Introduction to Ranked Tree Automata

Martin Braun

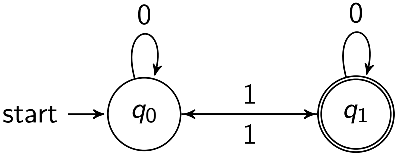
University of Bayreuth

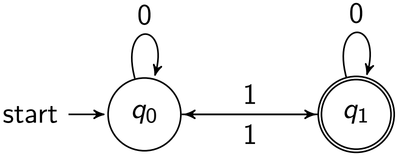
30.04.2014

Supervisor: Prof. Dr. Wim Martens

A first example

A look back to NFAs/DFAs

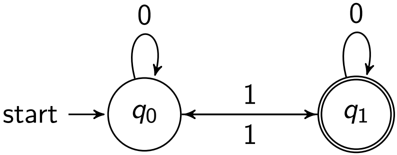




This DFA accepts all strings with an odd number of 1's

A first example

A look back to NFAs/DFAs



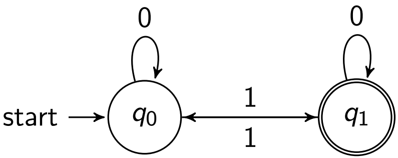
This DFA accepts all strings with an odd number of 1's

NFAs/DFAs consist of:

- a finite set of states Q
- a finite set of input symbols Σ
- a transitional relation $\Delta : Q \times \Sigma \to Q$
- an initial state $q_0 \in Q$
- a set of final states $Q_f \subseteq Q$

A first example

A look back to NFAs/DFAs



This DFA accepts all strings with an odd number of 1's

NFAs/DFAs consist of:

- a finite set of states Q
- a finite set of input symbols Σ
- a transitional relation $\Delta : Q \times \Sigma \to Q$
- an initial state $q_0 \in Q$
- a set of final states $Q_f \subseteq Q$

They are used to recognize strings.

A first example

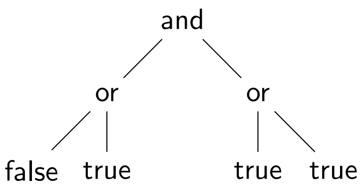
Strings are nice, but...

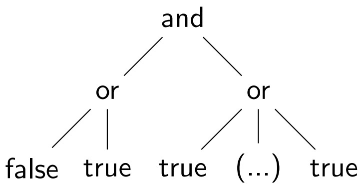
...what about languages that have an inherent structure?

A first example

Strings are nice, but...

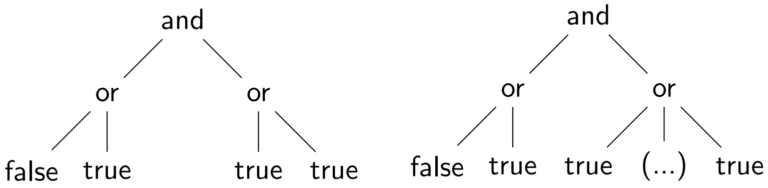
...what about languages that have an inherent structure?



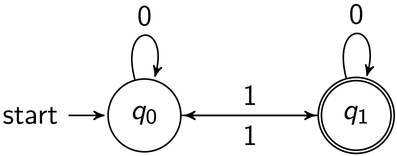


A first example
Strings are nice, but...

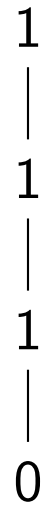
...what about languages that have an inherent structure?



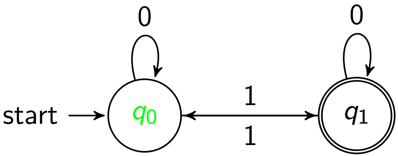
What if we want to recognize a tree-language that consists of all true boolean statements?



Now consider a run on this NFA with the input:
0 1 1 1



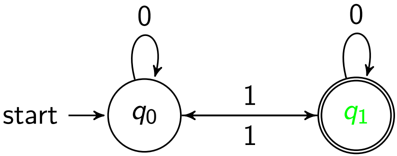
A first example



Input:
0 1 1 1

$$
\begin{array}{c}
1 \\
| \\
1 \\
| \\
1 \\
| \\
0
\end{array}
\qquad 0(q_0) \rightarrow q_0
$$

A first example



Input:

0 1 1 1

1

|

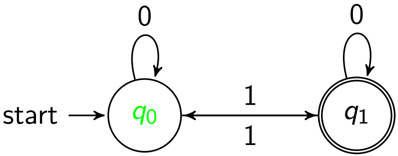
1

|

1 $\quad 1(q_0) \to q_1$

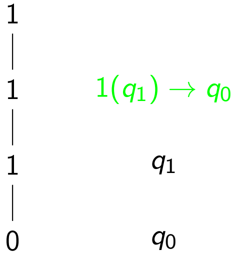
|

0 $\quad q_0$

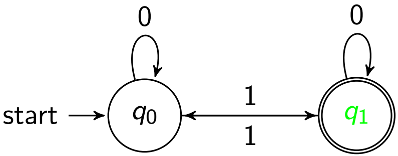


Input:

0 1 1 1



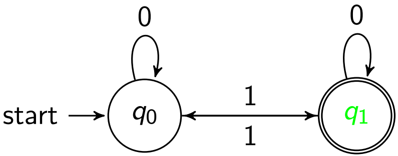
A first example



Input:
0 1 1 1

1	$1(q_0) \to q_1$
|	
1	q_0
|	
1	q_1
|	
0	q_0

A first example



Input:

0 1 1 1. Finished.

$State(nfa) = q_1 \implies Input \in L(nfa)$

1	q_1
1	q_0
1	q_1
0	q_0

From NFA to NFTA

BinaryCounter (1)

Example

We expand our NFA to count the 1's and 0's in "parallel". Normal strings aren't sufficient. A restructuration of the input is needed.

From NFA to NFTA

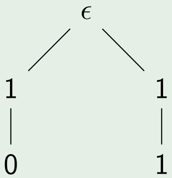
BinaryCounter (1)

Example

We expand our NFA to count the 1's and 0's in "parallel". Normal strings aren't sufficient. A restructuration of the input is needed.

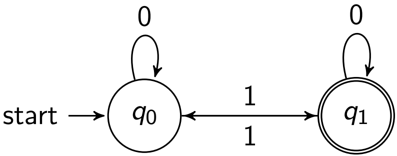
Example

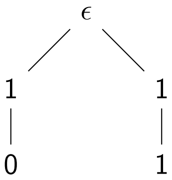
For 0 1 1 1 :



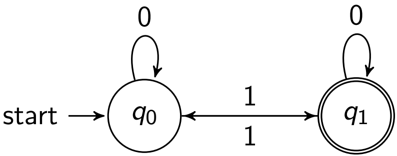
From NFA to NFTA

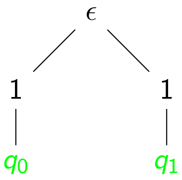
BinaryCounter (2)





BinaryCounter (3)



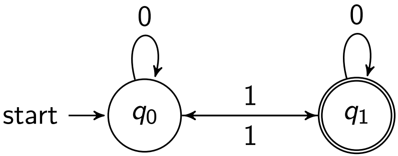


$0(q_0) \rightarrow q_0$

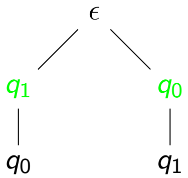
$1(q_0) \rightarrow q_1$

From NFA to NFTA

BinaryCounter (4)



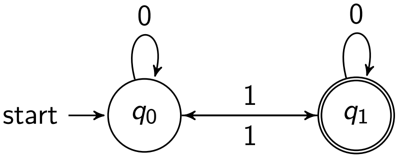
$1(q_0) \to q_1$

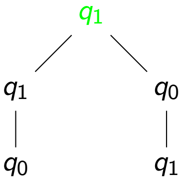


$1(q_1) \to q_0$

From NFA to NFTA

BinaryCounter (5)

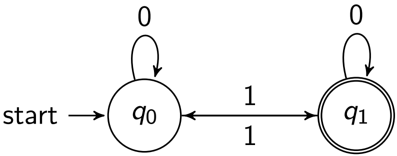


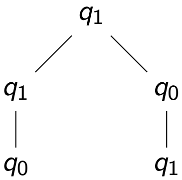


$\epsilon(q_0, q_1) \rightarrow q_1$

From NFA to NFTA

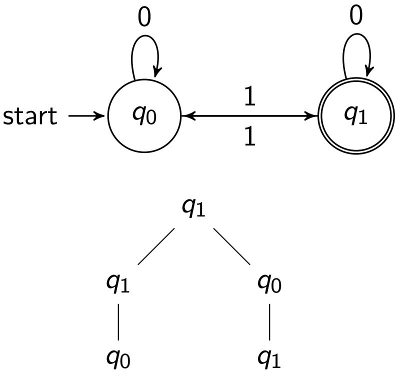
BinaryCounter (6)





From NFA to NFTA

BinaryCounter (6)



Accepted.

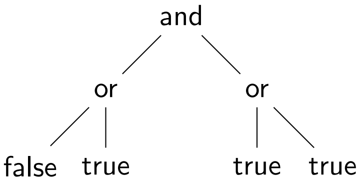
From NFA to NFTA

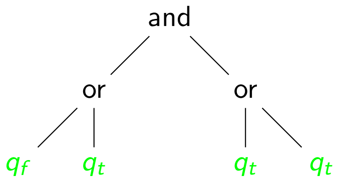
NFTAs

NFTAs behave the same as NFAs except for Δ. They can have more than one state as a arguments to their terms, i.e.: $term(q_m, q_n) \rightarrow q_x$

From NFA to NFTA

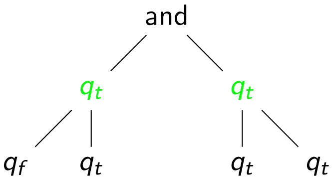
BooleanEvaluator (1)





- $false \rightarrow q_f$
- $true \rightarrow q_t$

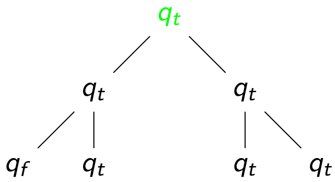
From NFA to NFTA
BooleanEvaluator (3)



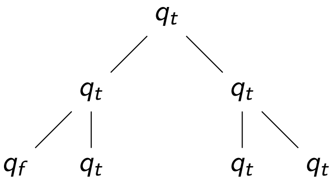
- $or(q_f, q_t) \to q_t$
- $or(q_t, q_t) \to q_t$

From NFA to NFTA

BooleanEvaluator (4)

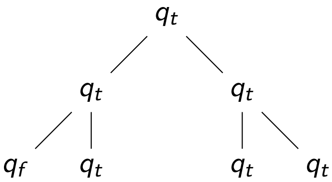


- $and(q_t, q_t) \to q_t$


q_t
q_t q_t
q_f q_t q_t q_t

From NFA to NFTA

BooleanEvaluator (5)



Accepted.

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{or, and, not, true, false\}$

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{or, and, not, true, false\}$
- $Q = \{q_f, q_t\}$

From NFA to NFTA

The formal BooleanEvaluator

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{or, and, not, true, false\}$
- $Q = \{q_f, q_t\}$
- $Q_f = \{q_t\}$

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{or, and, not, true, false\}$
- $Q = \{q_f, q_t\}$
- $Q_f = \{q_t\}$
- $\Delta = \{false \rightarrow q_f, true \rightarrow q_t,$
  $and(q_t, q_t) \rightarrow q_t, and(q_t, q_f) \rightarrow q_f, ...,$
  $or(q_t, q_t) \rightarrow q_t, or(q_t, q_f) \rightarrow q_t, ...,$
  $not(q_f) \rightarrow q_t, not(q_t) \rightarrow q_f\}$

Example

Our boolean tree-NFTA is therefore defined as:

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{or, and, not, true, false\}$
- $Q = \{q_f, q_t\}$
- $Q_f = \{q_t\}$
- $\Delta = \{false \rightarrow q_f, true \rightarrow q_t,$
  $and(q_t, q_t) \rightarrow q_t, and(q_t, q_f) \rightarrow q_f, ...,$
  $or(q_t, q_t) \rightarrow q_t, or(q_t, q_f) \rightarrow q_t, ...,$
  $not(q_f) \rightarrow q_t, not(q_t) \rightarrow q_f\}$

This NFTA regonizes all boolean expressions that evaluate to *true*.

Some additions...

...to the definition (1)

Definition

A tree automaton with no two rules of the type

- $f(q_1, ..., q_n) \rightarrow q_x$
- $f(q_1, ..., q_n) \rightarrow q_y$

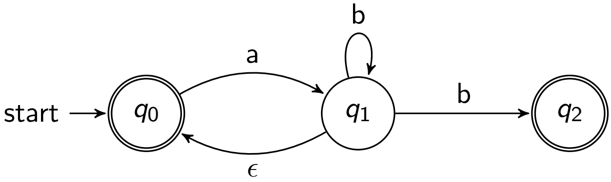
with $q_x \neq q_y$ and no rules of the type

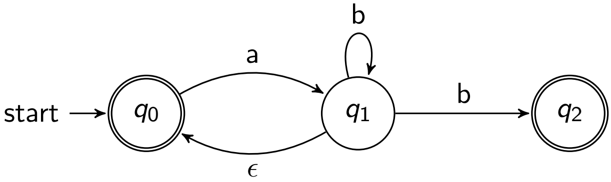
- $\epsilon(q_1, ..., q_n) \rightarrow q_x$

is called deterministic. (DFTA)

Determinization

What we already know (1)



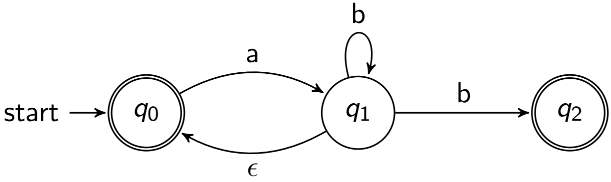


How it's done:

- Use $\epsilon - closure(q)$ instead of q for all $q \in Q$
- Start with the initial state
- Generate target states "on the fly" (=set of possible states in original Q)
- Repeat for all states including the newly generated ones

Determinization

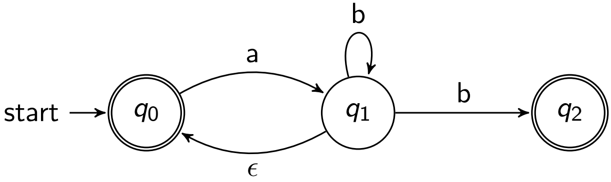
What we already know (2)



- Initial state: $\{q_0\}$

Determinization

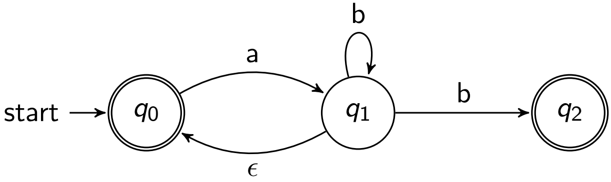
What we already know (2)



- Initial state: $\{q_0\}$
- $a(\{q_0\}) \to \{q_0, q_1\} \in Q'_f$

Determinization

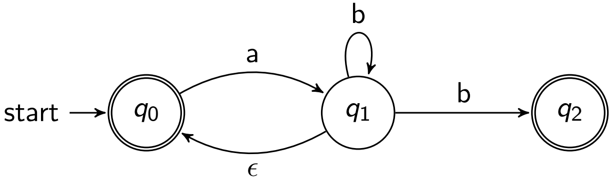
What we already know (2)



- Initial state: $\{q_0\}$
- $a(\{q_0\}) \rightarrow \{q_0, q_1\} \in Q'_f$
- $a(\{q_0, q_1\}) \rightarrow \{q_0, q_1\}$
  $b(\{q_0, q_1\}) \rightarrow \{q_0, q_1, q_2\} \in Q'_f$

Determinization

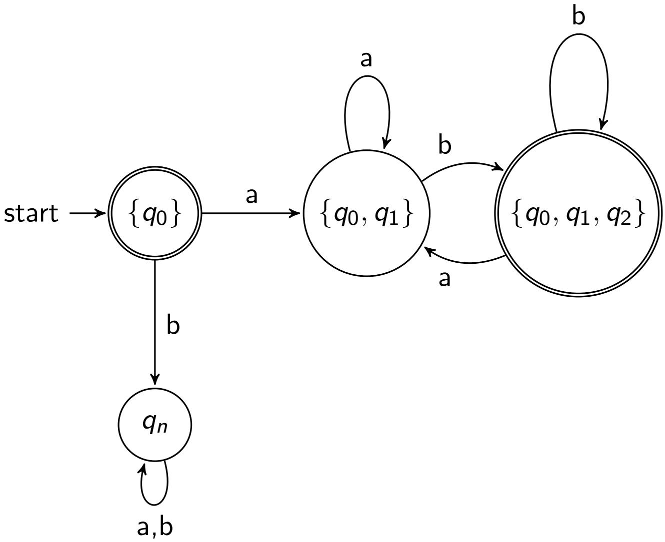
What we already know (2)



- Initial state: $\{q_0\}$
- $a(\{q_0\}) \to \{q_0, q_1\} \in Q'_f$
- $a(\{q_0, q_1\}) \to \{q_0, q_1\}$
  $b(\{q_0, q_1\}) \to \{q_0, q_1, q_2\} \in Q'_f$
- $a(\{q_0, q_1, q_2\}) \to \{q_0, q_1\}$
  $b(\{q_0, q_1, q_2\}) \to \{q_0, q_1, q_2\}$

Determinization

What we already know (3)



Determinization

Now for NFTAs (1)

Example

consider this NFTA that accepts unordered lists (HTML-style)

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{q_{ul}, q_{li1}, q_{li2}, q_{text}, q_{empty}\}$
- $Q_f = \{q_{ul}\}$
- $\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
  $li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
  $text \rightarrow q_{text}, empty \rightarrow q_{empty},$
  $\epsilon(q_{empty}) \rightarrow q_{text}\}$

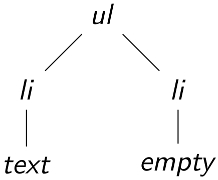
Determinization

Now for NFTAs (2)

```
<ul>
    <li>text</li>
    <li>empty</li>
</ul>
```

```
<ul>
    <li>text</li>
    <li>empty</li>
</ul>
```

Or as a tree input:



Determinization

Now for NFTAs (3)

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
$text \rightarrow q_{text}, empty \rightarrow q_{empty},$
$\epsilon(q_{empty}) \rightarrow q_{text}\}$

- Use $\epsilon - closure(q)$ instead of q for all $q \in Q$
- Start with the ground terms
- Generate target states "on the fly" (=set of possible states in original Q)
- Repeat for all states including the newly generated ones

Determinization

Now for NFTAs (4)

Example

$Q_f = \{q_{ul}\}$

$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$

$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$

$text \rightarrow q_{text}, empty \rightarrow q_{empty},$

$\epsilon(q_{empty}) \rightarrow q_{text}\}$

rules and states:

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
$text \rightarrow q_{text}, empty \rightarrow q_{empty},$
$\epsilon(q_{empty}) \rightarrow q_{text}\}$

rules and states:
$text \rightarrow \{q_{text}\}$

Determinization

Now for NFTAs (4)

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
$text \rightarrow q_{text}, empty \rightarrow q_{empty},$
$\epsilon(q_{empty}) \rightarrow q_{text}\}$

rules and states:
$text \rightarrow \{q_{text}\}$
$empty \rightarrow \{q_{text}, q_{empty}\}$

Determinization

Now for NFTAs (4)

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
$text \rightarrow q_{text}, empty \rightarrow q_{empty},$
$\epsilon(q_{empty}) \rightarrow q_{text}\}$

rules and states:
$text \rightarrow \{q_{text}\}$
$empty \rightarrow \{q_{text}, q_{empty}\}$
$li(\{q_{text}\}, \{q_{text}\}) \rightarrow \{q_{li1}, q_{li2}\}$

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \to q_{ul}, ul(q_{li2}, q_{li1}) \to q_{ul},$
$li(q_{text}, q_{text}) \to q_{li1}, li(q_{text}, q_{text}) \to q_{li2}$
$text \to q_{text}, empty \to q_{empty},$
$\epsilon(q_{empty}) \to q_{text}\}$

rules and states:
$text \to \{q_{text}\}$
$empty \to \{q_{text}, q_{empty}\}$
$li(\{q_{text}\}, \{q_{text}\}) \to \{q_{li1}, q_{li2}\}$
$li(\{q_{text}, q_{empty}\}, \{q_{text}, q_{empty}\}) \to \{q_{li1}, q_{li2}\}$

Determinization
Now for NFTAs (4)

Example

$Q_f = \{q_{ul}\}$
$\Delta = \{ul(q_{li1}, q_{li2}) \rightarrow q_{ul}, ul(q_{li2}, q_{li1}) \rightarrow q_{ul},$
$li(q_{text}, q_{text}) \rightarrow q_{li1}, li(q_{text}, q_{text}) \rightarrow q_{li2}$
$text \rightarrow q_{text}, empty \rightarrow q_{empty},$
$\epsilon(q_{empty}) \rightarrow q_{text}\}$

rules and states:
$text \rightarrow \{q_{text}\}$
$empty \rightarrow \{q_{text}, q_{empty}\}$
$li(\{q_{text}\}, \{q_{text}\}) \rightarrow \{q_{li1}, q_{li2}\}$
$li(\{q_{text}, q_{empty}\}, \{q_{text}, q_{empty}\}) \rightarrow \{q_{li1}, q_{li2}\}$
$ul(\{q_{li1}, q_{li2}\}, \{q_{li1}, q_{li2}\}) \rightarrow \{q_{ul}\}$

Determinization

Now for NFTAs (5)

Example

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{\{q_{ul}\}, \{q_{text}\}, \{q_{text}, q_{empty}\}, \{q_{li1}, q_{li2}\}\}$
- $Q_f = \{\{q_{ul}\}\}$
- $\Delta = \{text \rightarrow \{q_{text}\},$
  $empty \rightarrow \{q_{text}, q_{empty}\},$
  $li(\{q_{text}\}, \{q_{text}\}) \rightarrow \{q_{li1}, q_{li2}\},$
  $li(\{q_{text}, q_{empty}\}, \{q_{text}, q_{empty}\}) \rightarrow \{q_{li1}, q_{li2}\},$
  $ul(\{q_{li1}, q_{li2}\}, \{q_{li1}, q_{li2}\}) \rightarrow \{q_{ul}\}\}$

Determinization

Now for NFTAs (6)

Example

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{q_{ul}, q_{text}, q_{text2}, q_{li}\}$
- $Q_f = \{q_{ul}\}$
- $\Delta = \{text \rightarrow q_{text},$
  $empty \rightarrow q_{text2},$
  $li(q_{text}, q_{text}) \rightarrow q_{li},$
  $li(q_{text2}, q_{text2}) \rightarrow q_{li},$
  $ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

Minimization

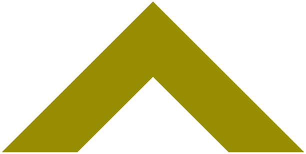
Context (Definition)

Definition

A Context C is a tree with a hole.

Definition

A Context C is a tree with a hole.

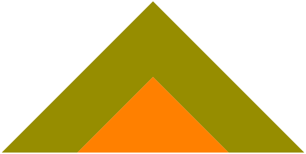


Definition

$C[t]$ is a Context application of t in the context of C.

Definition

$C[t]$ is a Context application of t in the context of C.



Definition

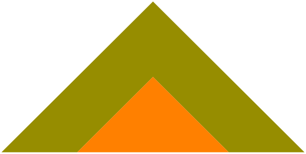
A congruence is an equivalence relation on trees closed under context:
$t_1 \equiv t_2 \Rightarrow \forall C : C[t_1] \equiv C[t_2]$

Definition

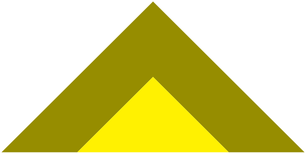
A congruence is an equivalence relation on trees closed under context:
$t_1 \equiv t_2 \Rightarrow \forall C : C[t_1] \equiv C[t_2]$

Definition

A congruence is an equivalence relation on trees closed under context:
$t_1 \equiv t_2 \Rightarrow \forall C : C[t_1] \equiv C[t_2]$



$\equiv$

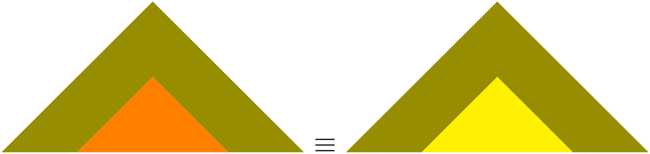


Minimization: Myhill-Nerode

Congruence (Definition)

Definition

A congruence is an equivalence relation on trees closed under context:
$t_1 \equiv t_2 \Rightarrow \forall C : C[t_1] \equiv C[t_2]$



Definition

For a tree language L we can define the congruence in L $\equiv_L$:
$t_1 \equiv_L t_2$ if $\forall$ Contexts C: $C[t_1] \in L \iff C[t_2] \in L$

Theorem

The following are equivalent:

- *L is a regular tree language*
- *L is the union of some congruence classes of finite index*
- *the relation $\equiv_L$ is a congruence of finite index*

Theorem

The following are equivalent:

- *L is a regular tree language*
- *L is the union of some congruence classes of finite index*
- *the relation $\equiv_L$ is a congruence of finite index*

$\Rightarrow$ $\mathit{sizeof}(A_{min})$ = number of equivalence classes in L

Example

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{q_{ul}, q_{text}, q_{text2}, q_{li}\}$
- $Q_f = \{q_{ul}\}$
- $\Delta = \{text \rightarrow q_{text},$
  $empty \rightarrow q_{text2},$
  $li(q_{text}, q_{text}) \rightarrow q_{li},$
  $li(q_{text2}, q_{text2}) \rightarrow q_{li},$
  $ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

Minimization
Example (1)

Example

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{q_{ul}, q_{text}, q_{text2}, q_{li}\}$
- $Q_f = \{q_{ul}\}$
- $\Delta = \{text \rightarrow q_{text},$
  $empty \rightarrow q_{text2},$
  $li(q_{text}, q_{text}) \rightarrow q_{li},$
  $li(q_{text2}, q_{text2}) \rightarrow q_{li},$
  $ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

Minimization

Example (2)

Example

$\Delta = \{text \to q_{text},$
$empty \to q_{text2},$
$li(q_{text}, q_{text}) \to q_{li},$
$li(q_{text2}, q_{text2}) \to q_{li},$
$ul(q_{li}, q_{li}) \to q_{ul}\}$

	q_{text}	q_{text2}	q_{li}	q_{ul}
q_{text}	-	-	-	-
q_{text2}		-	-	-
q_{li}			-	-
q_{ul}				-

Example

$\Delta = \{text \rightarrow q_{text},$

$empty \rightarrow q_{text2},$

$li(q_{text}, q_{text}) \rightarrow q_{li},$

$li(q_{text2}, q_{text2}) \rightarrow q_{li},$

$ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

	q_{text}	q_{text2}	q_{li}	q_{ul}
q_{text}	-	-	-	-
q_{text2}		-	-	-
q_{li}			-	-
q_{ul}	0	0	0	-

Minimization

Example (4)

Example

$\Delta = \{text \rightarrow q_{text},$
$empty \rightarrow q_{text2},$
$li(q_{text}, q_{text}) \rightarrow q_{li},$
$li(q_{text2}, q_{text2}) \rightarrow q_{li},$
$ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

	q_{text}	q_{text2}	q_{li}	q_{ul}
q_{text}	-	-	-	-
q_{text2}		-	-	-
q_{li}	1	1	-	-
q_{ul}	0	0	0	-

Minimization

Example (5)

Example

$\Delta = \{text \rightarrow q_{text},$
$empty \rightarrow q_{text2},$
$li(q_{text}, q_{text}) \rightarrow q_{li},$
$li(q_{text2}, q_{text2}) \rightarrow q_{li},$
$ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

	q_{text}	q_{text2}	q_{li}	q_{ul}
q_{text}	-	-	-	-
q_{text2}	(merge)	-	-	-
q_{li}	1	1	-	-
q_{ul}	0	0	0	-

Example

- $A = (Q, \Sigma, Q_f, \Delta)$
- $\Sigma = \{ul, li, text, empty\}$
- $Q = \{q_{ul}, q_{text_{new}}, q_{li}\}$
- $Q_f = \{q_{ul}\}$
- $\Delta = \{text \rightarrow q_{text_{new}},$
  $empty \rightarrow q_{text_{new}},$
  $li(q_{text_{new}}, q_{text_{new}}) \rightarrow q_{li},$
  $ul(q_{li}, q_{li}) \rightarrow q_{ul}\}$

Conclusion

- (ranked) NFTA/DTFAs behave similar to NFA/DFAs
- (ranked) NFTAs can be determinized
- (ranked) DFTAs can be minimized

Thank you for the attention!

Sources I

Hubert Comon, et al.
Tree Automata Techniques and Applications, 2007.

Prof. Dr. Wim Martens et al.
Automata and Logic on Trees
http://lrb.cs.uni-dortmund.de/~martens/data/esslli07/lecture01.pdf, http://lrb.cs.uni-dortmund.de/~martens/data/esslli07/lecture02.pdf,
2007

Michaela Regneri, Stefan Thater
Programmierkurs Python II SS 2011 (Determinization of NFAs)
http://www.coli.uni-saarland.de/courses/python2-11/folien/PythonII11-05.pdf,
2011

Sources II

Prof. Dr. J. Giesl et al.
Formale Systeme, Automaten, Prozesse SS 2010 (Minimization of DFAs)
http://verify.rwth-aachen.de/fosap10/uebungen/blaetter/Loesung5.pdf, 2010