Tree Algorithms: Two Taxonomies and a Toolkit

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

Mark van den Brand  Kees Hemerik  Bruce Watson

Software Engineering & Technology Group  
Dept. of Mathematics and Computer Science  
Technische Universiteit Eindhoven

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

• **Subject:** Classification & implementation of algorithms related to regular tree languages
  – Algorithms for *tree acceptance, tree pattern matching*
  – Classification in form of *taxonomy*
  – Implementation in form of *toolkit*

• **Context:** history, relevance
• Trees, grammars, applications
• Our classification: Taxonomies
• Our implementation: Toolkit
• Conclusions
Regular String Language Theory — I

• Regular Grammar (and Regular Expression)
  – Different types, transformations between them
• Finite Automaton
  – Nondeterministic with/without $\varepsilon$-transitions, deterministic
• Theoretical Results
  – Equivalence of NFA and DFA (subset construction)
  – Equivalence of RG, RE, and FA
• Problems
  – Membership/Acceptance $s \in L$
  – Keyword Pattern Matching (KPM)
  – ...

$s \in L$
Regular String Language Theory — II

• Theoreticians (1950s):
  – Solve by constructing and using FA based on RG/RE
  – Done, move on!

• In practice (1960s - now):
  – Many applications (text search, DNA processing)
  – Many FA constructions, acceptance/KPM algorithms
    • More efficient; for specific situations
  – Difficult to find, understand, compare
  – Dichotomy between theory and practice
  – Hard to compare and choose implementations

• Taxonomies and toolkits (Watson, 1995; Watson & Zwaan, 1996; Cleophas, Watson & Zwaan, 2004)
Regular Tree Language Theory — I

- **Regular Tree Grammar** (and **Regular Tree Expression**)
  - Different types, transformations between them
- **Finite Tree Automaton** (TA)
  - Nondeterministic with/without $\varepsilon$-transitions, deterministic
  - Undirected, root-to-frontier (RF), frontier-to-root (FR)
- **Theoretical Results**
  - Equivalence of TAs (except DRFTA) (*subset construction*)
  - Equivalence of RTG, RTE, and TA (except DRFTA)
- **Problems**
  - Membership/Tree (Grammar) Acceptance (TGA)
  - Tree Pattern Matching (TPM)
  - ...
Regular Tree Language Theory — II

• Theoreticians (1960s):
  – Solve by constructing and using TA based on RTG/RTE
  – Done, move on!

• In practice (1975 - now):
  – Applications in code generation, term rewriting
  – Many TA constructions, TGA/TPM algorithms
    • More efficient; for specific situations
  – Difficult to find, understand, compare
  – Dichotomy between theory and practice
  – Hard to compare and choose implementations

• Taxonomies and toolkit (Cleophas, Hemerik & Zwaan, 2005/2006; Strolenberg, 2007; Cleophas, 2007)
What kind of trees?

- **String as tree**
  
  String `efge` as tree `e(f(g(e(#))))`
  
  with symbols of **arity 1**

- **Generalization:**
  
  Allow symbols of arity > 1

- Fixed arity per symbol

- Order of siblings relevant
What are Regular Tree Grammars?

- Generalization of regular string grammar
  - Recall right regular string grammar production forms
    \[ A \rightarrow wB, \ A \rightarrow w \ (w \in \Sigma^*) \]

- Regular tree grammar
  - Form \( A \rightarrow t \) with \( t \) a tree, nonterminals at leaves

1. \[ S \rightarrow a \]
2. \[ S \rightarrow \begin{array}{c} a \\ B \\ d \end{array} \]
3. \[ S \rightarrow c, \]
4. \[ B \rightarrow b, \]
5. \[ B \rightarrow S, \]
6. \[ B \rightarrow d \]
What are the applications? — I

• Term Rewriting
  – Need to find instances of rewrite rules’ left hand sides in subject
  – *Tree Pattern Matching!*

\[
\begin{align*}
(1) \quad & \text{and} & \rightarrow & T \\
(2) \quad & \text{and} & \rightarrow & F \\
(3) \quad & \text{and} & \rightarrow & F \\
(4) \quad & \text{and} & \rightarrow & \text{and}
\end{align*}
\]
What are the applications? — II

• Instruction Selection
  - Given intermediate representation tree, determine covering/instructions
  - Use RTG; each production has associated instruction; determine derivation to obtain covering/instructions
  - *Tree Parsing*, extending *Tree Acceptance*

1. \( R_i \rightarrow c \)
   \[
   MOV \#c, R_i
   \]
2. \( R_i \rightarrow M \\
   \]
   \[
   \]
   \[
   MOV * R_j, R_i
   \]
3. \( R_i \rightarrow M \\
   \]
   \[
   \]
   \[
   MOV c(R_j), R_i
   \]
4. \( R_i \rightarrow + \\
   \]
   \[
   \]
   \[
   ADD R_i, R_j
   \]

\[
R_3
\]
Appearance of algorithms

- Brainerd, 1967 & 1969
- Kron, 1975
- Hatcher, 1985; Hatcher & Christopher, 1986
- Turner, 1986
- van Dinther, 1987
- Chase, 1987
- Aho, Ganapathi & Tjang, 1985, 1988
- van de Meerakker, 1988
- Weisgerber & Wilhelm, 1989
- Hemerik & Katoen, 1989
- Balachandran, Dhamdhere & Biswas, 1990
- Ferdinand, Seidl & Wilhelm, 1994
- Wilhelm & Mauer, 1995
- Comon et al., 2003
- Cleophas, Hemerik & Zwaan, 2005 & 2006
Appearance of algorithms — TPM

- Brainerd, 1967 & 1969
- Kron, 1975
- Hoffmann & O’Donnell, 1982
- Hatcher, 1985; Hatcher & Christopher, 1986
- Turner, 1986
- van Dinther, 1987
- Chase, 1987
- Aho, Ganapathi & Tjang, 1985, 1988
- van de Meerakker, 1988
- Weisgerber & Wilhelm, 1989
- Hemerik & Katoen, 1989
- Balachandran, Dhamdhere & Biswas, 1990
- Ferdinand, Seidl & Wilhelm, 1994
- Wilhelm & Mauer, 1995
- Comon et al., 2003
- Cleophas, Hemerik & Zwaan, 2005 & 2006
Appearance of algorithms — TGA/Parsing

- Brainerd, 1967 & 1969
- Kron, 1975
- Hoffmann & O’Donnell, 1982
- Hatcher, 1985; Hatcher & Christopher, 1986
- Turner, 1986
- van Dinther, 1987
- Chase, 1987
- Aho, Ganapathi & Tjang, 1985, 1988
- van de Meerakker, 1988
- Weisgerber & Wilhelm, 1989
- Hemerik & Katoen, 1989
- Balachandran, Dhamdhere & Biswas, 1990
- Ferdinand, Seidl & Wilhelm, 1994
- Wilhelm & Mauer, 1995
- Comon et al., 2003
- Cleophas, Hemerik & Zwaan, 2005 & 2006
What is a taxonomy?

• Classification of *algorithms* based on *essential details*
• Root: high-level solution, with specification
• Others derived by adding *details*  
  – refined solution to same/similar problem  
  – algorithms from literature, new algorithms  
  – correctness arguments
• Goals/benefits  
  – Clear, consistent, and correct presentation  
  – Eases comparison, discovery and teaching  
  – Helps in subsequent toolkit construction
Tree Algorithms: Two Taxonomies and a Toolkit - FASTAR/Espresso Workshop, Pretoria, SA, October 22 – 23, 2007

Tree Acceptance Taxonomy

van Dinther, 1987

Brainerd, 1967 & 1969
Turner, 1986
van Dinther, 1987
Weisgerber & Wilhelm, 1989
Hemerik & Katoen, 1989
Ferdinand, Seidl & Wilhelm, 1994
Wilhelm & Mauer, 1995

Chase, 1987
Hemerik & Katoen, 1989
Ferdinand, Seidl & Wilhelm, 1994

Aho, Ganapathi & Tjang, 1985, 1988
van de Meerakker, 1988
Weisgerber & Wilhelm, 1989
Ferdinand, Seidl & Wilhelm, 1994
Wilhelm & Mauer, 1995
Cleophas, Hemerik & Zwaan, 2005 & 2006
What are (Finite) Tree Automata?

- **Recall** *Finite Automaton*
  - Symbol transitions relate two states
    - Due to symbol arity of 1
- *(Finite) Tree Automaton is generalization*
  - Symbol transitions relate state and tuple of states
    - Due to symbol arity of at least 1
  - Transitions/proc. undirected, root-to-frontier, frontier-to-root
  - Nondeterministic with/without $\varepsilon$-transitions, deterministic
  - Equivalence between types, *except for DRFTA*
What are (Finite) Tree Automata? — Example

- Symbol transitions relate state and tuple of states
- Nondeterminism and \( \varepsilon \)-transitions
- `Leaf states’
- Root state \( q_s \)

\[
R_a = \{ (q_3, (q_1, q_2)), (q_4, (q_5, q_1)) \} \quad R_c = \{ (q_6, ()) \} \quad R_\varepsilon = \{ (q_s, q_3), (q_s, q_4), (q_s, q_6), (q_1, q_s), (q_1, q_0), (q_1, q_2) \}
\]

\[
R_b = \{ (q_5, (q_6)), (q_0, (q_1)) \} \quad R_d = \{ (q_2, ()) \}
\]
What are (Finite) Tree Automata? — Example

- Visual depiction, similar to string automata
What are (Finite) Tree Automata? — Example

- Directed Frontier-to-Root
- Start `below leaf states'
- End at root acc. state
One algorithm path — I

T-ACCEPTOR

MATCH-SET

S-PATH

RF

FR

REC

SP-MATCHER

DET

TABULATE

FILTER

DET

ACA-SPM

DRFTA-SPM

FILTER

TABULATE

TFILT

SFILT

IFILT

CFILT

Tree Algorithms: Two Taxonomies and a Toolkit - FASTAR/Espresso Workshop, Pretoria, SA, October 22 – 23, 2007
One algorithm path — II

\[
\begin{align*}
&\text{"\text{const } G = \ldots; \quad t : \ldots; \quad \text{var } b : B\text{"} } \\
&\mid \quad b := t \in \mathcal{L}(G) \\
&\] \\
\text{(\text{T-ACCEPTOR})}
\end{align*}
\]
One algorithm path — II

(T-ACCEPTOR, FR)

const \( \mathcal{G} = \ldots \);
\( t: \ldots \);
var \( b: \mathbb{B} \)

\( \textbf{let } M = \ldots \) be an \( \varepsilon \)NFRTA such that \( \mathcal{L}(M) = \mathcal{L}(\mathcal{G}) \);

\( b := \text{Traverse}(\varepsilon) \cap Q_{ra} \neq \emptyset \)

\( \textbf{func } \text{Traverse}(\downarrow n : D) : \mathcal{P}(Q) = \)

\[ \begin{array}{l}
\text{var } s_1, \ldots, s_n : \mathcal{P}(Q) \\
\textbf{let } a = t(n); \\
\text{if } n > 0 \rightarrow \\
\quad \text{for } i : 1 \leq i \leq n \rightarrow \\
\quad \quad s_i := \text{Traverse}(n \cdot i) \\
\quad \text{rof}; \\
\text{Traverse} := \emptyset; \\
\text{for } (q_1, \ldots, q_n) : q_1 \in s_1, \ldots, q_n \in s_n \rightarrow \\
\quad \text{Traverse} := \text{Traverse} \cup \mathcal{R}^*_{\varepsilon}(R_a(q_1, \ldots, q_n)) \\
\quad \text{rof} \\
\text{fi} \\
\quad n = 0 \rightarrow \\
\quad \text{Traverse} := \mathcal{R}^*_{\varepsilon}(R_a()) \\
\text{fi} \\
\end{array} \]
One algorithm path — III

(T-ACCEPTOR, FR, DET)

\[
\begin{align*}
\text{let } M = \ldots & \text{ be a DFRTA such that } \mathcal{L}(M) = \mathcal{L}(G); \\
\text{let } b = \text{Traverse}(\varepsilon) & \in Q_{ra}
\end{align*}
\]

\textbf{func} Traverse(\downarrow n : D) : Q =

\[
\begin{align*}
\text{let } a &= t(n); \\
\text{if } n > 0 \rightarrow \\
\text{Traverse} &= R_a(\text{Traverse}(n \cdot 1), \ldots, \text{Traverse}(n \cdot n)) \\
\text{if } n = 0 \rightarrow \\
\text{Traverse} &= R_a()
\end{align*}
\]
One algorithm path — TA constructions

• 1 basic, undirected construction
• 3 different state sets; correspond to the set of all subtrees of $RTG$ production rules, or reduction
• Add direction: undirected, RF, FR
• Remove $\varepsilon$-transitions
• Make deterministic by subset construction
• For $DFRTA$, additional filtering to reduce transition tables, based on $RTG$ structure
• Over 30 constructions in total
Two Taxonomies
Taxonomies — Some Results

• Two taxonomies closely related, mostly same details
• Provide factoring, show commonalities and variations
• Separation of TA use from construction
• Constructions
  – Start with basic, undirected TA; stepwise more complex
• Filtering functions
  – Most reduction by most complex filter (*Chase, 1987*)
    • Description/terminology hard to follow
  – Space saving almost as high in two new filters
  – Simplest filter already appeared in (*Turner, 1986*)
    • Unclear from description, hardly referenced by others
    • Patented in Canada!
Toolkit

• Why?
  – Hard to find implementations, let alone collections of them
  – Compare performance, verify assumptions on algorithms made in literature
  – Use to experiment and gain insight

• Design based on taxonomies
  – Factoring, commonalities and variations suggest design
  – Implementation easy given abstract algorithms
Toolkit vs Taxonomy — (T-ACCEPTOR, FR, DET)

```java
private static AbstractAutomatonState Traverse(AbstractDFRTA M, Node n) {
    AbstractTAState[] childStates = new AbstractTAState[n.children().size()];
    for (int i=0; i < n.children().size(); i++)
        childStates[i] = Traverse(M, n.children().get(i));

    if (n.children().size() > 0) {
        state = M.nextState(childStates, (RankedSymbol)n.symbol());
    } else {
        state = M.nextState(childStates, (RankedSymbol)n.symbol());
    }

    return state;
}
```
Toolkit — Some Results

- Implements
  - Basic datastructures (trees, alphabets, tree grammars, tree automata)
  - Most of algorithms near/at acceptance taxonomy leaves
  - Transformation algorithms for RTGs, TAs (chain rule removal, epsilon transition removal)

- Complemented by GUI to use and experiment

- Results
  - Total of ca. 6000 loc for toolkit, 6000 for GUI
  - Most work was in selecting efficient representations for the basic data structures
  - Adding algorithms given basic data structures quite easy
    - E.g. extension with tree parsing algorithms done in 2 hours
Concluding Remarks

• Classification & implementation of tree algorithms
• Sketched historical setting, motivation
• Two closely related taxonomies
  – Classify algorithms
  – Give additional insight
  – Lead to new results
• Taxonomy-based toolkit of algorithms
  – To provide usable implementations
  – To compare algorithms in practice
  – To gain more insight
Current & Future Work

- Completing toolkit
  - Adding algorithms
  - More experiments/benchmarking
  - Packaging
- Tree parsing
  - Create taxonomy based on acceptance one
  - Extend toolkit
  - Current toolkit already has two parsing algorithms
    - Implemented in two hours, tens of lines