TU/e technische universiteit eindhoven

University of Pretoria

T(h)ree taxonomies (and a toolkit ?) Treeology and Finite Tree Automata

Loek Cleophas

http://www.win.tue.nl/~lcleopha

SET Software Engineering & Technology

Software Engineering & Technology Group
(formerly Software Construction Group)
Dept. of Mathematics and Computer Science
Technische Universiteit Eindhoven

FASTAR

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

Introduction

- My research:
  - Subject: taxonomies & toolkits of algorithms related to *regular tree languages*
  - Supervised by Kees Hemerik, Bruce Watson, Mark van de Brand
- This talk:
  - Motivation
  - Approach
  - Concepts
  - Some results
  - Future work

Motivation – Why Regular Tree Languages?

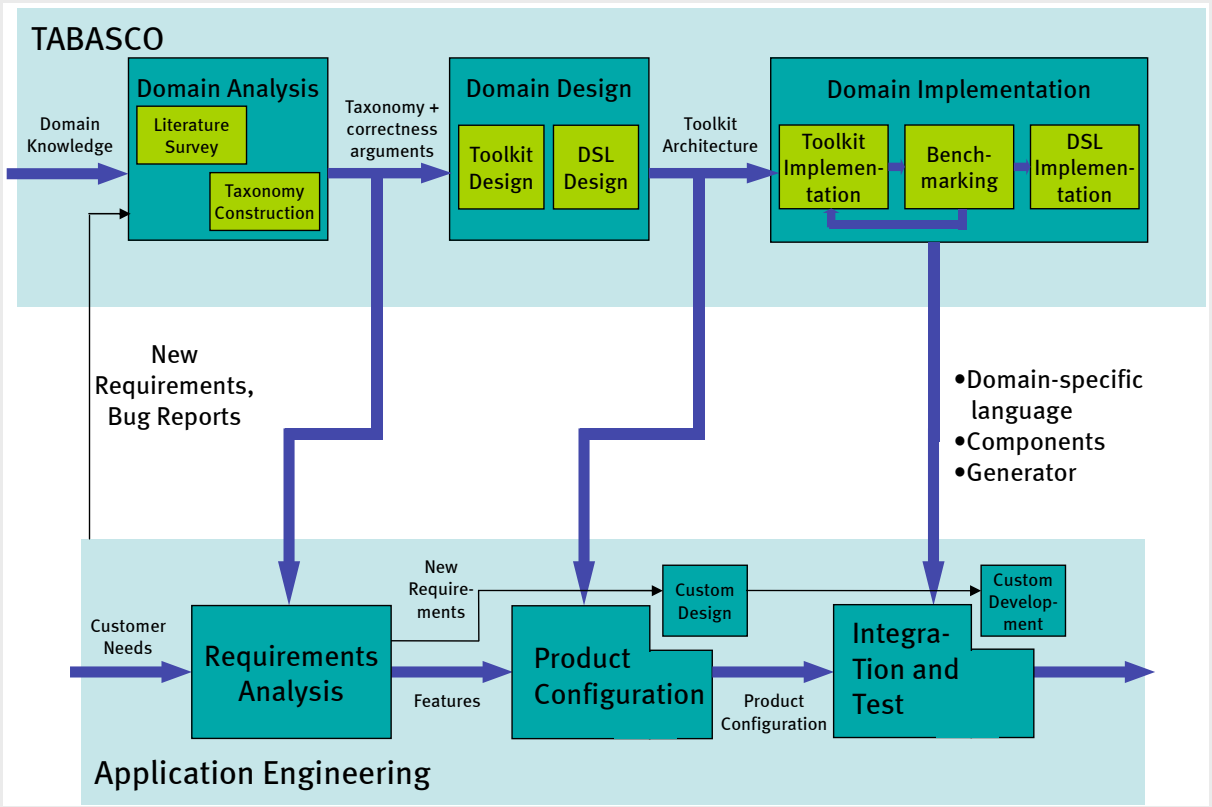
- Well-established theory
- Algorithmic problems
  - Tree Acceptance
  - Tree Parsing
  - Tree Pattern Matching
- Related problems, related solutions
- Many algorithms
- Many applications
  - Code Generation
  - Term Rewriting
  - XML Processing,Verification, Unification, ...

Motivation – Deficiencies

- Unaccessible
- Hard to compare
  - Style
  - Formality
- Distance between theory & practice
- No collection of implementations
- Hard to choose

TAxonomy-BAsed Software COnstruction

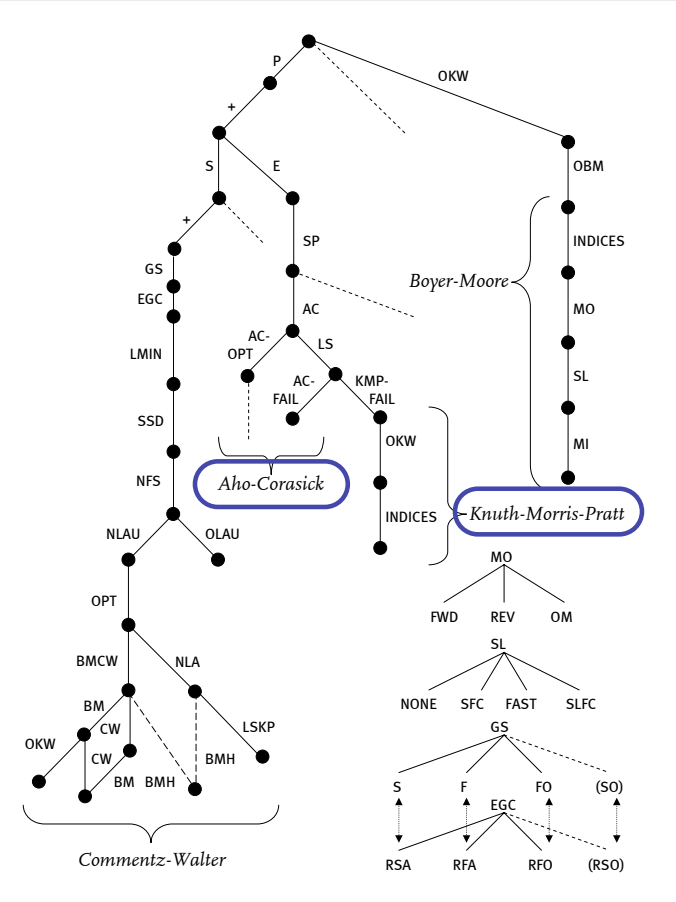
Process consists of multiple steps:



Taxonomy — I

- Classification of *algorithms* based on their *essential details*
- Root is high-level solution
  - with formal requirements
- Others derived by adding *details*
  - refined solution to same/similar problem
  - algorithms from literature, new algorithms
  - correctness arguments

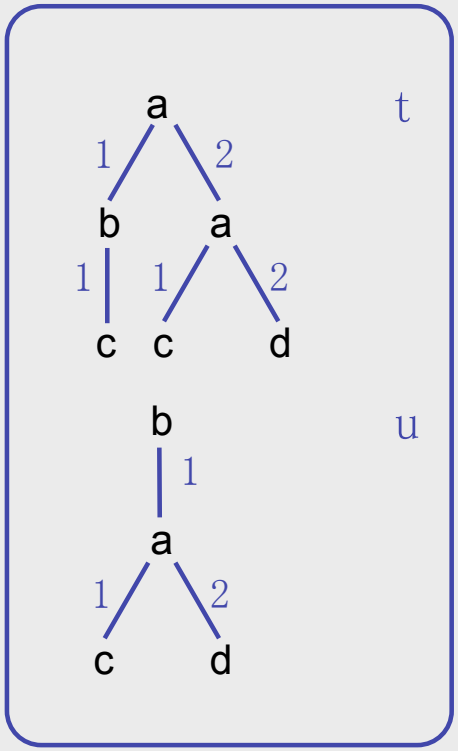
Example: Keyword Pattern Matching



Taxonomy — II

- Advantages
  - Eases comparison
  - Clear and correct presentation
  - Usable for discovering new algorithms, teaching
  - Aids in construction of toolkit
- Disadvantage
  - Time and effort

Concepts – Trees

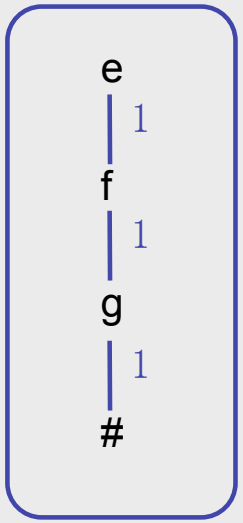


Trees

- Nodes labeled
- Siblings ordered
- Fixed symbol rank, arity

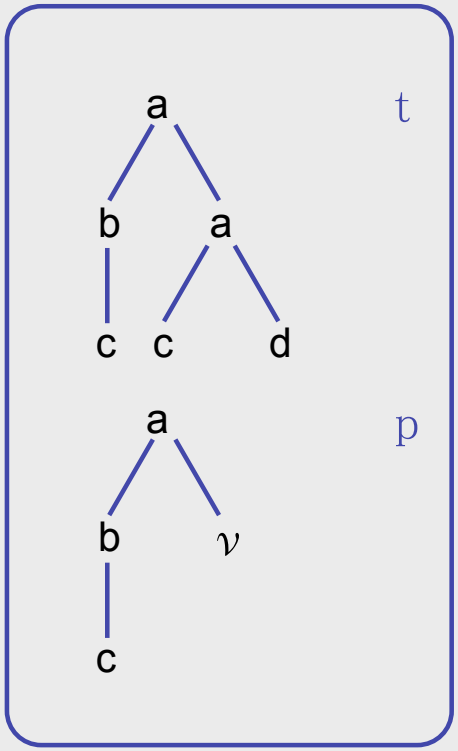
Example: ranked alphabet { (a,2), (b,1), (c,0), (d,0) },
trees t=a(b(c),a(c,d)) and u=b(a(c,d))

Generalization of strings:
string efg becomes tree
with alphabet { (e,1), (f, 1), (g, 1), (#, 0) }



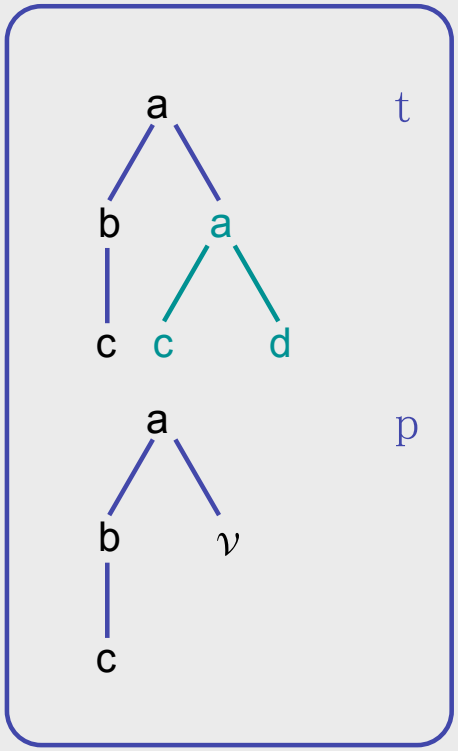
Concepts – Tree Patterns & Matching – I

Variable symbols of rank 0 indicate match of any tree



Example:
ranked alphabet $\{ (a,2), (b,1), (c,0), (d,0), (\nu ,0) \}$ with variable ν ,
tree pattern $p=a(b(c), \nu)$

Concepts – Tree Patterns & Matching – I

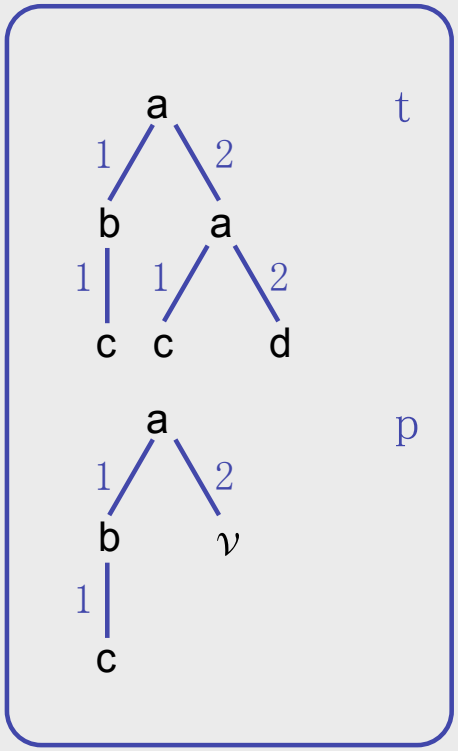


Variable symbols of rank 0 indicate match of any tree

Example:
ranked alphabet $\{ (a,2), (b,1), (c,0), (d,0), (\nu,0) \}$ with variable ν,
tree pattern $p=a(b(c), \nu)$

Tree and pattern match
$\Leftrightarrow$
tree equals pattern with substitution of variables

Concepts – Tree Patterns & Matching – II



Alternative: decompose trees into *stringpaths*

Example:

SPaths(t)={a1b1c,a2a1c,a2a2d},
SPaths(p)={a1b1c,a2 ν }

Tree and pattern match
⇔
each pattern stringpath matches a tree stringpath
(ignoring variable occurrences)

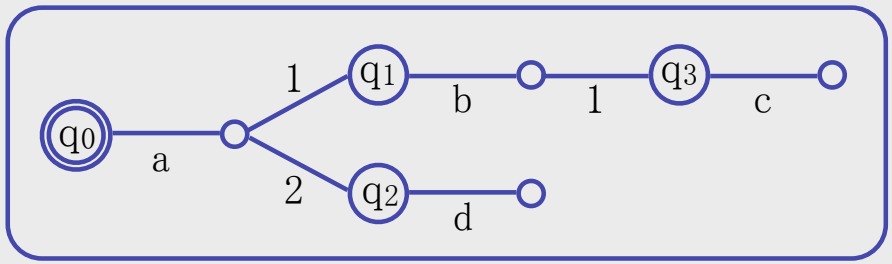
Concepts – Tree Automata – I

TA $M=(Q,V,r,R,Q_{ra})$ where

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

$Q=\{q_0,q_1,q_2,q_3\}$,
$R_a=\{ (q_0, (q_1,q_2)) \}$,
$R_b=\{ (q_1, (q_3)) \}$,
$R_c=\{ (q_3, ()) \}$,
$R_d=\{ (q_2, ()) \}$,
$Q_{ra}=\{ q_0 \}$.

Generalization of (undirected) finite (string) automata

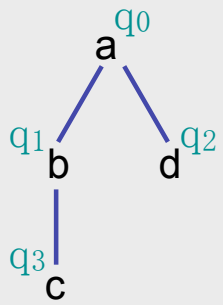


Concepts – Tree Automata – II

Subject tree accepted

$\Leftrightarrow$

state assignment exists respecting R and assigning root accepting state to root node

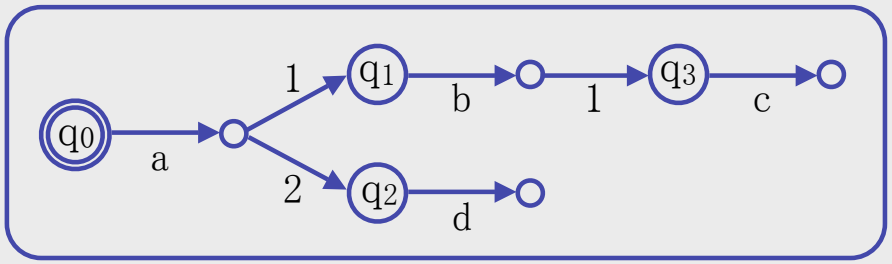


Accepted!

$q_0 \in Q_{ra}$,
$(q_0, (q_1, q_2)) \in R_a$,
$(q_1, (q_3)) \in R_b$,
$(q_2, ()) \in R_d$, and
$(q_3, ()) \in R_c$.

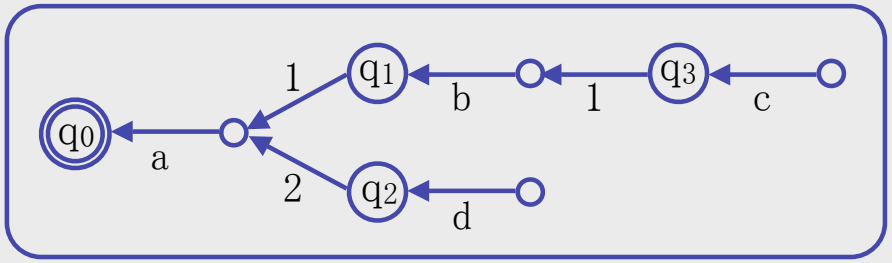
Concepts – Tree Automata – III

- Directed?
  - Root-to-frontier (RF) / top-down: Q_{ra} as `start' states



Concepts – Tree Automata – III

- Directed?
  - Root-to-frontier (RF) / top-down: $\mathbb{Q}_{ra}$ as `start' states
  - Frontier-to-root (FR) / bottom-up: $\mathbb{Q}_{ra}$ as `final' states



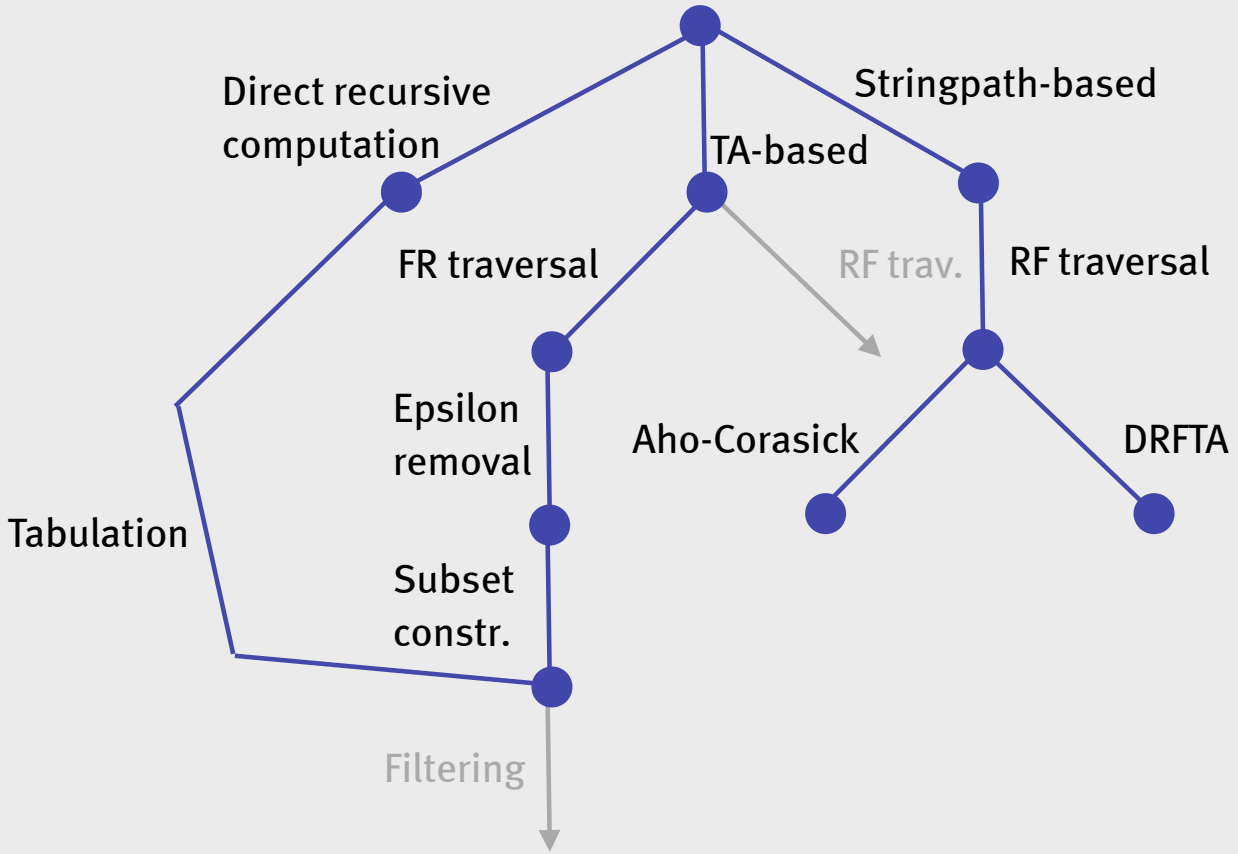
Concepts – Tree Automata – III

- Directed?
  - Root-to-frontier (RF) / top-down: Q_{ra} as `start' states
  - Frontier-to-root (FR) / bottom-up: Q_{ra} as `final' states
- Epsilon transitions? Deterministic?
- Epsilon removal, Subset construction
- All equally powerful, except $DRFTA$
  - yet $DRFTA$ still usable for stringpath matching

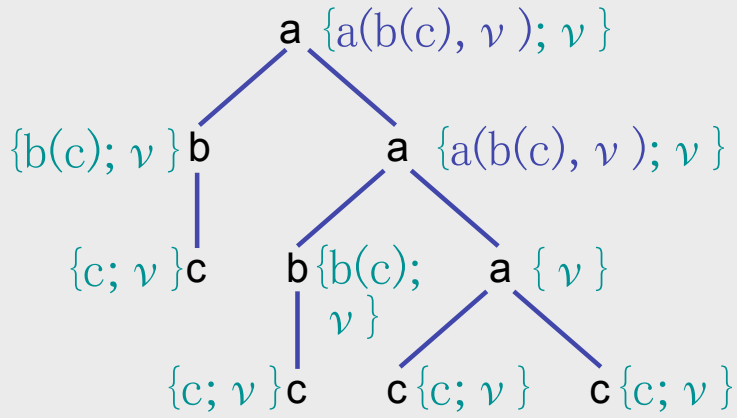
Sketch of results

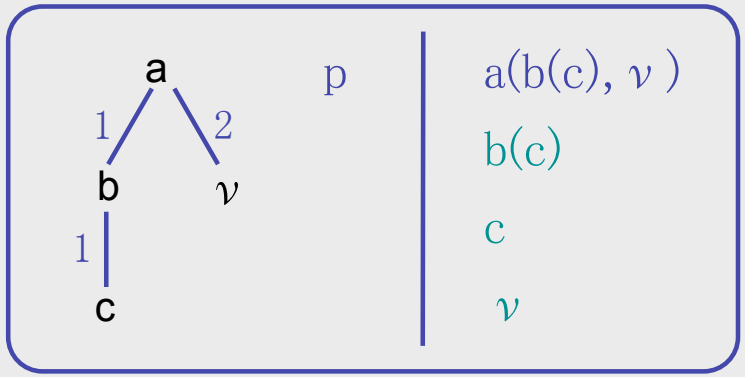
- Recall: Tree Pattern Matching
  - Determine all matches of pattern tree in subject tree
- Application: Term Rewriting
  - Rules of form *TreeA* -› *TreeB* can be applied to rewrite parts of subject tree
  - Requires knowing where *TreeA* matches
  - Instance of Tree Pattern Matching!
- Sketch of taxonomy and some solutions
  - Using FR traversal, direct recursive match computation
  - Using FR traversal, DFRTA
  - Using RF traversal, Aho-Corasick stringpath automaton
  - Using RF traversal, DRFTA for stringpath matching

Sketch of results – Taxonomy

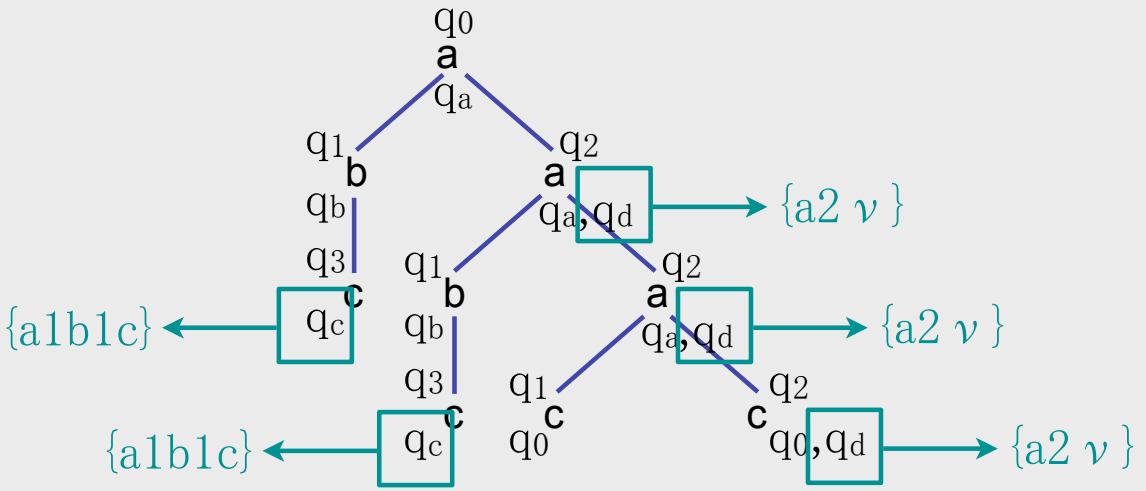


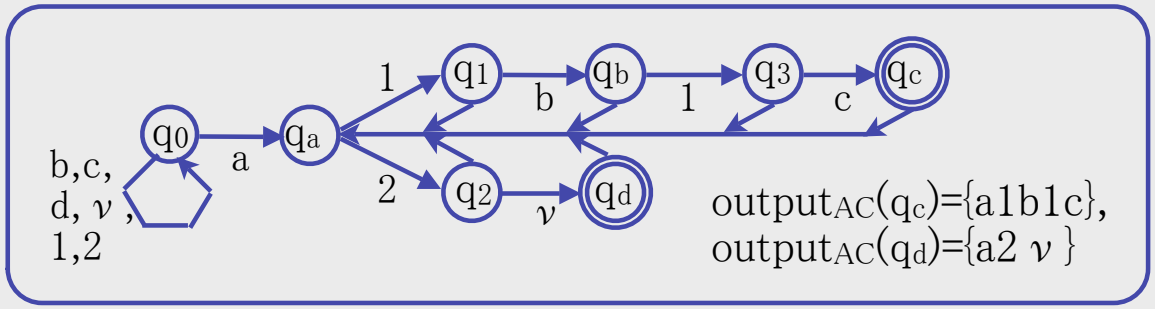
Direct Recursive Match Computation



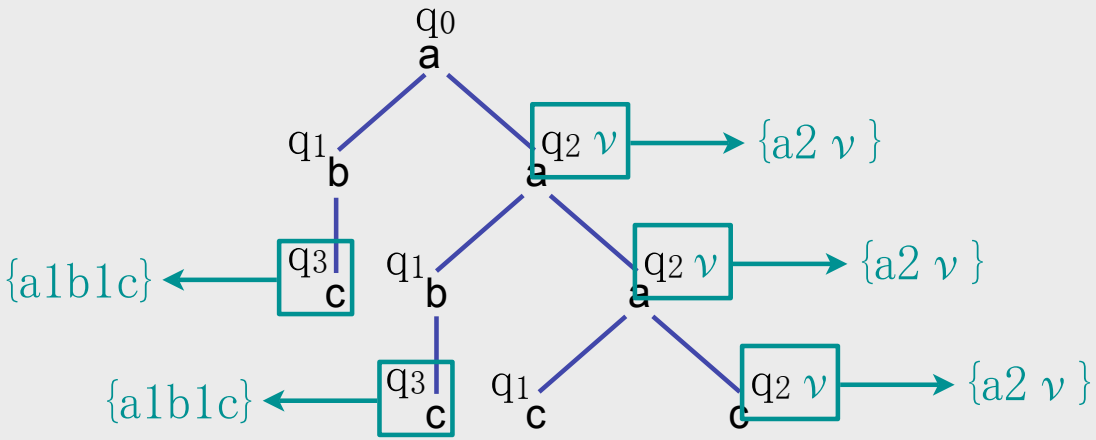


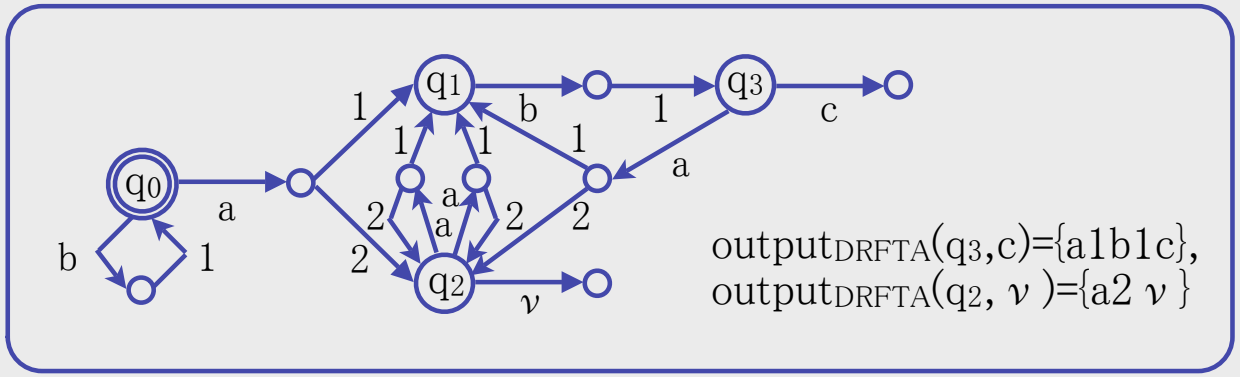
Aho-Corasick Stringpath Automaton



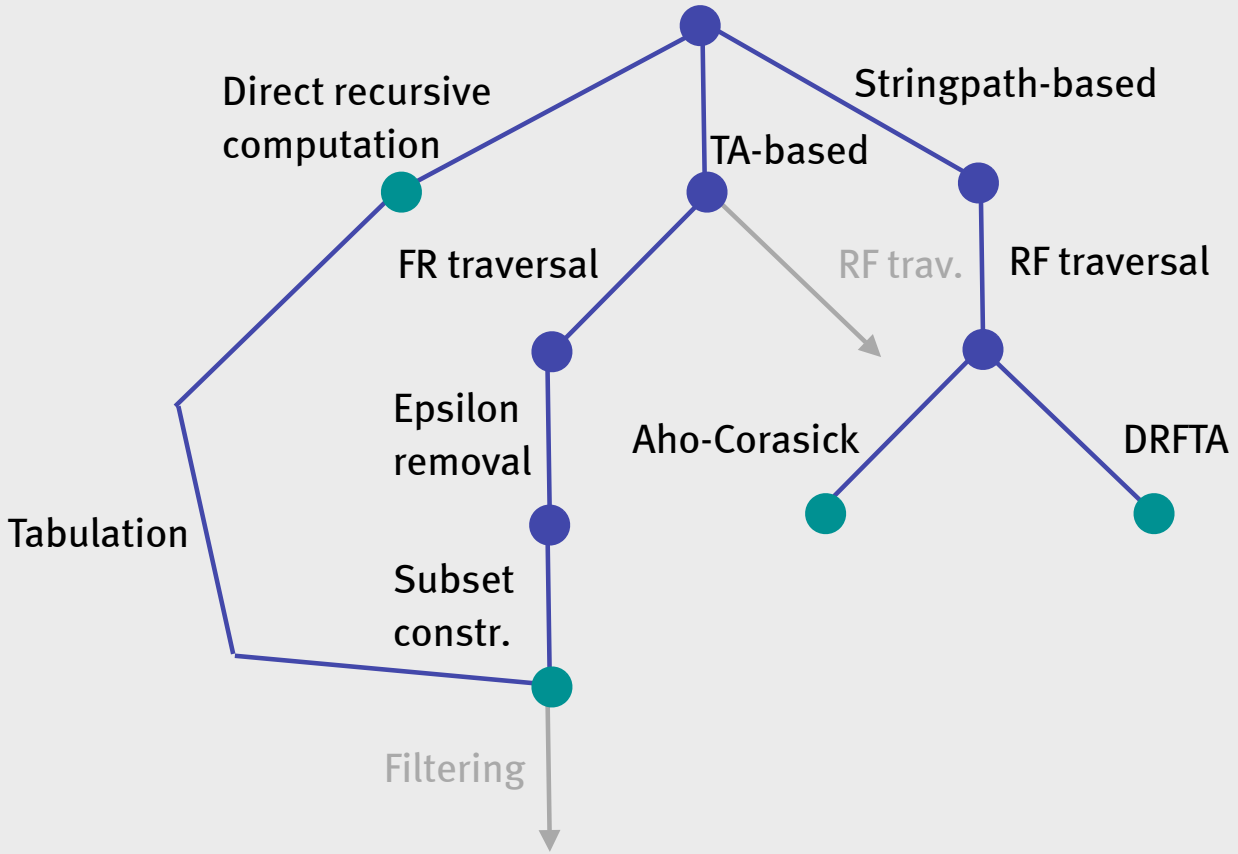


DRFTA for Stringpath Matching





Sketch of results – Taxonomy



Future Work

- Complete taxonomy presentation
  - Tree pattern matching
  - Tree acceptance
- Extend to tree parsing
- Toolkit?
  - Currently MSc student work