Colored Nested Words

Rajeev Alur and Dana Fisman

LATA'16
Prague, Czech Republic
Nested Words

• A model of linear and hierarchical structure in data.

• Examples of such data
  • Executions of procedural programs
  • Marked-up languages such as XML
  • Annotated linguistic data
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: P(n) {
    p1: try {
        p2: x = 2*n
        p3: x = Q(x)
    }
    p4: }
    p5: catch (int) {}
    p6: return x }

q0: Q(n) {
    q1: y = n / 2
    q2: y = R(y)
    q3: y = y + 1
    q4: return y }

r0: R(n) {
    r1: z = n - 1
    r2: if (z < 0)
    r3: throw 0
    r4: return z }
Example: Execution of a Structured Program as a Nested Word

3 procedures

```
p0: $P(n)$ {
p1: try {
p2: x = 2*n
p3: x = Q(x)
p4: }
p5: catch (int) {}np6: return x }
q0: $Q(n)$ {
q1: y = n / 2
q2: y = R(y)
q3: y = y+1
q4: return y }
r0: $R(n)$ {
r1: z = n-1
r2: if (z<0)
r3: throw 0
r4: return z }
```
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[
\begin{align*}
p0: & \quad P(n) \{ \\
p1: & \quad \text{try} \{ \\
p2: & \quad x = 2*n \\
p3: & \quad x = Q(x) \\
p4: & \quad \} \\
p5: & \quad \text{catch (int)} \{ \\
p6: & \quad \text{return } x \} \\
q0: & \quad Q(n) \{ \\
q1: & \quad y = n / 2 \\
q2: & \quad y = R(y) \\
q3: & \quad y = y+1 \\
q4: & \quad \text{return } y \} \\
\text{r0: } & \quad R(n) \{ \\
r1: & \quad z = n-1 \\
r2: & \quad \text{if } (z<0) \\
r3: & \quad \text{throw } 0 \\
r4: & \quad \text{return } z \} 
\end{align*}
\]
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: \( P(n) \) {
  try {
    p1: \( x = 2^n \)
    p2: \( x = Q(x) \)
    p3: \}
  p4: catch (int) {} 
  p5: return x 
}

q0: \( Q(n) \) {
  q1: \( y = n / 2 \)
  q2: \( y = R(y) \)
  q3: \( y = y+1 \)
  q4: return y 
}

r0: \( R(n) \) {
  r1: \( z = n-1 \)
  r2: if (z<0) 
  r3: throw 0 
  r4: return z 
}
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: \( P(n) \) {
  p1: \( \text{try} \) {
    p2: \( x = 2*n \)
    p3: \( x = Q(x) \)
    p4: \}
  p5: \( \text{catch (int)} \) {
  p6: \( \text{return } x \) }
} q0: \( Q(n) \) {
  q1: \( y = n / 2 \)
  q2: \( y = R(y) \)
  q3: \( y = y + 1 \)
  q4: \( \text{return } y \) }
} r0: \( R(n) \) {
  r1: \( z = n - 1 \)
  r2: \( \text{if } (z < 0) \)
  r3: \( \text{throw } 0 \)
  r4: \( \text{return } z \) }
}
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: \( P(n) \) 
\begin{align*}
p0: & \quad P(n) \{ \\
p1: & \quad \text{try} \{ \\
p2: & \quad x = 2*n \\
p3: & \quad x = Q(x) \\
p4: & \quad \} \\
p5: & \quad \text{catch (int)} \{ \} \\
p6: & \quad \text{return } x \} \\
\end{align*}

q0: \( Q(n) \) 
\begin{align*}
q0: & \quad Q(n) \{ \\
q1: & \quad y = n / 2 \\
q2: & \quad y = R(y) \\
q3: & \quad y = y + 1 \\
q4: & \quad \text{return } y \} \\
\end{align*}

r0: \( R(n) \) 
\begin{align*}
r0: & \quad R(n) \{ \\
r1: & \quad z = n - 1 \\
r2: & \quad \text{if } (z < 0) \\
r3: & \quad \text{throw } 0 \\
r4: & \quad \text{return } z \} \\
\end{align*}
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[
\begin{align*}
\text{p0: } & P(n) \{ \\
\text{p1: } & \text{try } \{ \\
\text{p2: } & x = 2 \times n \\
\text{p3: } & x = Q(x) \\
\text{p4: } & \} \\
\text{p5: } & \text{catch (int) } \{ \} \\
\text{p6: } & \text{return } x \} \\
\text{q0: } & Q(n) \{ \\
\text{q1: } & y = n / 2 \\
\text{q2: } & y = R(y) \\
\text{q3: } & y = y + 1 \\
\text{q4: } & \text{return } y \} \\
\text{r0: } & R(n) \{ \\
\text{r1: } & z = n - 1 \\
\text{r2: } & \text{if } (z < 0) \\
\text{r3: } & \text{throw } 0 \\
\text{r4: } & \text{return } z \} 
\end{align*}
\]
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[
\begin{align*}
p_0: & \quad P(n) \{ \text{ try } \{ \\
p_1: & \quad x = 2 \times n \\
p_2: & \quad x = Q(x) \\
p_3: & \quad \} \\
p_4: & \quad \text{ catch (int) } \{ \\
p_5: & \quad \text{ return } x \} \\
q_0: & \quad Q(n) \{ \\
q_1: & \quad y = n / 2 \\
q_2: & \quad y = R(y) \\
q_3: & \quad y = y + 1 \\
q_4: & \quad \text{return } y \} \\
r_0: & \quad R(n) \{ \\
r_1: & \quad z = n - 1 \\
r_2: & \quad \text{ if } (z < 0) \\
r_3: & \quad \text{ throw } 0 \\
r_4: & \quad \text{return } z \} 
\end{align*}
\]
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[ p_0: \begin{array}{l} \text{P}(n) \{ \\
\text{try} \{ \\
\text{p}1: \quad x = 2 \times n \\
\text{p}3: \quad x = \text{Q}(x) \\
\text{p}4: \} \\
\text{p}5: \quad \text{catch} \ (\text{int}) \{\} \\
\text{p}6: \quad \text{return} \ x \} \end{array} \]

\[ q_0: \begin{array}{l} \text{Q}(n) \{ \\
\text{q}1: \quad y = n / 2 \\
\text{q}2: \quad y = \text{R}(y) \\
\text{q}3: \quad y = y + 1 \\
\text{q}4: \quad \text{return} \ y \} \end{array} \]

\[ r_0: \begin{array}{l} \text{R}(n) \{ \\
\text{r}1: \quad z = n - 1 \\
\text{r}2: \quad \text{if} \ (z < 0) \\
\text{r}3: \quad \text{throw} \ 0 \\
\text{r}4: \quad \text{return} \ z \} \end{array} \]
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[\begin{align*}
p_0 & : \text{P(n) \{ } & q_0 & : \text{Q(n) \{ } & r_0 & : \text{R(n) \{ } \\
p_1 & : \text{try \{ } & q_1 & : \text{y = n / 2 } & r_1 & : \text{z = n-1 } \\
p_2 & : \text{x = 2*n } & q_2 & : \text{y = R(y) } & r_2 & : \text{if (z<0) } \\
p_3 & : \text{x = Q(x) } & q_3 & : \text{y = y+1 } & r_3 & : \text{throw 0 } \\
p_4 & : \text{\} } & q_4 & : \text{return y } & r_4 & : \text{return z } \\
p_5 & : \text{catch (int) \{ } & & & & \\
p_6 & : \text{return x } & & & & \\
\end{align*}\]
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: P(n) {
    try {
        p1: x = 2*n
        p2: x = Q(x)
        p3: }
    p4: }
    p5: catch (int) {}
    p6: return x }

q0: Q(n) {
    q1: y = n / 2
    q2: y = R(y)
    q3: y = y+1
    q4: return y }

r0: R(n) {
    r1: z = n-1
    r2: if (z<0)
    r3: throw 0
    r4: return z }
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: P(n) { 
  p1: try {
    p2: x = 2*n
    p3: x = Q(x)
  }
  p4: }
  p5: catch (int) {} 
  p6: return x }

q0: Q(n) { 
  q1: y = n / 2 
  q2: y = R(y) 
  q3: y = y+1 
  q4: return y }

r0: R(n) { 
  r1: z = n-1 
  r2: if (z<0) 
  r3: throw 0 
  r4: return z }
Example: Execution of a Structured Program as a Nested Word

3 procedures

\[ p0: \text{P(n)} \{ \]
\[ p1: \text{try} \{ \]
\[ p2: \quad x = 2*n \]
\[ p3: \quad x = Q(x) \]
\[ p4: \} \]
\[ p5: \quad \text{catch (int)} \{ \}
\[ p6: \quad \text{return} \ x \} \]

\[ q0: \underline{\text{Q(n)}} \{ \]
\[ q1: \quad y = n / 2 \]
\[ q2: \quad y = R(y) \]
\[ q3: \quad y = y+1 \]
\[ q4: \quad \text{return} \ y \} \]

\[ r0: \underline{\text{R(n)}} \{ \]
\[ r1: \quad z = n-1 \]
\[ r2: \quad \text{if} \ (z<0) \]
\[ r3: \quad \text{throw} \ 0 \]
\[ r4: \quad \text{return} \ z \} \]
Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: P(n) {
   p1: try {
   p2: x = 2*n
   p3: x = Q(x)
   p4: }
   p5: catch (int) {}
   p6: return x }

q0: Q(n) {
   q1: y = n / 2
   q2: y = R(y)
   q3: y = y+1
   q4: return y }

r0: R(n) {
   r1: z = n-1
   r2: if (z<0)
   r3: throw 0
   r4: return z }

Example: Execution of a Structured Program as a Nested Word

3 procedures

p0: \( P(n) \) {
  try {
    p1: \( y = n / 2 \)
    p2: \( z = n - 1 \)
    p3: \( x = Q(n) \)
    p4: \( y = R(y) \)
    p5: \( y = y + 1 \)
    p6: return x }
}

q0: \( Q(n) \) {
  q1: \( y = n / 2 \)
  q2: \( z = n - 1 \)
  q3: \( y = y + 1 \)
  q4: return y }

r0: \( R(n) \) {
  r1: \( y = n / 2 \)
  r2: if \( z < 0 \)
  r3: throw 0
  r4: return z }

Nested Words Automata (NWA)

- Special kinds of pushdown automata whose push/pop actions are directed by the hierarchical structure in the input nested word.
  - On every open-tag, the automaton pushes
  - On every close-tag, the automaton pops
  - On data with no hierarchical connections, no stack operation is made.
- A language of nested words is said to be regular if it can be recognized by a NWA.
Properties of Regular Languages of Nested Words Automata

• Deterministic nested word automata are as expressive as their non-deterministic counterparts

• The class is closed under:
  - union
  - intersection
  - complementation
  - concatenation
  - Kleene-*
  - prefixes
  - reversal
  - homomorphism

• The following problems are decidable:
  - emptiness
  - membership
  - inclusion
  - equivalence
Weakness of Nested Words

- But what if an exceptions is thrown?

```
p0: P(n) {  
p1: try {  
p2: x = 2*n  
p3: x = Q(x)  
p4: }  
p5: catch (int) {}  
p6: return x }

q0: Q(n) {  
q1: y = n / 2  
q2: y = R(y)  
q3: y = y+1  
q4: return y }

r0: R(n) {  
r1: z = n-1  
r2: if (z<0)  
r3: throw 0  
r4: return z }
```
Weakness of Nested Words

• But what if an exceptions is thrown?

```plaintext
p0: P(n) {  
  p1: try {  
  p2: x = 2*n  
  p3: x = Q(x)  
  p4: }  
  p5: catch (int) {}  
  p6: return x }

q0: Q(n) {  
  q1: y = n / 2  
  q2: y = R(y)  
  q3: y = y+1  
  q4: return y }

r0: R(n) {  
  r1: z = n-1  
  r2: if (z<0)  
  r3: throw 0  
  r4: return z }
```

```
\[P(n) \{ \text{try}\{ \text{x = 2*n \} \text{catch (int) } \{ \text{return x}\}}\}
\[Q(n) \{ \text{y = n / 2 \} \text{y = R(y) \} \text{y = y+1 \} \text{return y}\}}\}
\[R(n) \{ \text{z = n-1 \} \text{if (z<0) \{ throw 0 \}} \text{return z }\}}\]
```
Weakness of Nested Words

• But what if an exceptions is thrown?

```
p0: P(n) {
  p1: try {
  p2:   x = 2*n
  p3:   x = Q(x)
  p4: }
  p5: catch (int) {}
  p6: return x }

q0: Q(n) {
  q1: y = n / 2
  q2: y = R(y)
  q3: y = y+1
  q4: return y }

r0: R(n) {
  r1: z = n-1
  r2: if (z<0)
  r3: throw 0
  r4: return z }
```
Weakness of Nested Words

• But what if an exceptions is thrown?

```plaintext
p0: \textcolor{red}{P(n)} \{ 
\textcolor{blue}{p1: \text{try} \{ 
\textcolor{green}{p2: \text{x = 2*n}} 
\textcolor{orange}{p3: \text{x = Q(x)}} 
\textcolor{green}{p4: \}}}
\textcolor{green}{p5: \text{catch (int) \{}}}
\textcolor{green}{p6: \text{return x \}}
q0: \textcolor{red}{Q(n)} \{ 
q1: \textcolor{blue}{y = n / 2}
q2: \textcolor{blue}{y = R(y)}
q3: \textcolor{blue}{y = y+1}
q4: \textcolor{blue}{\text{return y \}}
r0: \textcolor{red}{R(n)} \{ 
r1: \textcolor{blue}{z = n-1}
r2: \textcolor{blue}{\text{if (z<0)}}
r3: \textcolor{green}{\text{throw 0}}
r4: \textcolor{blue}{\text{return z \}}
```

![Diagram showing the flow of the program with try-catch block and exception handling.](image)
Weakness of Nested Words

• But what if an exceptions is thrown?

```
p0: P(n) {
p1: try {
p2: x = 2*n
p3: x = Q(x)
p4: }
p5: catch (int) {}
p6: return x }
```

```
q0: Q(n) {
q1: y = n / 2
q2: y = R(y)
q3: y = y+1
q4: return y }
```

```
r0: R(n) {
r1: z = n-1
r2: if (z<0)
r3: throw 0
r4: return z }
```

The desired sequence is shown in the left diagram, while what we may get is shown in the right diagram.
def isPrime(n):
    if n<2: return False
    elif n==2: return True
    else:
        i = 2
        while i < sqrt(n)+1:
            if n%i==0:
                return False
            else:
                i = i+1
        return True

Closes scopes 5,4,3
Colored Nested Words - Encoding

- **Explicitly**, using a graph with several types of edges (linear, mono-chromatic and bi-chromatic) and defining the constraints they should satisfy.

- **Implicitly**, using colored/types parenthesis.

\[
(p_0 (p_1 p_1 p_3 (q_0 q_1 q_2 (r_0 r_1 r_2 r_3) p_5 p_6))
\]

\[
[p_0 (p_1 p_1 p_3 [q_0 q_1 q_2 [r_0 r_1 r_2 r_3) p_5 p_6]]
\]
Regularity

• How can we define *regularity* of a *language* of colored nested words?

• Intuitively, we’d like the *language* obtained by inserting the missing closing tags to be a *well-colored regular language* of nested words.

\[
(p_0 (p_1 p_1 p_3 (q_0 q_1 q_2 (r_0 r_1 r_2 r_3) p_5 p_6))
\]

\[
(p_0 (p_1 p_1 p_3 (q_0 q_1 q_2 (r_0 r_1 r_2 r_3) p_5 p_6))
\]
Regularity

• Suppose we have such a mapping from CNW to NW

\[ f(w) = w' \]

such that \( w' \) has all the missing closing tags

• Definition:

Let \( L \) be a language of \( \text{CNW} \).

Then \( L \) is regular if

\[ \{ f(w) \mid w \in L \} \] is a regular lang. of \( \text{NW} \)
How can we process regular CNW?

- But how can we implement this transducer?
- It is not regular, and not regular NW, because it needs to pop more than one letter when reading a “rescuing closing tag”!
Colored Nested Words Automata

• The pushdown automaton marks the pushed letters with the respective color.

• A read of an x-colored close-tag pops from the stack all symbols of y-colored symbols up to the first x-colored symbol (as well as this letter itself).
Colored Nested Words Automata

- The pushdown automaton marks the pushed letters with the respective color.
- A read of an x-colored close-tag pops from the stack all symbols of y-colored symbols up to the first x-colored symbol (as well as this letter itself).
Blind vs. Sighted CNA

• Can the automaton view all the letters in the stack above the matching x-colored letter?
• Or can it just see that x-colored letter?
• We call the first version Sighted CNA

R(n)
Q(n)
try
P(n)
Blind vs. Sighted CNA

• Can the automaton view all the letters in the stack above the matching x-colored letter?
• Or can it just see that x-colored letter?
• We call the first version Sighted CNA
• And the second version Blind CNA
• Theorem 1

Sighted CNA and Blind CNA have the same expressive power
• Theorem 2

Sighted CNA and Blind CNA accept all the regular languages of colored nested words
Other Examples – broken html

```html
<div id="s">
  <ol>
    <li>ML</li>
    <li>Haskell</li>
    <li>Scheme</li>
  </ol>
</div>

<ol>
  <li>C</li>
  <li>C++</li>
  <li>Perl</li>
</ol>
```
CNA – html example

Accepts well formed html

Accepts also broken html

Push trans
Pop trans
Internal
ε-trans
CNA - Examples

Accepts word where #a’s between () is even

(')' allowed only in even

(')' always takes us to even

\[
\begin{align*}
\text{even} & \rightarrow \text{\(\hat{a}\)} \rightarrow \text{odd} \\
\text{[ \(\downarrow e\) } & \rightarrow \text{ even } \\
\text{[ \(\downarrow o\) } & \rightarrow \text{ odd }
\end{align*}
\]
CNA - Examples

Accepts word where #a’s between () is even and #a’s between [] is odd

Also need to record the type of open tag ‘(‘ / ‘[‘

‘)‘ allowed only in even

‘]‘ allowed only in odd
CNA - Examples

Accepts word where #a’s between () is even and #a’s between [] is odd

a ( may be unmatched if encapsulated by []

We need different colors for ‘()’ and ‘[]’

We can pop unmatched ‘(’
Properties of Regular Languages of Colored Nested Words Automata

• Deterministic nested word automata are as expressive as their non-deterministic counterparts.

• The class is closed under:
  • union
  • intersection
  • complementation
  • Kleene- *
  • prefixes
  • reversal
  • concatenation
  • homomorphism

• The following problems are decidable:
  • emptiness
  • membership
  • inclusion
  • equivalence

not closed under reversal
There exists a grammar characterization for NWs

- It classifies variables to two sets
- One that disallows pending calls
- One that disallows pending returns

There exists a grammar characterization for CNWs

- It classifies variables $|C|+2$ sets
- One per each color
- One that disallows pending calls
- One that disallows pending returns
Thank you for your attention!

Comments / Questions?