Taxonomies of Regular Tree Algorithms

Loek Cleophas                     Kees Hemerik
loek@loekcleophas.com
http://www.loekcleophas.com

FASTAR Research Group
Department of Computer Science
Faculty of Engineering, Built environment & IT

PSC 2009, Prague, September 2nd 2009
Overview

• Context
  History & relevance, deficiencies, role of taxonomies & toolkits

• Domain
  Trees, patterns & matching, regular tree grammars

• Taxonomies
  Algorithm taxonomies, taxonomies of regular tree algorithms

• Tree Acceptance Taxonomy
  Algorithms b/o tree automata, match sets, stringpath matching

• Concluding Remarks
Overview

• Context
  *History & relevance, deficiencies, role of taxonomies & toolkits*

• Domain
  *Trees, patterns & matching, regular tree grammars*

• Taxonomies
  *Algorithm taxonomies, taxonomies of regular tree algorithms*

• Tree Acceptance Taxonomy
  *Algorithms b/o tree automata, match sets, stringpath matching*

• Concluding Remarks
• Regular Tree Grammar (RTG), Finite Tree Automaton (TA), Regular Tree Expression
Regular Tree Language Theory

• Regular Tree Grammar (RTG), Finite Tree Automaton (TA), Regular Tree Expression

• Algorithmic Problems
  • Membership/Tree Acceptance
  • Tree Pattern Matching
  • Tree Parsing
Regular Tree Language Theory

• **Regular Tree Grammar (RTG), Finite Tree Automaton (TA), Regular Tree Expression**

• **Algorithmic Problems**
  • Membership/Tree Acceptance
  • Tree Pattern Matching
  • Tree Parsing

• **1960s**
  • Equivalence between formalisms (except Deterministic Top-Down/Root-to-Frontier TA), transformations between them
  • Construct and use TA based on RTG or pattern set
Regular Tree Language Theory

- **Regular Tree Grammar (RTG), Finite Tree Automaton (TA), Regular Tree Expression**

- **Algorithmic Problems**
  - Membership/Tree Acceptance
  - Tree Pattern Matching
  - Tree Parsing

- **1960s**
  - Equivalence between formalisms (except Deterministic Top-Down/Root-to-Frontier TA), transformations between them
  - Construct and use TA based on RTG or pattern set

- **Since ca. 1975**
  - Applications in instruction selection, term rewriting, model checking
  - Many TA constructions, algorithms
Context
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
• **Domain deficiencies**
  • inaccessible, difficult to find
  • difficult to compare, choose
  • separation between theory and practice
Domain deficiencies
  • inaccessible, difficult to find
  • difficult to compare, choose
  • separation between theory and practice

.. yet
  • well-established theory
  • algorithmic problems related, with related solutions

Tree Acceptance, Tree Pattern Matching (TPM), Tree Parsing
• Domain deficiencies
  • inaccessible, difficult to find
  • difficult to compare, choose
  • separation between theory and practice
• .. yet
  • well-established theory
  • algorithmic problems related, with related solutions
    Tree Acceptance, Tree Pattern Matching (TPM), Tree Parsing
• hence
  • taxonomies (Cleophas, Hemerik & Zwaan, 2005/2006; Cleophas, 2008)
    systematic classifications of problems & solutions in (algorithmic) domain,
    to bring order to the domain
  • toolkit (Strolenberg, 2007; Cleophas, 2008)
    taxonomy-based
Overview

• Context
  History & relevance, deficiencies, role of taxonomies & toolkits

• Domain
  Trees, patterns & matching, regular tree grammars

• Taxonomies
  Algorithm taxonomies, taxonomies of regular tree algorithms

• Tree Acceptance Taxonomy
  Algorithms b/o tree automata, match sets, stringpath matching

• Concluding Remarks
• **Node-labeled, ordered, ranked trees**

• Generalization of strings: Allow symbols of arity/rank > 1
  • Fixed arity per symbol
• *Node-labeled, ordered, ranked trees*

• Generalization of strings:
  Allow symbols of arity/rank > 1
  • Fixed arity per symbol
  • Order of siblings relevant
Domain
Trees, Patterns, Tree Pattern Matching

- **Node-labeled, ordered, ranked trees**
- Generalization of strings:
  Allow symbols of arity/rank > 1
  - Fixed arity per symbol
  - Order of siblings relevant
- Tree patterns with wildcards at leaves
Domain
Regular Tree Grammars

- Generalization of regular string grammar
- Recall right regular string grammar production forms

\[ A \rightarrow wB, \ A \rightarrow w \ (w \in \Sigma^*) \]
Domain
Regular Tree Grammars

- Generalization of regular string grammar
- Recall right regular string grammar production forms
  \[ A \rightarrow wB, A \rightarrow w \quad (w \in \Sigma^*) \]
- Regular tree grammar
  - Form \( A \rightarrow t \) with \( t \) a tree, nonterminals at leaves
Domain
Regular Tree Grammars

- 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

1. \[ S \rightarrow a, \quad \begin{array}{c}
    B \\
    d
  \end{array} \]
2. \[ S \rightarrow a, \quad \begin{array}{c}
    b \\
    B \\
    c
  \end{array} \]
3. \[ S \rightarrow c, \]
4. \[ B \rightarrow b, \quad \begin{array}{c}
    B
  \end{array} \]
5. \[ B \rightarrow S, \]
6. \[ B \rightarrow d \]
Domain
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) \( S \rightarrow a \),

(3) \( S \rightarrow c \),

(4) \( B \rightarrow b \),

(5) \( B \rightarrow S \),

(6) \( B \rightarrow d \)
Overview

• Context
  *History & relevance, deficiencies, role of taxonomies & toolkits*

• Domain
  *Trees, patterns & matching, regular tree grammars*

• Taxonomies
  *Algorithm taxonomies, taxonomies of regular tree algorithms*

• Tree Acceptance Taxonomy
  *Algorithms b/o tree automata, match sets, stringpath matching*

• Concluding Remarks
Taxonomies
Algorithm Taxonomies

• Similar to biological taxonomies
Taxonomies
Algorithm Taxonomies

• Similar to biological taxonomies
• *Algorithm taxonomies classify algorithms*
Taxonomies
Algorithm Taxonomies

• Similar to biological taxonomies
• Algorithm taxonomies classify algorithms
• Depicted as tree/directed acyclic graph
  Nodes refer to algorithms, branches to details
Taxonomies
Algorithm Taxonomies

• Similar to biological taxonomies

• Algorithm taxonomies classify algorithms

• Depicted as tree/directed acyclic graph
  Nodes refer to algorithms, branches to details

• From abstract, general to concrete, specific
Taxonomies
Algorithm Taxonomies

- Similar to biological taxonomies
- Algorithm taxonomies classify algorithms
- Depicted as tree/directed acyclic graph
  Nodes refer to algorithms, branches to details
- From abstract, general to concrete, specific
- Properties (details) explicit
Taxonomies
Algorithm Taxonomies

- Similar to biological taxonomies
- *Algorithm taxonomies* classify *algorithms*
- Depicted as tree/directed acyclic graph
  *Nodes refer to algorithms, branches to details*
- From abstract, general to concrete, specific
- Properties (details) explicit
- Allow comparison, discovery of new algorithms
Taxonomies Construction

• Bottom-up
Taxonomies Construction

- Bottom-up
- Literature survey
Taxonomies Construction

- Bottom-up
- Literature survey
- Rephrase algorithms in common presentation style
Taxonomies Construction

• Bottom-up
• Literature survey
• Rephrase algorithms in common presentation style
• Analyze to determine essential details
Taxonomies Construction

- Bottom-up
- Literature survey
- Rephrase algorithms in common presentation style
- Analyze to determine essential details
- Abstracting over details of algorithms yields common ancestors
Taxonomies
Construction

• Bottom-up
• Literature survey
• Rephrase algorithms in common presentation style
• Analyze to determine essential details
• Abstracting over details of algorithms yields common ancestors
• New combinations may lead to new algorithms
Taxonomies
Presentation & Correctness

• Top-down
Taxonomies
Presentation & Correctness

• Top-down
• Root represents high-level algorithm
  • Correctness easily shown
Taxonomies
Presentation & Correctness

• Top-down
• Root represents high-level algorithm
  • Correctness easily shown
• Adding *detail*
  • Obtains (new) refinement/variation
  • Branch connecting algorithm node to child node
  • Associated correctness arguments
Taxonomies
Presentation & Correctness

• Top-down
• Root represents high-level algorithm
  • Correctness easily shown
• Adding detail
  • Obtains (new) refinement/variation
  • Branch connecting algorithm node to child node
  • Associated correctness arguments
• Correctness of root and of details on rootpath imply correctness of node
Taxonomies
Tree Acceptance Taxonomy

T-ACCEPTOR

MATCH-SET

S-PATH

RF

FR

REC

SP-MATCHER

DET

DET

TABULATE

FILTER

DET

TABULATE

ACA-SPM

DRFTA-SPM

FILTER

TFILT

SFILT

IFILT

CFILT
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
Overview

• Context
  *History & relevance, deficiencies, role of taxonomies & toolkits*

• Domain
  *Trees, patterns & matching, regular tree grammars*

• Taxonomies
  *Algorithm taxonomies, taxonomies of regular tree algorithms*

• Tree Acceptance Taxonomy
  *Algorithms b/o tree automata, match sets, stringpath matching*

• Concluding Remarks
Tree Acceptance Taxonomy
Overview - I
Algorithms based on correspondence between Regular Tree Grammars and Finite Tree Automata
• Algorithms based on correspondence between Regular Tree Grammars and Finite Tree Automata
• For every RTG, undirected TA can be constructed
• Algorithms based on correspondence between Regular Tree Grammars and Finite Tree Automata
• For every RTG, undirected TA can be constructed
• Adding details e.g. direction, determinacy, restricting grammar elements used for state set leads to other constructions
Tree Acceptance Taxonomy
String Automata, Tree Automata

a b b
• View automata as assigning states to positions ‘in between’ symbols
• View automata as assigning states to positions ‘in between’ symbols

\[
\begin{array}{cccc}
q_0 & q_1 & q_2 & q_3 \\
\end{array}
\]

\[(q_0, q_1) \in R_a \text{ etc.}\]
View automata as assigning states to positions ‘in between’ symbols.

\[ (q_0, q_1) \in R \text{ etc.} \]
View automata as assigning states to positions ‘in between’ symbols
• View automata as assigning states to positions ‘in between’ symbols

• For tree case, assigning states to positions of tree

\[
\begin{array}{c}
q_0 \ x \ q_1 \ x \ q_2 \ x \ q_3 \\
| \\
q_0 \\
| \\
q_1 \\
| \\
q_2 \\
| \\
q_3 \\
| \\
\bot
\end{array}
\]

\[
(q_0, q_1) \in R_a \text{ etc.}
\]

Note \((q_3, (\ )) \in R_{\bot}\)
• View automata as assigning states to positions ‘in between’ symbols

• For tree case, assigning states to positions of tree
• View automata as assigning states to positions ‘in between’ symbols
• For tree case, assigning states to positions of tree
• As for string case
  • construct $TA$ from $RTG$
  • use this $TA$ to solve the tree acceptance problem
• As for string case
  • construct $TA$ from $RTG$
  • use this $TA$ to solve the tree acceptance problem
• Basic idea: states of nondeterministic $TA$ correspond to subtrees of grammar production right hand sides
Tree Acceptance Taxonomy
String Automata, Tree Automata

• As for string case
  • construct \( TA \) from \( RTG \)
  • use this \( TA \) to solve the tree acceptance problem
• Basic idea: states of nondeterministic \( TA \) correspond to subtrees of grammar production right hand sides
• A few slightly different subtree sets
Tree Acceptance Taxonomy
String Automata, Tree Automata

• As for string case
  • construct \( TA \) from \( RTG \)
  • use this \( TA \) to solve the tree acceptance problem
• Basic idea: states of nondeterministic \( TA \) correspond to subtrees of grammar production right hand sides
• A few slightly different subtree sets
• Direction
  • Top-Down (Root-to-Frontier)
Tree Acceptance Taxonomy
String Automata, Tree Automata

• As for string case
  • construct $TA$ from $RTG$
  • use this $TA$ to solve the tree acceptance problem
• Basic idea: states of nondeterministic $TA$ correspond to subtrees of grammar production right hand sides
• A few slightly different subtree sets
• Direction
  • Top-Down (Root-to-Frontier)
  • Bottom-Up (Frontier-to-Root)
Tree Acceptance Taxonomy
String Automata, Tree Automata

• As for string case
  • construct TA from RTG
  • use this TA to solve the tree acceptance problem
• Basic idea: states of nondeterministic TA correspond to subtrees of grammar production right hand sides
• A few slightly different subtree sets
• Direction
  • Top-Down (Root-to-Frontier)
  • Bottom-Up (Frontier-to-Root)
Tree Acceptance Taxonomy
String Automata, Tree Automata

- As for string case
  - construct $TA$ from $RTG$
  - use this $TA$ to solve the tree acceptance problem
- Basic idea: states of nondeterministic $TA$ correspond to subtrees of grammar production right hand sides
- A few slightly different subtree sets
- Direction
  - Top-Down (Root-to-Frontier)
  - Bottom-Up (Frontier-to-Root)
- Epsilon-removal
Tree Acceptance Taxonomy
String Automata, Tree Automata

- As for string case
  - construct $TA$ from $RTG$
  - use this $TA$ to solve the tree acceptance problem
- Basic idea: states of nondeterministic $TA$ correspond to subtrees of grammar production right hand sides
- A few slightly different subtree sets
- Direction
  - Top-Down (Root-to-Frontier)
  - Bottom-Up (Frontier-to-Root)
- Epsilon-removal
- Determinization
Tree Acceptance Taxonomy
String Automata, Tree Automata

• As for string case
  • construct $TA$ from $RTG$
  • use this $TA$ to solve the tree acceptance problem
• Basic idea: states of nondeterministic $TA$ correspond to subtrees of grammar production right hand sides
• A few slightly different subtree sets
• Direction
  • Top-Down (Root-to-Frontier)
  • Bottom-Up (Frontier-to-Root)
• Epsilon-removal
• Determinization
• For $DFRTA$, 4 types of filtering to reduce tables
• About 50 different constructions in tree acceptance and tree pattern matching taxonomies
Tree Acceptance Taxonomy
Tree Automata Constructions

- About 50 different constructions in tree acceptance and tree pattern matching taxonomies
- Construction presentation
  - uniform style
  - defines state set, transition relation, ...
  - gives example
  - discusses correctness arguments
  - discusses related constructions and literature
  - identified by sequence of labels indicating details, e.g. (TGA-TA:ALL-SUB:REM-Epsilon:FR:SUBSET)
Tree Acceptance Taxonomy
Tree Automata Constructions

• Basic construction (TGA-TA:ALL-SUB)
  • RF and FR variants appear in literature - van Dinther (1987)
• Basic construction (TGA-TA:ALL-SUB)
  • RF and FR variants appear in literature - van Dinther (1987)
  • Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
Tree Acceptance Taxonomy
Tree Automata Constructions

- Basic construction (TGA-TA:ALL-SUB)
  - RF and FR variants appear in literature - van Dinther (1987)
  - Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
  - Restricted state set to prevent unreachable states - Ferdinand et al. (1994)
Basic construction (TGA-TA:ALL-SUB)
- RF and FR variants appear in literature - van Dinther (1987)
- Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
- Restricted state set to prevent unreachable states - Ferdinand et al. (1994)
- RTG productions of form $A \rightarrow a(A_1, \ldots, A_n)$ only - FR version in Gecseg & Steinby (1984)
• Basic construction (TGA-TA:ALL-SUB)
  • RF and FR variants appear in literature - van Dinther (1987)
  • Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
  • Restricted state set to prevent unreachable states - Ferdinand et al. (1994)
  • RTG productions of form $A \rightarrow a(A_1, ..., A_n)$ only - FR version in Gecseg & Steinby (1984)
  • Additionally of form $A \rightarrow B$ - FR version already in Brainerd (1969), RF version in Comon et al. (2007)
Tree Acceptance Taxonomy
Tree Automata Constructions

• Basic construction (TGA-TA:ALL-SUB)
  • RF and FR variants appear in literature - van Dinther (1987)
  • Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
  • Restricted state set to prevent unreachable states - Ferdinand et al. (1994)
    • RTG productions of form $A \rightarrow a(A_1, ..., A_n)$ only - FR version in Gecseg & Steinby (1984)
    • Additionally of form $A \rightarrow B$ - FR version already in Brainerd (1969), RF version in Comon et al. (2007)
  • RF variants with SUBSET do not appear - restricted power
Tree Acceptance Taxonomy
Tree Automata Constructions

- **Basic construction (TGA-TA:ALL-SUB)**
  - RF and FR variants appear in literature - van Dinther (1987)
  - Applying REM-Epsilon inside construction - Ferdinand et al. (1994)
  - Restricted state set to prevent unreachable states - Ferdinand et al. (1994)
  - RTG productions of form $A \rightarrow a(A_1, ..., A_n)$ only - FR version in Gecseg & Steinby (1984)
  - Additionally of form $A \rightarrow B$ - FR version already in Brainerd (1969), RF version in Comon et al. (2007)
- RF variants with SUBSET do not appear - restricted power
- FR variants with SUBSET - in e.g. Ferdinand et al. (1994) Hemerik & Katoen (1990), Chase (1987)
Tree Acceptance Taxonomy
Overview - II
• Algorithms based on suitable generalization of $S \Rightarrow t$
- Algorithms based on suitable generalization of $S \Rightarrow t$
- For each subtree of $t$, compute items $p$ such that $p \Rightarrow t$ - match set
• Algorithms based on suitable generalization of $S \Rightarrow t$
  • For each subtree of $t$, compute items $p$ such that $p \Rightarrow t$ - match set
  • Then $t$ is accepted if and only if its match set contains $S$
• Algorithms based on suitable generalization of $S \Rightarrow^* t$
• For each subtree of $t$, compute items $p$ such that $p \Rightarrow^* t$ - match set
• Then $t$ is accepted if and only if its match set contains $S$
• Algorithms differ in item set used, computation of match sets
• Algorithms based on suitable generalization of $S \Rightarrow t$
  • For each subtree of $t$, compute items $p$ such that $p \Rightarrow t$ - match set
  • Then $t$ is accepted if and only if its match set contains $S$
  • Algorithms differ in item set used, computation of match sets
  • For efficiency, compute recursively, tabulate
• **Algorithms based on suitable generalization of** $S \Rightarrow t$
  • For each subtree of $t$, compute *items* $p$ such that $p \Rightarrow t$ - *match set*
  • Then $t$ is accepted if and only if its match set contains $S$
  • Algorithms differ in item set used, computation of match sets
  • For efficiency, compute recursively, tabulate
  • Just a different view on TAs!
Tree Acceptance Taxonomy

Computing match sets; tree automata
Tree Acceptance Taxonomy

Computing match sets; tree automata
Tree Acceptance Taxonomy

Computing match sets; tree automata
Tree Acceptance Taxonomy

Computing match sets; tree automata
Tree Acceptance Taxonomy

Computing match sets; tree automata
Tree Acceptance Taxonomy

*Computing match sets; tree automata*

\[
\begin{align*}
&\text{b} \\
&\text{c} \\
&\text{a} \\
&\text{c} \\
&\text{b} \\
&\text{c} \\
&\text{a} \\
&\text{c} \\
&\text{a} \\
\end{align*}
\]
Tree Acceptance Taxonomy

*Computing match sets; tree automata*

Diagram of tree structures and a state transition diagram.
Tree Acceptance Taxonomy
Practical Results - *Filtering match sets for instruction selection*

- Intel X86 CPU

<table>
<thead>
<tr>
<th>Memory use (MiB)</th>
<th>Tabulation time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unf.</td>
<td>273009</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory use (MiB)</th>
<th>Tabulation time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unf.</td>
<td>144.3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tree Acceptance Taxonomy
Practical Results - Filtering match sets for instruction selection

- Intel X86 CPU

<table>
<thead>
<tr>
<th></th>
<th>Tabulation time (ms)</th>
<th>Memory use (MiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unf.</td>
<td>273009</td>
<td>144.3</td>
</tr>
<tr>
<td>Turner</td>
<td>7398</td>
<td>2.5</td>
</tr>
<tr>
<td>Chase</td>
<td>288</td>
<td>10.1</td>
</tr>
<tr>
<td>Sym</td>
<td>267</td>
<td>4.6</td>
</tr>
<tr>
<td>Index</td>
<td>2883</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Wednesday, September 2, 2009
Tree Acceptance Taxonomy
Overview - III
• Algorithms based on decomposing items into stringpaths, use of string matching
• Algorithms based on decomposing items into stringpaths, use of string matching
• Algorithms based on decomposing items into stringpaths, use of string matching
  • Based on stringpath matches found, item matches and hence match sets can be computed for each subtree of $t$
Algorithms based on decomposing items into stringpaths, use of string matching

- Based on stringpath matches found, item matches and hence match sets can be computed for each subtree of \( t \)
- Different automata may be used for stringpath matching
Overview

• **Context**
  *History & relevance, deficiencies, role of taxonomies & toolkits*

• **Domain**
  *Trees, patterns & matching, regular tree grammars*

• **Taxonomies**
  *Algorithm taxonomies, taxonomies of regular tree algorithms*

• **Tree Acceptance Taxonomy**
  *Algorithms b/o tree automata, match sets, stringpath matching*

• **Concluding Remarks**
• Similar taxonomy of tree pattern matching algorithms
Concluding Remarks

• Similar taxonomy of tree pattern matching algorithms
• Each taxonomy presents algorithms, constructions in common framework
  • Improves accessibility
  • Shows algorithm/construction relations
Concluding Remarks

- Similar taxonomy of tree pattern matching algorithms
- Each taxonomy presents algorithms, constructions in common framework
  - Improves accessibility
  - Shows algorithm/construction relations
- Taxonomy construction involves lot of effort
  - Abstraction, sequential addition of details essential
Concluding Remarks

- Similar taxonomy of tree pattern matching algorithms
- Each taxonomy presents algorithms, constructions in common framework
  - Improves accessibility
  - Shows algorithm/construction relations
- Taxonomy construction involves lot of effort
  - Abstraction, sequential addition of details essential
- Lead to new/rediscovered algorithms/constructions
Concluding Remarks
Concluding Remarks

• Form starting point for coherent toolkit
  • Taxonomy hierarchy determines toolkit’s class/interface hierarchy
  • Abstract algorithms included lead to straightforward implementations
Concluding Remarks

- Form starting point for coherent toolkit
  - Taxonomy hierarchy determines toolkit’s class/interface hierarchy
  - Abstract algorithms included lead to straightforward implementations
- Main effort on toolkit was in choice of representations
Concluding Remarks

• Form starting point for coherent toolkit
  • Taxonomy hierarchy determines toolkit’s class/interface hierarchy
  • Abstract algorithms included lead to straightforward implementations
• Main effort on toolkit was in choice of representations
• Algorithms & automata constructions from the taxonomies, fundamental datastructures & algorithms, tree parsing
Concluding Remarks

- Form starting point for coherent toolkit
  - Taxonomy hierarchy determines toolkit’s class/interface hierarchy
  - Abstract algorithms included lead to straightforward implementations
- Main effort on toolkit was in choice of representations
- Algorithms & automata constructions from the taxonomies, fundamental datastructures & algorithms, tree parsing

Implementation
- *Forest FIRE* toolkit, *FIRE Wood* GUI; 138 interfaces/classes, ~16K LOC
- *Java*, *SWT*, multi-platform
- Available via [http://www.fastar.org](http://www.fastar.org)