Taxonomies and toolkits of regular tree language algorithms

Loek Cleophas
loek@loekcleophas.com
http://www.win.tue.nl/~lcleopha

Software Engineering & Technology Group
Department of Mathematics & Computer Science

2IS95 SET seminar
September 18th 2008
Overview

• Context
  *Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits*

• Domain
  *Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms*

• Taxonomies
  *Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms*

• Toolkits
  *Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms*

• Concluding Remarks & Research Problems
Overview

• Context
  Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits

• Domain
  Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms

• Taxonomies
  Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Toolkits
  Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Concluding Remarks & Research Problems
Context

• SET research themes:
  • Model-driven software engineering
  • Verified software engineering
  • Generic language technology
    – Theory, tools and applications
    – Taxonomies and toolkits of formal language algorithms
      – Taxonomies and toolkits of regular tree language algorithms
• **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)*
  • ...
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
- Separation between theory and practice
- Hard to compare and choose implementations

• **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):**
    Given a regular tree grammar and an input tree, determine whether the input tree is in the language generated by the grammar.
  • **Tree Pattern Matching (TPM)**
  • ...
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
- Separation between theory and practice
- Hard to compare and choose implementations

Taxonomies and toolkit (Cleophas, Hemerik & Zwaan, 2005/2006; Strolenberg, 2007; Cleophas, 2008)
Context
Domain Deficiencies & Taxonomies

- Inaccessibility of theory and algorithms
- Difficulty of comparing algorithms
  - Difference in style
  - Difference in formality
- Distance between theory & practice
- Lack of large collection of implementations
- Difficulty of choosing between algorithms

*Taxonomy* shows commonality & variation in algorithm structure & data representation, discusses theoretical complexity, not actual performance
Context
Domain Deficiencies & Toolkits

- Inaccessibility of theory and algorithms
- Difficulty of comparing algorithms
  - Difference in style
  - Difference in formality
- Distance between theory & practice
- Lack of large collection of implementations
- Difficulty of choosing between algorithms

*Toolkit & GUI* give insight into algorithm properties, performance; starting point in solving last two deficiencies
Overview

• Context
  Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits

• Domain
  Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms

• Taxonomies
  Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Toolkits
  Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Concluding Remarks & Research Problems
• Algorithms ...
  • for acceptance, pattern matching, parsing
  • on node-labeled, ordered, ranked trees
• Usable in ...
  • compilers, for instruction selection
  • term rewriting
• Algorithms ...
  • for acceptance, pattern matching, parsing
  • on node-labeled, ordered, ranked trees
• Usable in ...
  • compilers, for instruction selection
  • term rewriting
• Generalization of regular string grammar
  • Recall right regular string grammar production forms
    \[ A \rightarrow wB, \quad A \rightarrow w \quad (w \in \Sigma^*) \]
• Regular tree grammar
  • Form \( A \rightarrow t \) with \( t \) a tree, nonterminals at leaves

\begin{align*}
(1) \quad S & \rightarrow a \quad B \quad d \\
(2) \quad S & \rightarrow a \quad b \quad B \\
(3) \quad S & \rightarrow c \\
(4) \quad B & \rightarrow b \\
(5) \quad B & \rightarrow S \\
(6) \quad B & \rightarrow d
\end{align*}
Domain
Computing match sets; tree automata

Idea: Compute matching subpatterns \( \rightarrow \) includes patterns
Idea: Do so recursively
Idea: Precompute, tabulate \( \rightarrow \) Det. FR tree automaton
• Need to find instances of rewrite rules’ left hand sides in subject

• Tree Pattern Matching!

\(\text{and} \quad (1) \quad \text{and} \rightarrow T\)

\(T \quad T\)

\(\text{and} \quad (2) \quad \text{and} \rightarrow F\)

\(F \quad \nu\)

\(\text{and} \quad (3) \quad \text{and} \rightarrow F\)

\(\nu \quad F\)

\(\text{and} \quad (4) \quad \text{and} \rightarrow \text{and}\)

\(\nu_1 \quad \nu_2 \quad \nu_3\)

\(\nu_1 \quad \nu_2\)
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!

Domain
Application: Instruction Selection

1. $R_i \rightarrow c$
   \[ MOV \ #c, R_i \]

2. $R_i \rightarrow M$
   \[ R_j \]
   \[ MOV \ * R_j, R_i \]

3. $R_i \rightarrow M$
   \[ + \]
   \[ c \]
   \[ R_j \]
   \[ MOV \ c(R_j), R_i \]

4. $R_i \rightarrow +$
   \[ R_i \]
   \[ R_j \]
   \[ ADD \ R_i, R_j \]

\[ MOV \ c_2(R_1), R_2 \]
\[ MOV \ #c_1, R_3 \]
\[ ADD \ R_3, R_2 \]
Domain
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
- Cleophas, 2008
Overview

• Context
  Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits

• Domain
  Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms

• Taxonomies
  Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Toolkits
  Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Concluding Remarks & Research Problems
Taxonomies

Biological Taxonomies

• Classify *organisms*
• From abstract, general to concrete, specific
• Properties (details) explicit
• Allow comparison
Taxonomies
Algorithm Taxonomies

- Algorithm taxonomies classify algorithms
- From abstract, general to concrete, specific
- Properties (details) explicit
- Allows comparison
Taxonomies
Algorithm Taxonomy – I

• Classification of *algorithms* based on their *essential details*
• Algorithms in it solve one algorithmic problem
• Depicted as tree/directed acyclic graph
  • Nodes refer to algorithms
  • Branches refer to details
• Root represents high-level algorithm
  • Correctness easily shown
Taxonomies
Algorithm Taxonomy – II

• Adding *detail* to algorithm
  • Depicted by branch connecting algorithm node to new child node
  • Obtains refinement or variation
  • Leads to algorithms from literature, new algorithms
  • Associated correctness arguments

• Algorithm correctness follows from correctness of root and of details added
Taxonomies

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
Cleophas, 2008

Aho, Ganapathi & Tjang, 1985, 1988
van de Meerakker, 1988
Weisgerber & Wilhelm, 1989
Ferdinand, Seidl & Wilhelm, 1994
Wilhelm & Mauer, 1995
Cleophas, Hemerik & Zwaan, 2005 & 2006
Taxonomies
One Algorithm Path — I
\begin{align*}
&\text{()}
\\
&\begin{array}{c}
\text{[ const } G = \ldots; \\
\quad t : \ldots; \\
\var b : \mathbb{B} \\
\mid b := t \in \mathcal{L}(G) \\
\end{array}
\\
&\text{[ (T-ACCEPTOR)}
\\
&\begin{array}{c}
\text{[ const } G = \ldots; \\
\quad t : \ldots; \\
\var b : \mathbb{B} \\
\mid \text{let } M = \ldots \text{ be a TA such that } \mathcal{L}(M) = \mathcal{L}(G); \\
\mid b := t \in \mathcal{L}(M) \\
\end{array}
\end{align*}
(T-ACCEPTOR, FR)

\[
\begin{array}{l}
\text{[ [ } \text{ const } G = \ldots ; \\
\quad t : \ldots ; \\
\text{ var } b : \mathbb{B} \\
\quad \text{ let } M = \ldots \text{ be an } \varepsilon\text{-NFRTA such that } \mathcal{L}(M) = \mathcal{L}(G); \\
\quad b := \text{Traverse}(\varepsilon) \cap Q_{ra} \neq \emptyset \\
\text{ func } \text{Traverse}(\downarrow n : D) : \mathcal{P}(Q) = \\
\quad [ [ \text{ var } s_1, \ldots, s_n : \mathcal{P}(Q) \\
\quad \text{ let } a = t(n); \\
\quad \text{ if } n > 0 \rightarrow \\
\quad \quad \text{ for } i : 1 \leq i \leq n \rightarrow \\
\quad \quad \quad s_i := \text{Traverse}(n \cdot i) \\
\quad \quad \text{ rof; } \\
\quad \text{ Traverse } := \emptyset; \\
\quad \text{ for } (q_1, \ldots, q_n) : q_1 \in s_1, \ldots, q_n \in s_n \rightarrow \\
\quad \quad \text{ Traverse } := \text{Traverse} \cup R^*_\varepsilon(R_a(q_1, \ldots, q_n)) \\
\quad \text{ rof } \\
\quad \text{ if } n = 0 \rightarrow \\
\quad \text{ Traverse } := R^*_\varepsilon(R_a()) \\
\text{ fi } \\
\quad ] ] \\
\end{array}
\]
(T-ACCEPTOR, FR, DET)

[[
  \textbf{const} \ G = \ldots ;\\
  \ t : \ldots ;\\
  \textbf{var} \ b : \mathbb{B}\\
  | \textbf{let} \ M = \ldots \ \text{be a DFRTA such that} \ \mathcal{L}(M) = \mathcal{L}(G)\\
  \ b : = \text{Traverse}(\varepsilon) \in Q_{ra}\\

  \textbf{func} \ Traverse(\downarrow n : D) : Q =\\
  [\ [ \textbf{var} \ q_1, \ldots , q_n : Q\\
  | \textbf{let} \ a = t(n)\\
  | \textbf{if} \ n > 0 \rightarrow\\
  \ \ \ \text{Traverse} : = R_a(\text{Traverse}(n \cdot 1) , \ldots , \text{Traverse}(n \cdot n))\\
  | \textbf{fi}\\
  ] ] ]]
Taxonomies
One Algorithm Path — Tree Automaton 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 $RTG$/patterns’ structure
• Over 30 constructions in total
• Similar kind/amount for tree pattern matching taxonomy
Taxonomies
New filters $\rightarrow$ new algorithms

Turner, 1986
Chase, 1987
Hemerik & Katoen, 1989
Ferdinand, Seidl & Wilhelm, 1994

Cleophas, 2008
Overview

• Context
  Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits

• Domain
  Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms

• Taxonomies
  Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Toolkits
  Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Concluding Remarks & Research Problems
Toolkit
Motivation

• 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
func Traverse(\( n : D \)) : Q =
|| var \( q_1, \ldots, q_n : Q \)
| let \( a = t(n) \);
| if \( n > 0 \)
| \( Traverse := R_a(Traverse(n \cdot 1), \ldots, Traverse(n \cdot n)) \)
| \( n = 0 \rightarrow \)
| \( Traverse := R_a() \)
| fi
||

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

Content

- Representations of ...
  - trees, tree patterns, regular tree grammars, tree automata
- Tree pattern matching, acceptance, parsing algorithms
  - most algorithms near taxonomy leafs
  - corresponding automata generators/tabulators
- Analyses, transformations (chain rule removal, epsilon transition removal)
- Implementation
  - *Forest FIRE* toolkit 82 interfaces/classes, ~8800 LOC
  - *FIRE Wood* GUI 56 interfaces/classes, ~7400 LOC
  - *Java, SWT*
  - *OS X, Windows XP, Linux*
Toolkit

Some Results

• Provides large, coherent set of implementations
  • Simplify choice, use
• Most work was in selecting efficient representations for the basic data structures
• Adding algorithms given basic data structures quite easy
  • Pseudo-code part of algorithms’ presentation in taxonomies
  • Even further extensions: two tree parsing algorithms done in 2 hours
• New practical algorithms
  • Two new filters outperforming existing ones
Toolkit

Results: Match set automata for instruction selection

- Intel X86 CPU

![Graph showing tabulation time (ms) and memory use (MiB)]
Demo?: GUI, match set tabulation, matching
Demo: *Instruction selection using tree parsing*
Overview

• Context
  Regular string and tree language theory, domain deficiencies, role of taxonomies & toolkits

• Domain
  Tree algorithms, trees, tree pattern matching, tree grammars, match sets & tree automata, applications, algorithms

• Taxonomies
  Algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Toolkits
  Motivation, toolkit vs. taxonomy, tree toolkit content, algorithm taxonomies, taxonomies of tree acceptance and tree pattern matching algorithms

• Concluding Remarks & Research Problems
Concluding Remarks

• Classification & implementation of tree algorithms
  • Increase accessibility, link theory and practice
  • Two closely related taxonomies
    – Classify algorithms, show commonalities and differences
    – Give additional insight
    – Lead to new algorithms
• Taxonomy-based toolkit of algorithms
  – Provides usable implementations
  – Allowed comparing algorithms in practice
  – Gave additional insight
Research problems

- Tree parsing
- Tree automata minimization
- Tree automata construction based on tree regular expressions
- Combining different tree pattern matching algorithms and different term rewriting strategies
  - vs. just leftmost-innermost; simplifies specification?
- Tree transducer construction