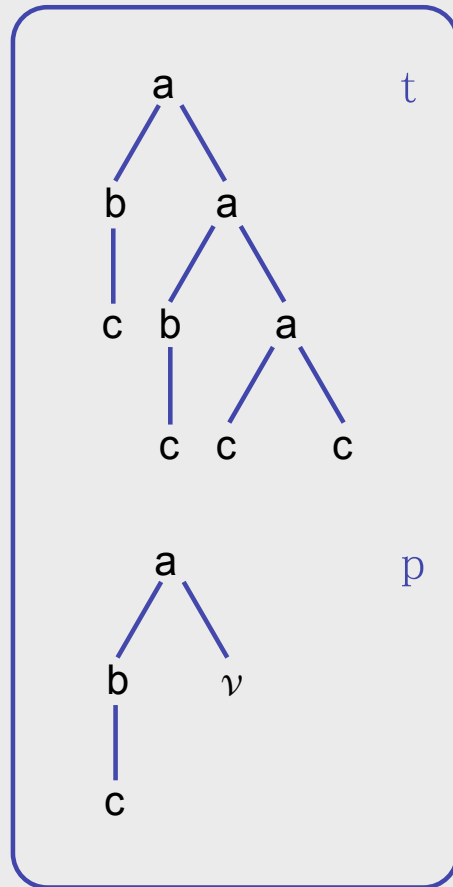
Tree Pattern Matching – I

We extend ranked alphabet (V, r) to (V', r') by adding symbol ν of rank 0, a variable or 'wildcard,' indicating a match of any tree from $\text{Tree}(V, r)$.

Function $\text{Match} \in \text{Tree}(V', r') \times \text{Tree}(V, r) \times D \rightarrow \text{Bool}$ is defined for every pattern $p \in \text{Tree}(V', r')$, subject $t \in \text{Tree}(V, r)$, node $n \in D_t$ by

$$\text{Match}(p, t, n) = \langle \exists s_1, \dots, s_k: s_1, \dots, s_k \in \text{Tree}(V, r): p[s_1, \dots, s_k] = t@n \rangle$$

where $p[s_1, \dots, s_k]$ is the tree obtained by substituting s_1, \dots, s_k respectively for the k instances of ν in p .



Tree Pattern Matching – I

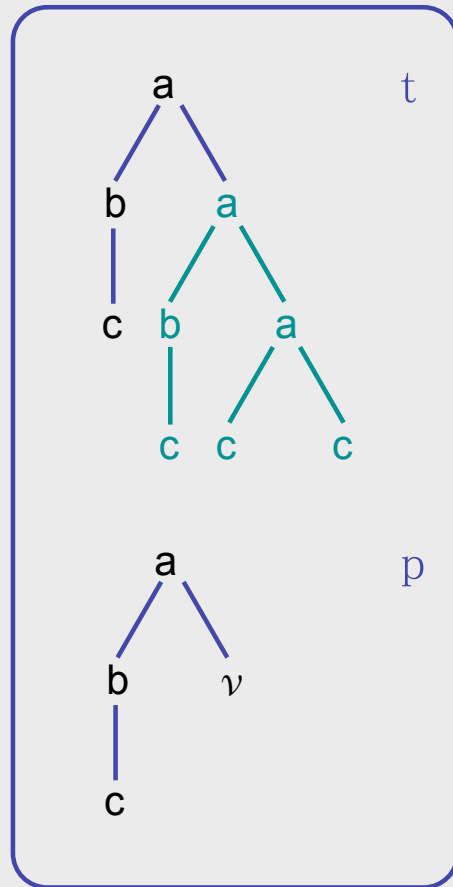
We extend ranked alphabet (V, r) to (V', r') by adding symbol ν of rank 0, a variable or 'wildcard,' indicating a match of any tree from $\text{Tree}(V, r)$.

Function $\text{Match} \in \text{Tree}(V', r') \times \text{Tree}(V, r) \times D \rightarrow \text{Bool}$ is defined for every pattern $p \in \text{Tree}(V', r')$, subject $t \in \text{Tree}(V, r)$, node $n \in D_t$ by

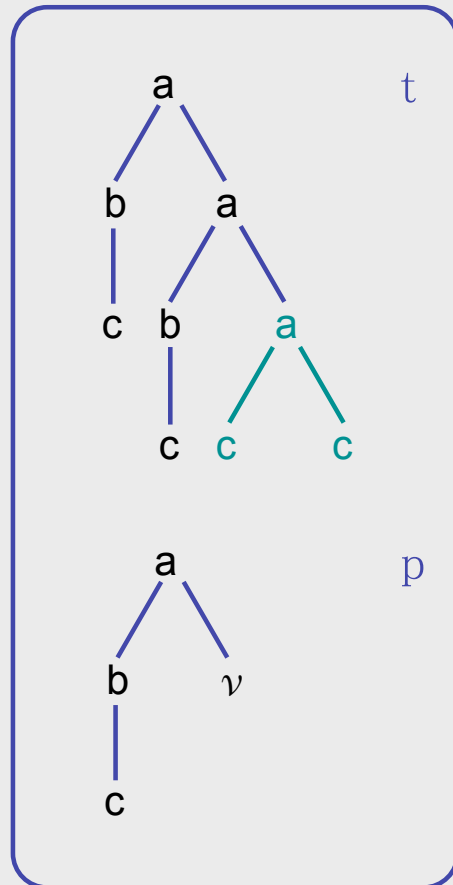
$$\text{Match}(p, t, n) = \langle \exists s_1, \dots, s_k: s_1, \dots, s_k \in \text{Tree}(V, r): p[s_1, \dots, s_k] = t@n \rangle$$

where $p[s_1, \dots, s_k]$ is the tree obtained by substituting s_1, \dots, s_k respectively for the k instances of ν in p .

Example: $\text{Match}(p, t, \varepsilon)$ holds since $t@_\varepsilon = p[a(b(c), a(c, c))]$,



Tree Pattern Matching – I



We extend ranked alphabet (V,r) to (V',r') by adding symbol ν of rank 0, a variable or 'wildcard,' indicating a match of any tree from $\text{Tree}(V,r)$.

Function $\text{Match} \in \text{Tree}(V',r') \times \text{Tree}(V,r) \times D \rightarrow \text{Bool}$ is defined for every pattern $p \in \text{Tree}(V',r')$, subject $t \in \text{Tree}(V,r)$, node $n \in D_t$ by

$$\text{Match}(p,t,n) = \langle \exists s_1, \dots, s_k: s_1, \dots, s_k \in \text{Tree}(V,r): p[s_1, \dots, s_k] = t@n \rangle$$

where $p[s_1, \dots, s_k]$ is the tree obtained by substituting s_1, \dots, s_k respectively for the k instances of ν in p .

Example: $\text{Match}(p,t,\varepsilon)$ holds since $t@_\varepsilon = p[a(b(c), a(c,c))]$,
 $\text{Match}(p,t,2)$ holds since $t@_2 = p[a(c,c)]$.

Tree Pattern Matching – II

Tree also uniquely characterized by set of stringpaths, i.e. all labeled root to leaf paths

Function $SPaths \in Tree(V', r') \rightarrow \mathcal{P}((V' \cdot \mathbb{N}_+)^* \cdot V')$ is defined for every tree $t \in Tree(V', r')$ to return its stringpaths

Example:

$$SPaths(t) = \{a1b1c, a2a1b1c, a2a2a1c, a2a2a2c\},$$

$$SPaths(t@2) = \{a1b1c, a2a1c, a2a2c\},$$

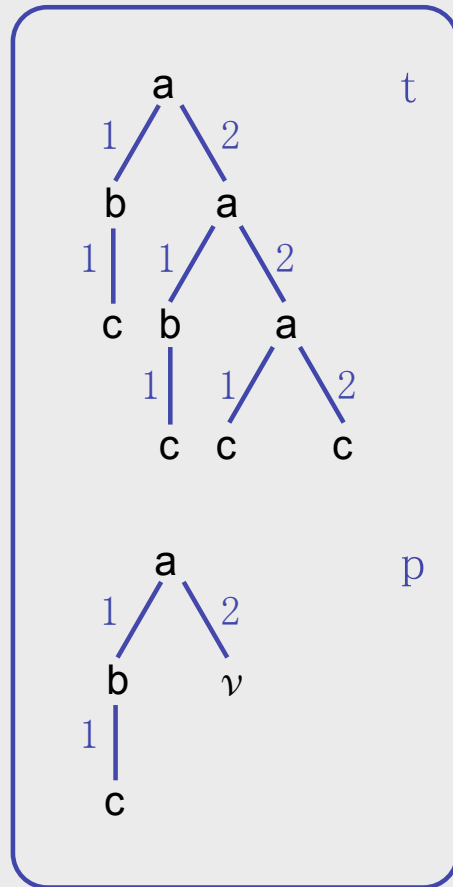
$$SPaths(p) = \{a1b1c, a2 \nu\}$$

Stringpath sp matches in t starting at n

\Leftrightarrow

sp is in $SPaths(t@n)$ or

$sp \uparrow 1 = \nu \wedge sp \downarrow 1$ is prefix of some stringpath in $SPaths(t@n)$



Tree Pattern Matching – III

We can rephrase matching in terms of stringpaths:

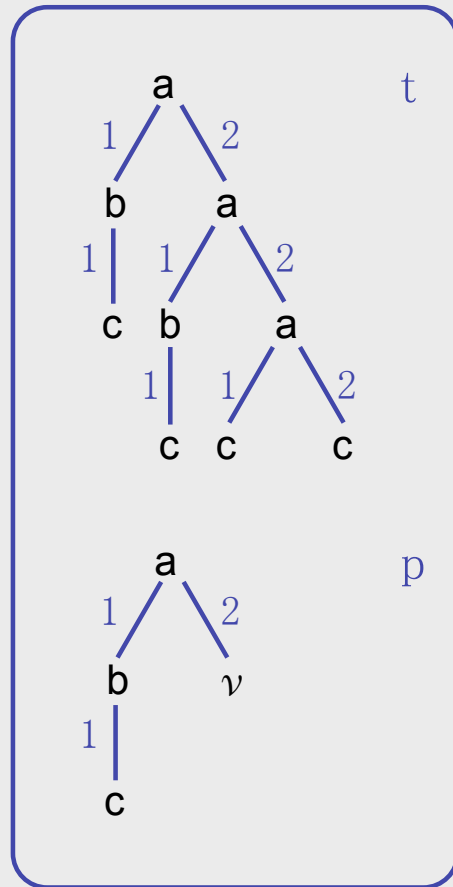
$\text{Match}(p, t, n)$

\Leftrightarrow

each stringpath in $\text{SPaths}(p)$ matches in t starting at n

Example:

$\text{Match}(p, t, 2)$ holds since $a1b1c \in \text{SPaths}(t@2)$ and $a2 \vee \exists v = v \wedge a2 \vee \exists v1$ is prefix of $a2a1c, a2a2c \in \text{SPaths}(t@2)$

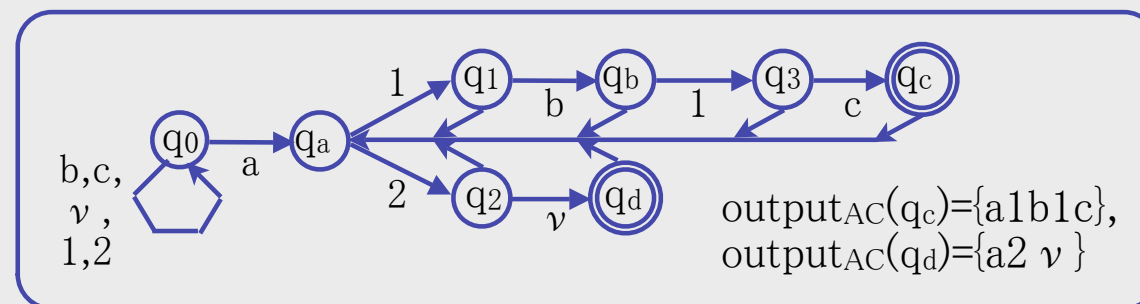


- Match registration in *RF* traversal easiest at endpoint
- Can be adapted to register matches at beginpoints
- Can then be adapted to register matches of complete pattern trees based on stringpath matches

Using Aho-Corasick Automata – I

High-level construction:

1. Construct *trie* recognizing $SPaths(p)$
2. For states corresponding to stringpath match, define output equal to stringpath
3. Add 'self-loop' transition on every symbol to q_0
4. Determinize automaton and output function



Result usable for stringpath matching in *RF* traversal

Using Aho-Corasick Automata – II

```

{ Pre:  $q = \delta^*(q_0, RPath(t, n) \downarrow 1)$  }
proc Traverse( $q:Q, n:D$ ) =
[[ var  $q_{next}:Q; i:\mathbb{N}_+; sp:(V' \cdot \mathbb{N}_+)^* \cdot V'$ 
| for  $i:1 \leq i \leq r'(t(n)) \rightarrow$ 
     $q_{next} := \delta(\delta(q, t(n)), i);$ 
    Traverse( $q_{next}, n \cdot i$ )
rof;
for  $sp:sp \in output_{AC}(\delta(q, t(n))) \rightarrow$ 
    ``register  $sp$  match at its endpoint  $n$ ``; rof;
for  $sp:sp \in output_{AC}(\delta(q, v)) \rightarrow$ 
    ``register  $sp$  match at its endpoint  $n$ ``; rof;
]];
{ Post: every stringpath match in  $t$  with endpoint in  $t@n$ 
      has been registered at its endpoint }

```


Using Deterministic RF Tree Automata – I

New *RF TPM* algorithm, uses *DRFTA* for matching pattern stringpaths, combined with *RF* traversal of subject tree

High-level construction:

1. Construct *DRFTA* recognizing pattern tree p
2. For states and symbols corresponding to stringpath match, define output equal to stringpath
3. Add 'self-loop' transitions on every symbol of rank >0 to q_0
4. Determinize automaton and output function

Result usable for stringpath matching in *RF* traversal

Using Deterministic RF Tree Automata – II

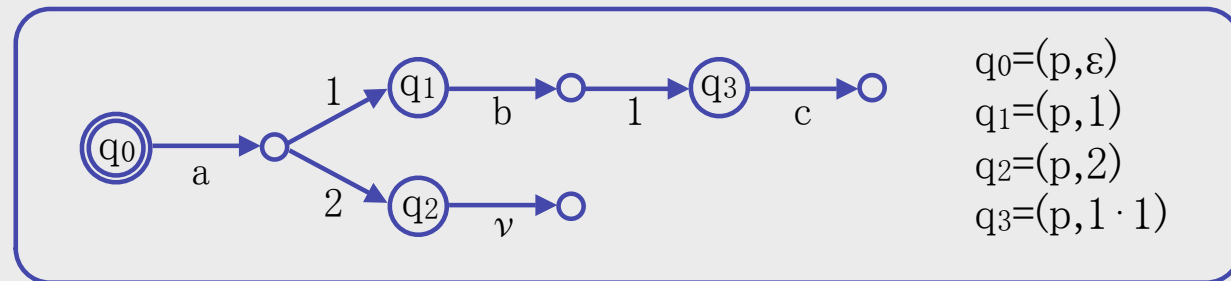
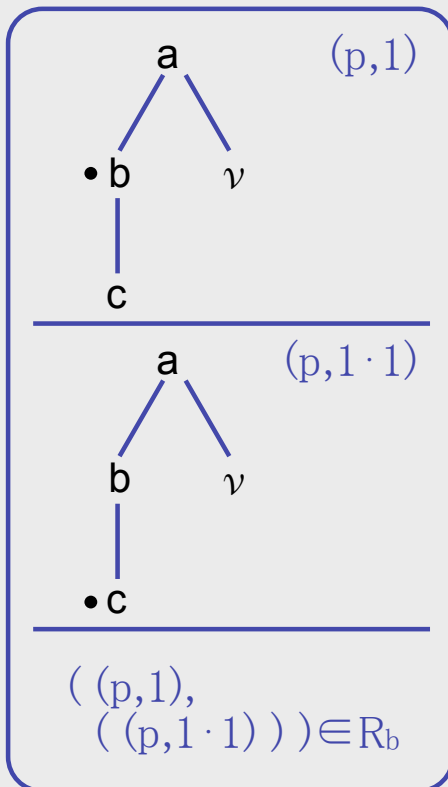
Step 1: Construction of DRFTA accepting pattern tree

Let $p \in \text{Tree}(V', r')$, then $M = (Q, V', r', R, Q_{ra}, Q_{la})$ where

$Q = \text{DT}(p)$, the set of *dotted trees* on p

$Q_{ra} = \{(p, \varepsilon)\}$,

$R_a = \langle \text{Set } m: (p, m) \in Q \wedge p(m) = a: ((p, m),$
 $(p, m \cdot 1),$
 $\dots,$
 $(p, m \cdot r'(a))) \rangle \rangle$ for all $a \in V'$.



Using Deterministic RF Tree Automata – III

Step 2: Definition of output function

Theorem: Given subject tree t , pattern tree p , nodes $n \in D_t$ and $m \in D_p$, and a *DRFTA* as in preceding construction,

(p, m) is assigned to node n
 by the *DRFTA* computation \Rightarrow $RPath(p, m)$ matches
 ending at node n
 $\wedge (t(n)=p(m) \vee \nu = p(m))$

Proof: by structural induction, see paper.

Pattern stringpaths are rootpaths ending in symbols of rank 0, so define $output_{DRFTA}$ for $(p, m) \in Q$ and $a \in V'$ such that $a=p(m) \wedge r'(a)=0$ by $output_{DRFTA}((p, m), a) = RPath(p, m)$.

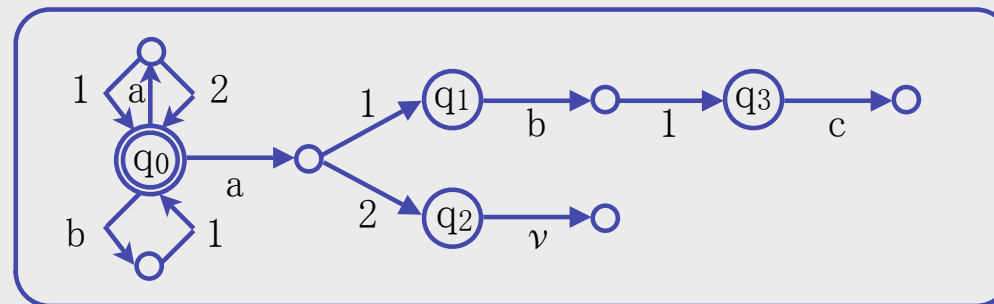
Note that $output_{DRFTA}$ is used with $a=t(n)$ or $a=\nu$ only.

Using Deterministic RF Tree Automata – IV

Step 3: Addition of 'self loops'

Given M as before, let $M' = (Q, V', r', R', Q_{ra}, Q_{la})$ where

$$R'_a = R_a \cup \begin{cases} \{ ((p, \varepsilon), ((p, \varepsilon)^{r'(a)})) \} & \text{for all } a \in V' \text{ with } r'(a) > 0 \\ \emptyset & \text{for all } a \in V' \text{ with } r'(a) = 0 \end{cases}$$



Reverse implication of theorem holds for this automaton

Proof: by structural induction again, see paper.

Using Deterministic RF Tree Automata – V

Step 4: Determinization

- Resulting *DRFTA* in general recognizes superset of *NRFTA*'s language
- Superset is set of trees of which every stringpath occurs in some tree from *NRFTA*'s language
 - No problem for stringpath matching

Corollary: Given subject tree t , pattern tree p , nodes $n \in D_t$ and $m \in D_p$, and a *DRFTA* obtained by subset construction,

(p,m) is part of state assigned to node n by the *DRFTA* computation $\wedge (t(n)=p(m) \vee \nu =p(m)) \equiv \text{RPath}(p,m)$ matches ending at node n

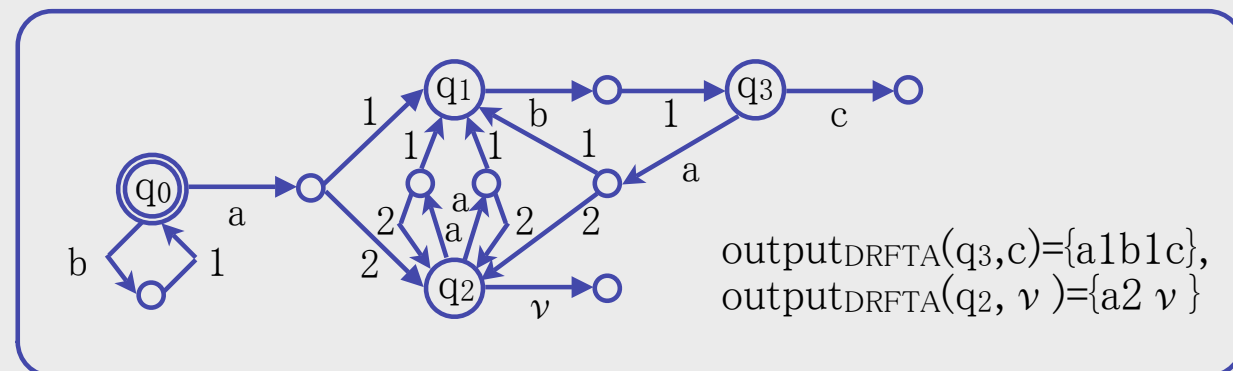
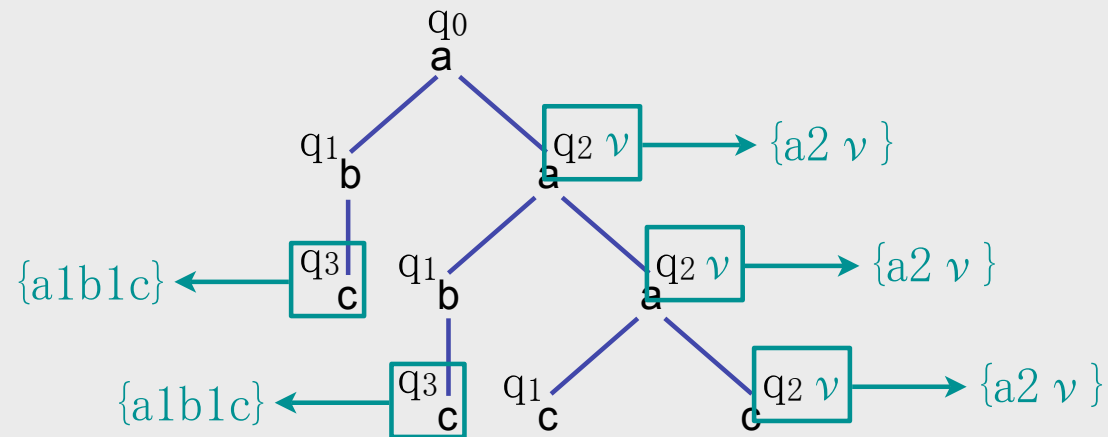
Using Deterministic RF Tree Automata – VI

```

{ Pre:  $q = p_k$  where  $k = |n| \wedge p_0 = q_0 \wedge$ 
   $\langle \forall i : 1 \leq i \leq k : p_i = \pi_{RPath(t,n)_{2i}}(RPath(t,n)_{2i-1}(p_{i-1})) \rangle \}$ 
proc Traverse( $q:Q, n:D$ ) =
[[ var  $q_{next}:Q; i:\mathbb{N}_+; sp:(V' \cdot \mathbb{N}_+)^* \cdot V'$ 
| for  $i:1 \leq i \leq r'(t(n)) \rightarrow$ 
   $q_{next} := \pi_i(R_{t(n)}(q));$ 
  Traverse( $q_{next}, n \cdot i$ )
rof;
for  $sp:sp \in output_{DRFTA}(q, t(n)) \rightarrow$ 
  ``register  $sp$  match at its endpoint  $n$ ''; rof;
for  $sp:sp \in output_{DRFTA}(q, v) \rightarrow$ 
  ``register  $sp$  match at its endpoint  $n$ ''; rof;
]];
{ Post: every stringpath match in  $t$  with endpoint in  $t@n$ 
  has been registered at its endpoint }

```

Using Deterministic RF Tree Automata – VII



Conclusions & Future Work

- Presented two algorithms for stringpath-based *TPM*
 - One well known: based on *RF* traversal, Aho-Corasick
 - One new: based on *RF* traversal, use of *DRFTA*
- Presented them in similar style, highlighting similarities
 - Provided missing link between *TPM* algorithms using *TA* and those using stringpath automata
 - Very similar, difference in automata and output functions
- More detailed automata comparison
- Extension to multiple patterns, tree parsing
- Use of different keyword pattern matching algorithms