Growing a Syntax

Eric Allen¹, Ryan Culpepper², Janus Dam Nielsen³, Jon Rafkind⁴, and Sukyoung Ryu¹

¹Sun Microsystems
²Northeastern University
³Aarhus University
⁴University of Utah

Foundations of Object-Oriented Languages, 2009
Outline

1. Growing a Language
   - Fortress Introduction
   - Growing a Language
   - XML Example Introduction

2. Objectives and Example
   - Syntactic Abstraction Goals
   - XML as an Example

3. Syntax Normalization
   - Parsing and Transformation
Outline

1. Growing a Language
   - Fortress Introduction
     - Growing a Language
     - XML Example Introduction

2. Objectives and Example
   - Syntactic Abstraction Goals
   - XML as an Example

3. Syntax Normalization
   - Parsing and Transformation
The Fortress Programming Language

A multicore language for scientists and engineers
The Fortress Programming Language

- A multicore language for scientists and engineers
- Run your whiteboard in parallel!
The Fortress Programming Language

- A multicore language for scientists and engineers
- Run your whiteboard in parallel!

\[ v_{\text{norm}} = \frac{v}{\|v\|} \]
\[ c_{ij} = \sum_{k \leftarrow 0:n} a_{ik} b_{kj} \]
\[ C = A \cup B \]
\[ y = 3x \sin x \cos 2x \log \log x \]
The Fortress Programming Language

- A **multicore language** for scientists and engineers
- Run your **whiteboard** in parallel!

\[
v_{\text{norm}} = \frac{v}{\|v\|}
\]

\[
c_{ij} = \sum_{k \leftarrow 0:n} a_{ik} b_{kj}
\]

\[
C = A \cup B
\]

\[
y = 3x \sin x \cos 2x \log \log x
\]
The Fortress Programming Language

- A **multicore language** for scientists and engineers
- Run your **whiteboard** in **parallel**!

\[
\begin{align*}
    v_{\text{norm}} &= v / \| v \| \\
    c_{ij} &= \sum_{k \leftarrow 0:n} a_{ik} b_{kj} \\
    C &= A \cup B \\
    y &= 3x \sin x \cos 2x \log \log x
\end{align*}
\]

- Growing a Language
  Guy L. Steele Jr., keynote talk OOPSLA 1998
  *Higher-Order and Symbolic Computation* 12, 221-236 (1999)
# Outline

1. **Growing a Language**
   - Fortress Introduction
   - Growing a Language
   - XML Example Introduction

2. **Objectives and Example**
   - Syntactic Abstraction Goals
   - XML as an Example

3. **Syntax Normalization**
   - Parsing and Transformation
“So I think the sole way to win is to plan for growth with help from the users.”

Guy L. Steele Jr.
keynote talk, OOPSLA 1998;
Higher-Order and Symbolic Computation 12, 221-236 (1999)
Consider how a proposed language feature might be provided by a library rather than building features directly into the compiler.

This requires control over both syntax and semantics, not just the ability to add new functions and methods.
Outline

1 Growing a Language
   • Fortress Introduction
   • Growing a Language
   • XML Example Introduction

2 Objectives and Example
   • Syntactic Abstraction Goals
   • XML as an Example

3 Syntax Normalization
   • Parsing and Transformation
Growing a Language

Objectives and Example

Synopsis Normalization

XML Example Introduction

**XML in Fortress**

\[ x = \texttt{<html xmlns="http://www.w3.org/1999/xhtml">} \\
   \texttt{<title> Project Fortress </title>} \\
   \texttt{<body/>} \\
\texttt{</html>} \\
\texttt{<body/>}.hasElements

\[ x.children \\
\texttt{x.attributes} \\
\]

- XML **literals** in Fortress
- Seamless integration
- DOM operations on values
Growing a Language
Objectives and Example
Syntax Normalization
Summary

XML Example Introduction

XML in Fortress

\[
x = \langle \text{html} \text{ xmlns} = \text{"http://www.w3.org/1999/xhtml"} \rangle
  \langle \text{title} \rangle \text{ Project Fortress } \langle /\text{title} \rangle
  \langle \text{body} /\rangle
\langle /\text{html} \rangle
\]

\text{<body/>}.\text{hasElements}
\text{x.children}
\text{x.attributes}

- XML \text{ literals} in Fortress
- \text{Seamless} integration
- \text{DOM operations} on values
XML in Fortress

\[ x = \langle html \ xmlns = \"http://www.w3.org/1999/xhtml\rangle \\
\langle title \rangle \text{Project Fortress} \langle /title \rangle \\
\langle body \rangle \\
\langle /body \rangle \\
\langle /html \rangle \\
\langle body \rangle . \text{hasElements} \\
\langle x.\text{children} \rangle \\
\langle x.\text{attributes} \rangle \\
\]

- XML literals in Fortress
- Seamless integration
- DOM operations on values
Desugaring XML

The XML literal:

```xml
<html xmlns="http://www.w3.org/1999/xhtml">
  <title>Project Fortress</title>
  <body/>
</html>
```

Desugars to:

```xml
Element(
  Header("html",
    ⟨ Attribute("xmlns","http://www.w3.org/1999/xhtml") ⟩),
  ⟨ Element(
    Header("title", ⟨ ⟩),
    ⟨ CData("Project Fortress"), "title"),
    Element(Header("body", ⟨ ⟩), ⟨⟩, "body"), "html")
  ⟩,
  ⟨⟩,
  ⟨⟩,
  "html")
)
object Element(info : Header, contents : List[Content],
        endTag : String) extends Content

    getter name() : String
    getter hasElements() : Boolean
    getter children() : List[Element]
    getter cdata() : CData
    getter attributes() : List[Attribute]
    getter toXml() : String
end

trait Content end

object CData(c : String) extends Content end

object Header(startTag : String, attributes : List[Attribute]) end

object Attribute(key : String, val : String) end
Design Strategy
Consider how a proposed language feature might be provided by a library rather than building features directly into the compiler

XML desugaring is provided by syntactic abstraction in a library
Syntactic Abstraction Goals

- New syntax indistinguishable from core syntax
- Similar syntax for definition/use of a language extension
- Composition of independent language extensions
- Expansion into other language extensions
- Mutually recursive definition of language extensions
Growing a Language

Objectives and Example

Syntax Normalization

Outline

1. Growing a Language
   - Fortress Introduction
   - Growing a Language
   - XML Example Introduction

2. Objectives and Example
   - Syntactic Abstraction Goals
   - XML as an Example

3. Syntax Normalization
   - Parsing and Transformation
Grammar of XML Literals in Fortress

```
grammar xml extends { Expression, Symbols }

Expr ::=  x: XExpr  ⇒  <[x]>
XExpr ::=  
  b: startTag  c: XmlContent  e: endTag  ⇒  <[Element(b, c, e)]>
  |  b: startTag  e: endTag  ⇒  <[Element(b, ⟨⟩, e)]>

XmlContent ::=  
  s: XmlIdentifier  ⇒  <[⟨CDATA(s)⟩]>
  |  {x: XExpr  SPACE}+  ⇒  <[⟨x**⟩]>

startTag ::=  
  <s: XmlIdentifier  {a: Attribute SPACE}*  >  ⇒  <[Header(s, ⟨a**⟩)]>

endTag ::=  
  </s: XmlIdentifier>  ⇒  <[s]>
```

Examples of Goals

- New syntax indistinguishable from core syntax
  
  \[
  \text{<body /> .hasElements}
  \]

- Similar syntax for definition/use of a language extension
  
  \[
  \text{< s : XmlIdentifier \{a : Attribute SPACE\} */> ⇒ ...}
  \]

  \[
  \text{<html xmlns = “http://www.w3.org/1999/xhtml”/>}
  \]
Examples of Goals

- Composition of independent language extensions
  ```
  grammar xml extends {Expression, Symbols}
  ```
- Expansion into other language extensions
  ```
  ... ⇒<]{<body />}>}
  ```
- Mutually recursive definition of language extensions
Outline

1. Growing a Language
   - Fortress Introduction
   - Growing a Language
   - XML Example Introduction

2. Objectives and Example
   - Syntactic Abstraction Goals
   - XML as an Example

3. Syntax Normalization
   - Parsing and Transformation
Syntax Normalization

- Parsing stage - transforms a source program (in text) into a *parsed program* (in node expression)

- Transformation stage - transforms the parsed program into a program in executable core Fortress AST
Parsing and Transformation

Syntax Normalization

- Parsing stage - transforms a source program (in text) into a *parsed program* (in node expression)
  - First step
    parses the macro definition into an intermediate form to generate a parser that recognizes the new syntax
  - Second step
    uses the generated parser to parse a source program using the new syntax
- Transformation stage - transforms the parsed program into a program in executable core Fortress AST
Node Expressions - Intermediate Representation

\[
\text{NodeExpr} ::= \text{PatternVar} \\
\quad | \text{Transformer} (\text{NodeExpr}) \\
\quad | \text{NodeConstructor}(\text{NodeExpr}) \\
\quad | \text{Ellipses} (\text{NodeExpr}, \text{NodeExpr}) \\
\quad | \text{case PatternVar of} \\
\quad | \quad \text{Empty} \Rightarrow \text{NodeExpr} \\
\quad | \quad \text{Cons}(\text{PatternVar}, \text{PatternVar}) \Rightarrow \text{NodeExpr} \\
\quad | \text{end}
\]
Transformation: Evaluation of Node Expressions

- Pattern variables are substituted by the corresponding values
- Transformers are replaced with their bodies, substituting pattern variables along the way
- Core Fortress AST nodes transform their arguments
- Case expression match its input to constructors `Empty` and `Cons`, and invokes the corresponding transformer
Summary

- Fortress is a growable language
- Syntactic abstraction supports language growth
- Implementation is available as part of the open-source Fortress reference implementation at:
  http://projectfortress.sun.com
Evaluation of Node Expressions (1)

[Pattern Variable]

\[
\Gamma(x) = v \\
\therefore \Gamma, \Gamma \vdash x \rightarrow \Gamma, \Gamma \vdash v
\]

[Node Constructor]

\[
\Upsilon, \Gamma \vdash \overline{n} \rightarrow \Upsilon, \Gamma \vdash \overline{n}' \\
\therefore \Upsilon, \Gamma \vdash c(\overline{n}) \rightarrow \Upsilon, \Gamma \vdash c(\overline{n}')
\]
Evaluation of Node Expressions (2)

[Macro Invocation Arguments]

\[
\begin{align*}
\forall, \Gamma \vdash \bar{n} & \rightarrow \forall, \Gamma \vdash \bar{n}' \\
\forall, \Gamma \vdash t(\bar{n}) & \rightarrow \forall, \Gamma \vdash t(\bar{n}')
\end{align*}
\]

[Macro Invocation]

\[
\begin{align*}
\forall (t) &= t \bar{x}.n \\
\forall, \Gamma \vdash t(\bar{v}) & \rightarrow \forall, \Gamma [\bar{x} \mapsto \bar{v}] \vdash n
\end{align*}
\]
[Case Empty]

\[
\begin{align*}
\forall \alpha, \beta \vdash x &\rightarrow \alpha, \beta \vdash \text{Empty} \\
\forall \alpha, \beta \vdash \text{case } x \text{ of} &\rightarrow \alpha, \beta \vdash n_1 \\
\text{Empty} &\Rightarrow n_1 \\
\text{Cons}(x_1, x_2) &\Rightarrow n_2 \\
\text{end} &
\end{align*}
\]

[Case Cons]

\[
\begin{align*}
\forall \alpha, \beta \vdash x &\rightarrow \alpha, \beta \vdash \text{Cons}(v_1, v_2) \\
\forall \alpha, \beta \vdash \text{case } x \text{ of} &\rightarrow \alpha, \beta [x_1 \leftarrow v_1][x_2 \leftarrow v_2] \vdash n_2 \\
\text{Empty} &\Rightarrow n_1 \\
\text{Cons}(x_1, x_2) &\Rightarrow n_2 \\
\text{end} &
\end{align*}
\]
Evaluation of Node Expressions (4)

[Ellipses First]
\[
|\overline{v}'| + 1 = i \leq size(n) \quad x_j \in PV(n) \quad |\Gamma(x_j)| > 1
\]
\[
\Gamma' = \Gamma \left[ x_j \mapsto \Gamma(x_j).nth(i) \right] \quad \Upsilon, \Gamma' \vdash n'' \rightarrow \Upsilon, \Gamma' \vdash n'''
\]
\[
\Upsilon, \Gamma \vdash \text{Ellipses}(n, \overline{v}'n'') \rightarrow \Upsilon, \Gamma \vdash \text{Ellipses}(n, \overline{v}'n''')
\]

[Ellipses Middle]
\[
|\overline{v}'| + 1 = i - 1 < size(n) \quad x_j \in PV(n) \quad |\Gamma(x_j)| > 1
\]
\[
\Gamma' = \Gamma \left[ x_j \mapsto \Gamma(x_j).nth(i) \right] \quad \Upsilon, \Gamma' \vdash n \rightarrow \Upsilon, \Gamma' \vdash n'''
\]
\[
\Upsilon, \Gamma \vdash \text{Ellipses}(n, \overline{v}'v'') \rightarrow \Upsilon, \Gamma \vdash \text{Ellipses}(n, \overline{v}'v''n''')
\]

[Ellipses Last]
\[
|\overline{v}'| + 1 = size(n)
\]
\[
\Upsilon, \Gamma \vdash \text{Ellipses}(n, \overline{v}'v'') \rightarrow \Upsilon, \Gamma \vdash \overline{v}'v''