Precedence Automata and Languages

Violetta Lonati

DSI - Università degli studi di Milano

Joint work with
Dino mandrioli and Matteo Pradella
DEI - Politecnico di Milano

CSR 2011

Outline

Outline

Historical background (1)

Floyd languages

- Operator precedence grammars, inspired by the structure of arithmetic expressions, and aimed at deterministic (hence efficient) parsing [Floyd 1963]
- Dyck languages and the fundamental theorem by Chomsky and Shützemberger [1963]
- Parenthesis grammars [McNaughton 1967]
- Tree automata [Thatcher 1967]
- Some early investigation on the algebraic properties of operator precedence grammars and closure properties of context-free languages compared with regular languages [Crespi Reghizzi, Mandrioli and Martin 1978]

Historical background (2)

After 40 years and more

- Visibly Pushdown Languages [Alur and Madhusudan 2004]
  - suitable to model structured mark-up languages such as XML and apply model checking techniques
  - closure properties of regular languages (boolean, concatenation, kleene $*$, ...)
- various related familes have been then defined and studied [Nowotka and Srba 2007, Caucal 2008]

FLs are back

FLs actually include VPLs
and enjoy the same closure properties

[Crespi Reghizzi and Mandrioli 2009]

Outline

Let's start with an example

We have an arithmetic expression

$$4 + 5 \times 6$$

we want to evaluate it respecting the precedence among operators

Let's start with an example

We have an arithmetic expression

$$4 + 5 \times 6$$

we want to evaluate it respecting the precedence among operators

$$\begin{array}{cccccc} 4 & + & 5 & \times & 6 & \\ & & \multicolumn{3}{c}{\overline{\qquad\qquad 30}} & \\ \multicolumn{6}{c}{\overline{\qquad\qquad\qquad\qquad 34}} \end{array}$$

Let's start with an example

We have an arithmetic expression

$$4 + 5 \times 6$$

we want to evaluate it respecting the precedence among operators

$$\begin{array}{cccccc} 4 & + & 5 & \times & 6 \\ & & \multicolumn{3}{c}{\overline{\qquad\quad 30}} \\ \multicolumn{6}{c}{\overline{\qquad\qquad\qquad\qquad 34}} \end{array}$$

It's like we add some parentheses

$$\big(4 + (5 \times 6)\big)$$

How does the parsing work?

Implicitly, we refer to a context-free grammar:

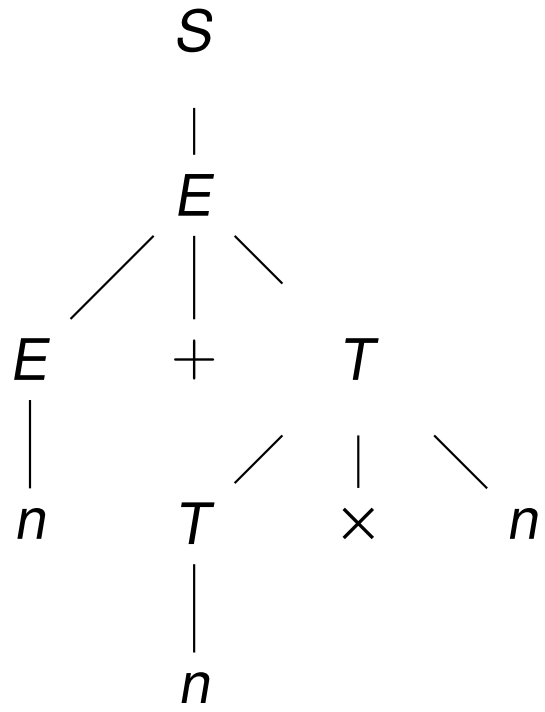
$$
\begin{aligned}
S &\rightarrow E \\
E &\rightarrow E + T \\
E &\rightarrow T \times n \\
E &\rightarrow n \\
T &\rightarrow T \times n \\
T &\rightarrow n
\end{aligned}
$$

How does the parsing work?

Implicitly, we refer to a context-free grammar:

$$
\begin{aligned}
S &\to E \\
E &\to E + T \\
E &\to T \times n \\
E &\to n \\
T &\to T \times n \\
T &\to n
\end{aligned}
$$

$n + n \times n$

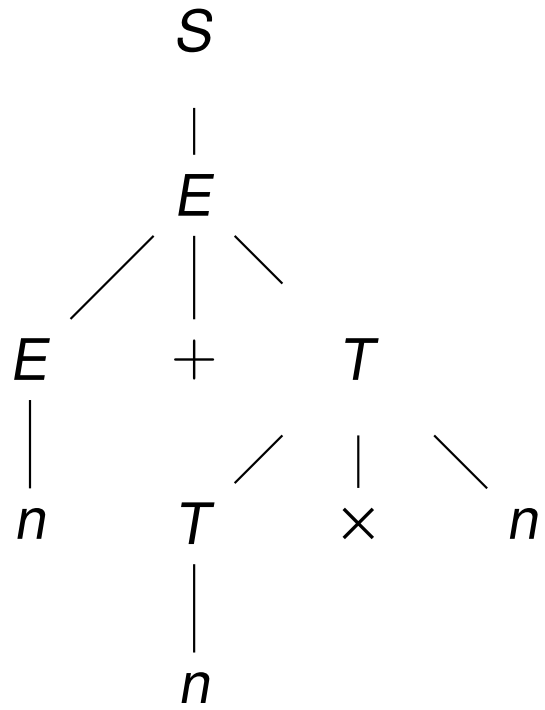


How does the parsing work?

Implicitly, we refer to a context-free grammar:

$$
\begin{aligned}
S &\to E \\
E &\to E + T \\
E &\to T \times n \\
E &\to n \\
T &\to T \times n \\
T &\to n
\end{aligned}
$$

$$n + n \times n$$



Suppose you have a method for inserting the proper parentheses into an expression . . . you get the **stencil** of the parsing tree

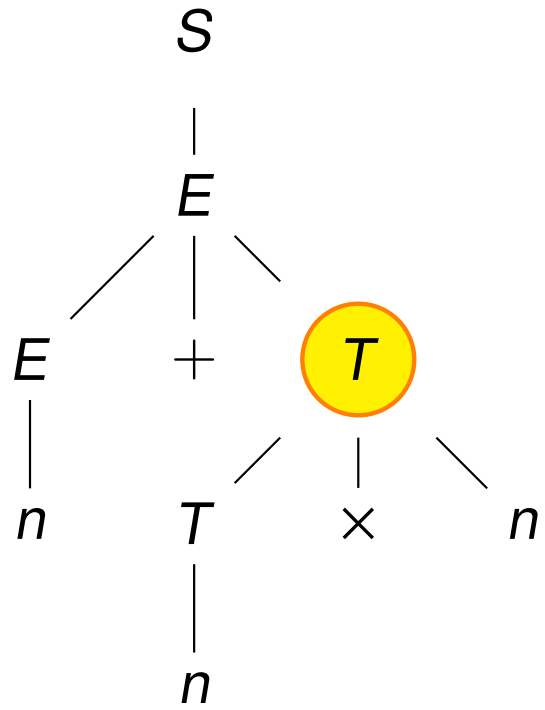
$$\Big(n + \ (n \times n)\ \Big)$$

How does the parsing work?

Implicitly, we refer to a context-free grammar:

$$
\begin{aligned}
S &\to E \\
E &\to E + T \\
E &\to T \times n \\
E &\to n \\
T &\to T \times n \\
T &\to n
\end{aligned}
$$

$$n + n \times n$$



Suppose you have a method for inserting the proper parentheses into an expression . . . you get the **stencil** of the parsing tree

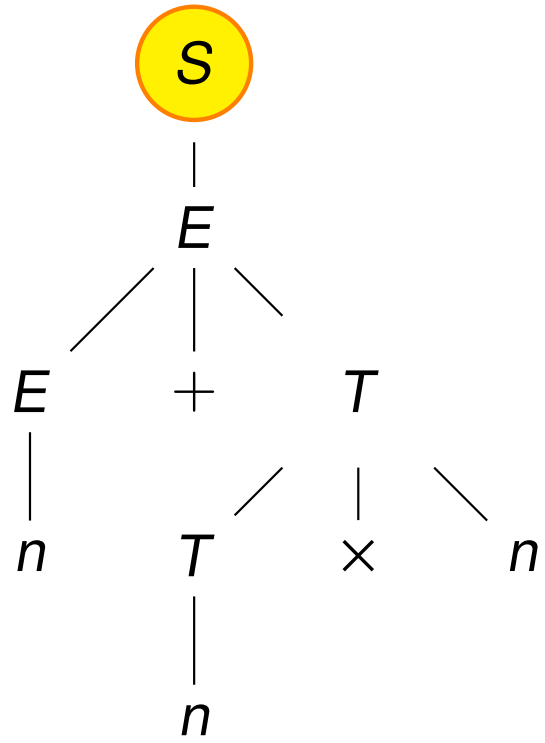
$$\big(n + \ (n \times n)\ \big)$$

How does the parsing work?

Implicitly, we refer to a context-free grammar:

$$
\begin{aligned}
S &\to E \\
E &\to E + T \\
E &\to T \times n \\
E &\to n \\
T &\to T \times n \\
T &\to n
\end{aligned}
$$

$n + n \times n$



Suppose you have a method for inserting the proper parentheses into an expression . . . you get the **stencil** of the parsing tree

$$\big(n + \ (n \times n)\ \big)$$

Precedence table

Whenever a parenthesis could be inserted between two symbol, we introduce a **parentheses generator**

- open-parentheses generator $\lessdot$
- closed-parentheses generator $\gtrdot$
- no parentheses $\doteq$

If $a \lessdot b$ we say that a **yields precedence**
if $a \gtrdot b$ we say that a **takes precedence**

For each pair of symbols we determine the admissible parentheses generator and obtain the **precedence table**

Precedence table - example

Grammar

$$S \to E$$
$$E \to E + T$$
$$E \to T \times n$$
$$E \to n$$
$$T \to T \times n$$
$$T \to n$$

Precedence table - example

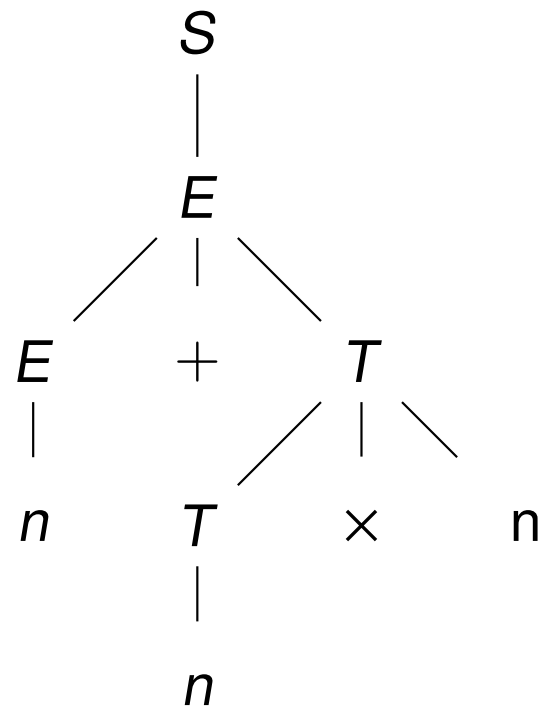
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n			
$+$			
$\times$			

Derivation tree



Precedence table - example

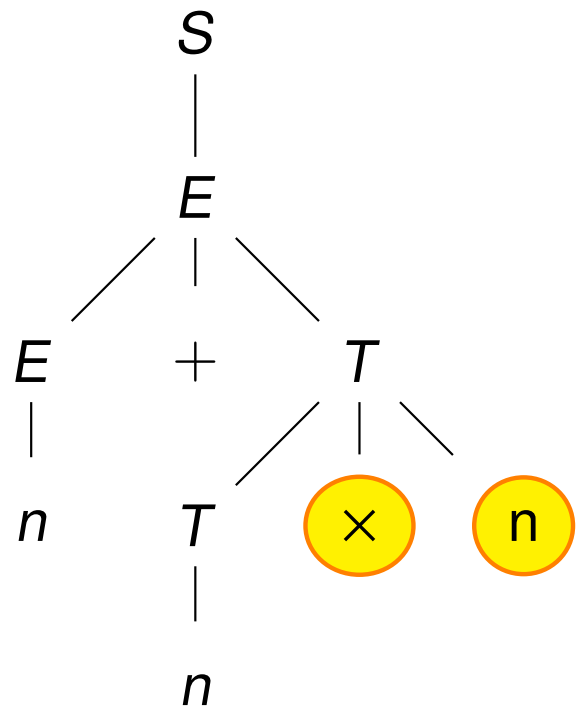
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n			
$+$			
$\times$	$\doteq$		

Derivation tree



Precedence table - example

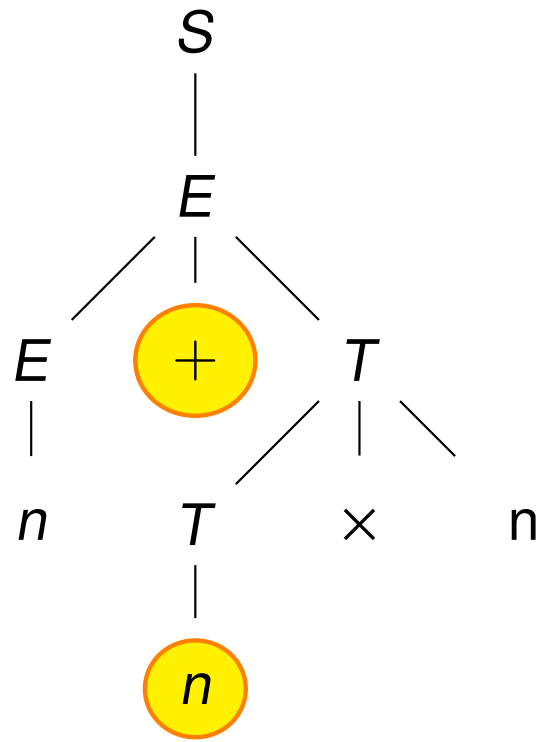
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n			
$+$	$\lessdot$		
$\times$	$\doteq$		

Derivation tree



Precedence table - example

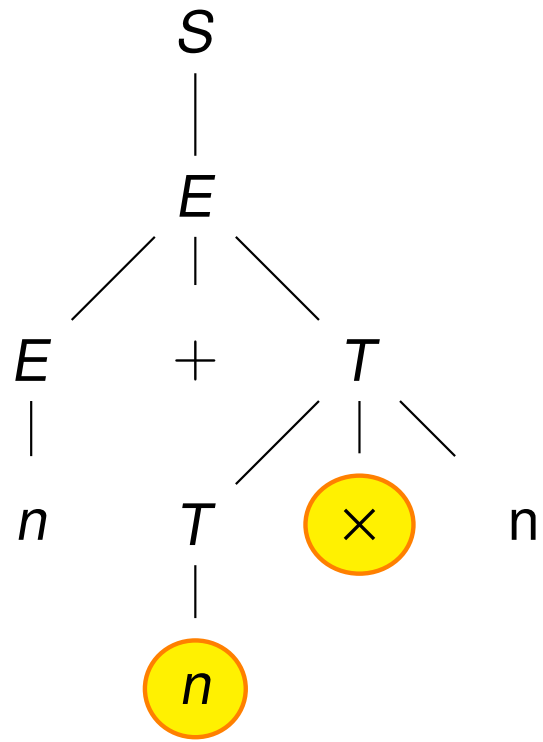
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n			$\gtrdot$
$+$	$\lessdot$		
$\times$	$\doteq$		

Derivation tree



Precedence table - example

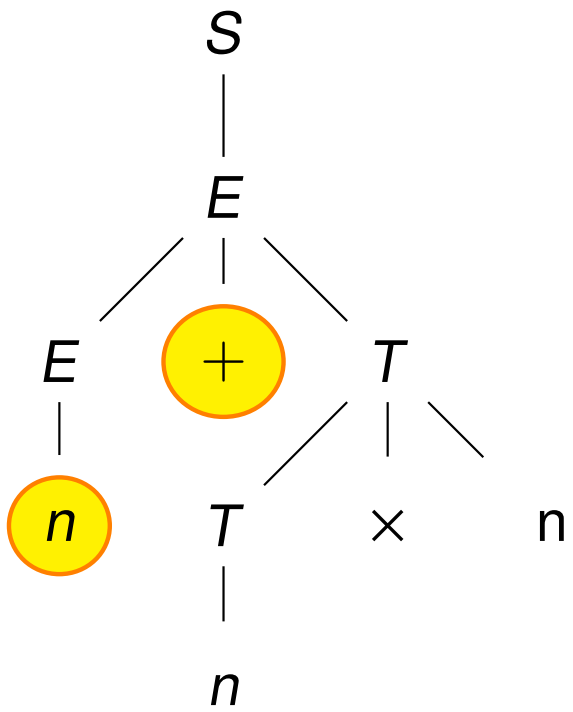
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n		$\gtrdot$	$\gtrdot$
$+$	$\lessdot$		
$\times$	$\doteq$		

Derivation tree



Precedence table - example

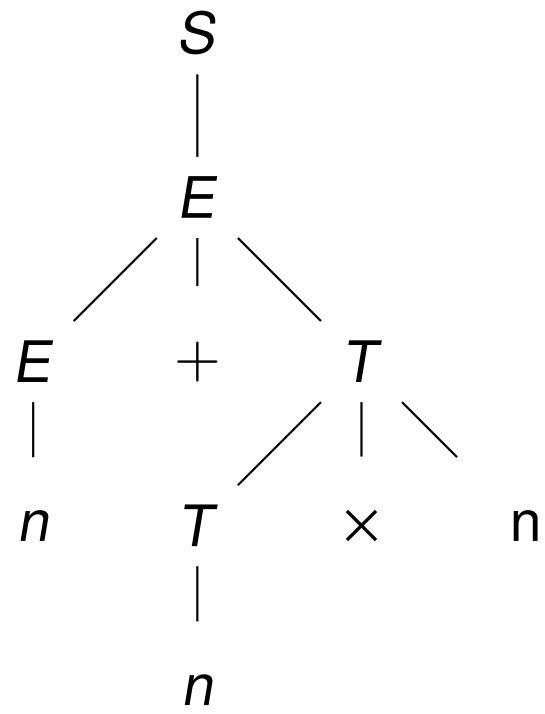
Grammar

$S \to E$
$E \to E + T$
$E \to T \times n$
$E \to n$
$T \to T \times n$
$T \to n$

Precedence table

	n	$+$	$\times$
n		$\gtrdot$	$\gtrdot$
$+$	$\lessdot$	$\gtrdot$	$\lessdot$
$\times$	$\doteq$		

Derivation tree



Operator precedence grammars, or Floyd grammars

- A CF grammar is an **operator grammar** if each right-hand side is not empty and no right-hand side contains two consecutive terminals. (Every grammar admits an equivalent operator precedence grammar.)
- An operator grammar is a **operator-precedence grammar** if there are no conflicts in its precedence matrix (given a pair of subsequent symbols, there is at most one parenthesis generator).

We name them **Floyd grammar**
in honour of their inventor Robert W. Floyd.

Outline

Automata for Floyd languages

The stack alphabet consists of pairs [a q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

Automata for Floyd languages

The stack alphabet consists of pairs $[\ a' \quad q\ ]$ where

- a is a symbol from the input alphabet (sort of),
- q is a state.

Variant: the symbol may be marked

Automata for Floyd languages

The stack alphabet consists of pairs [# q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

Variant: special starting symbol #

Automata for Floyd languages

The stack alphabet consists of pairs [a q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

3 kinds of moves, depending on the precedence relation between the symbol a on top of the stack and next input symbol b:

Automata for Floyd languages

The stack alphabet consists of pairs [a q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

3 kinds of moves, depending on the precedence relation between the symbol a on top of the stack and next input symbol b:

push move when $a \doteq b$
b is pushed on the stack

Automata for Floyd languages

The stack alphabet consists of pairs [a q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

3 kinds of moves, depending on the precedence relation between the symbol a on top of the stack and next input symbol b:

push move when $a \doteq b$
b is pushed on the stack

mark move when $a \lessdot b$
b' is pushed on the stack

Automata for Floyd languages

The stack alphabet consists of pairs [a q] where

- a is a symbol from the input alphabet (sort of),
- q is a state.

3 kinds of moves, depending on the precedence relation between the symbol a on top of the stack and next input symbol b:

push move when $a \doteq b$
b is pushed on the stack

mark move when $a \lessdot b$
b' is pushed on the stack

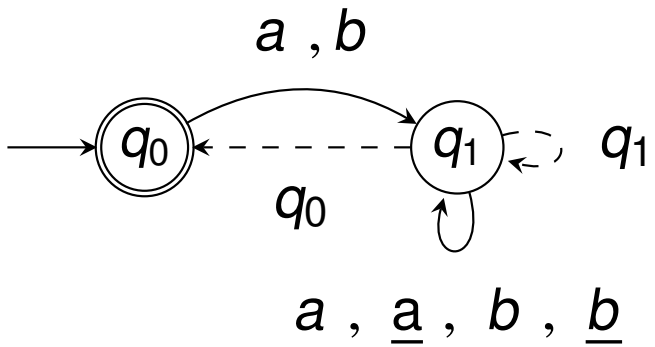
flush move when $a \gtrdot b$
symbols are popped from the stack until a marked symbol is found; b is not consumed

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis, with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



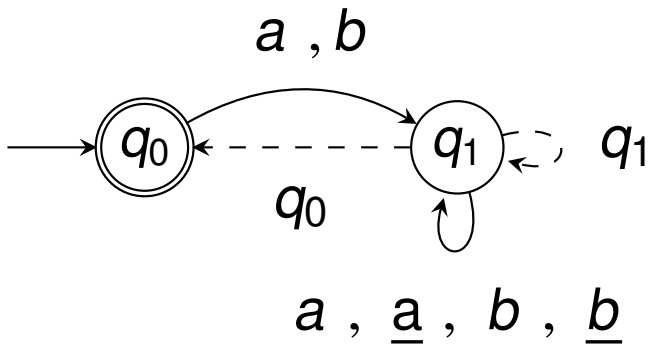
$aba\underline{ab}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \, q_0]$

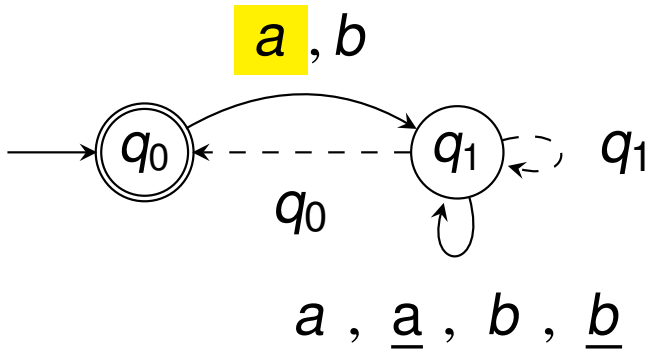
$aba\underline{ab}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



mark

$[\# \, q_0]$

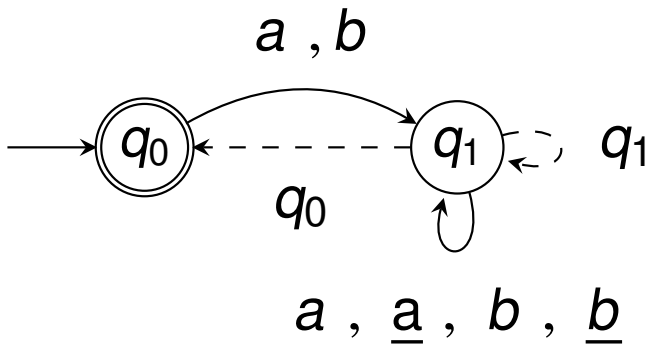
$aba\underline{ba}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1]$

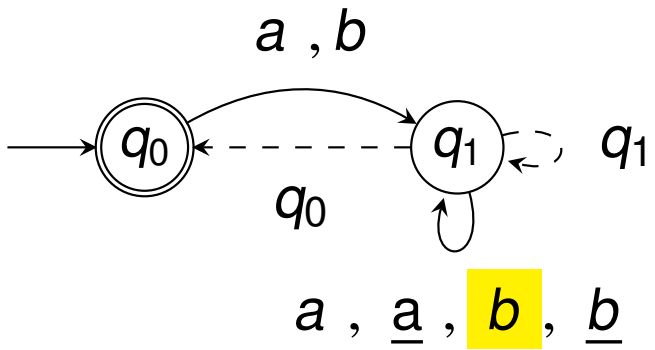
$aba\underline{ab}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



mark

$[\# \; q_0][a' \; q_1]$

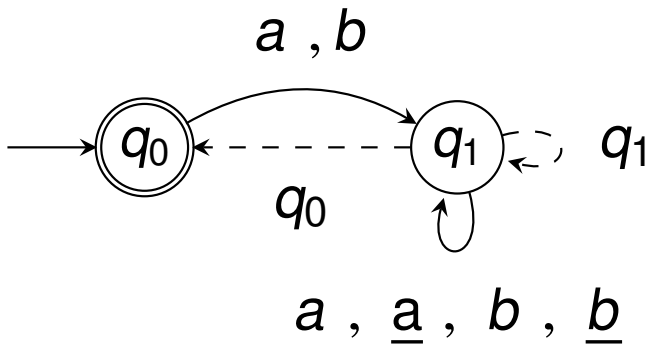
$aba\underline{ab}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][b' \; q_1]$

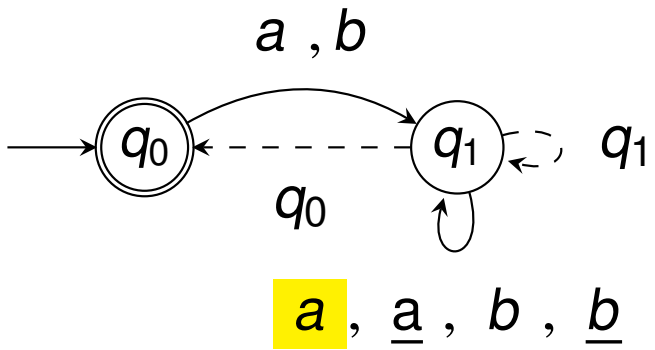
$ab\underline{aabaa}\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



mark

$[\# \; q_0][a' \; q_1][b' \; q_1]$

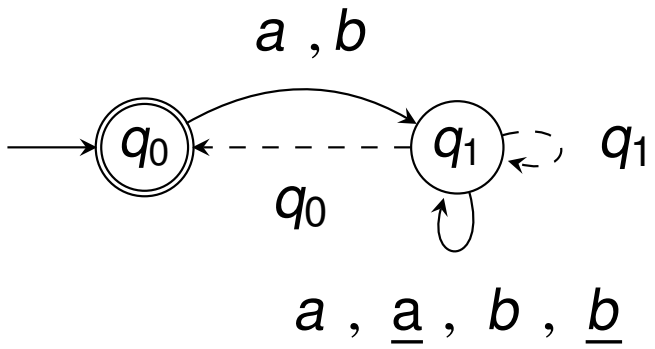
$ab\underline{aabaa}\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][b' \; q_1][a' \; q_1]$

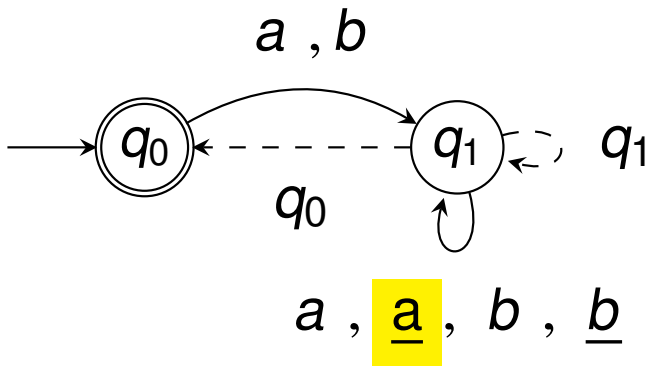
$aba\underline{ab}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



push

$[\# \; q_0][a' \; q_1][b' \; q_1][a' \; q_1]$

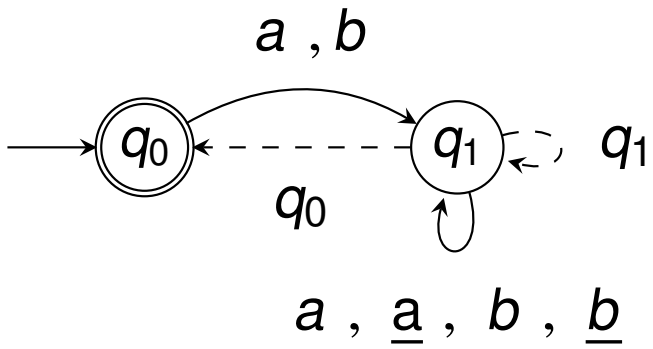
$aba\underline{aba}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



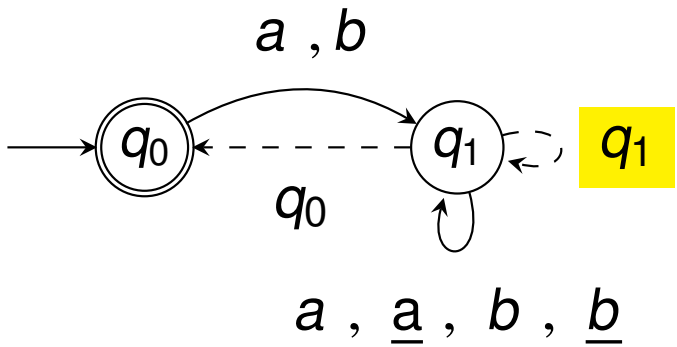
$[\# \; q_0][a' \; q_1][b' \; q_1][a' \; q_1][\underline{a} \; q_1]$ $abaa\underline{b}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



flush

$[\# \; q_0][a' \; q_1][b' \; q_1][a' \; q_1][\underline{a} \; q_1]$

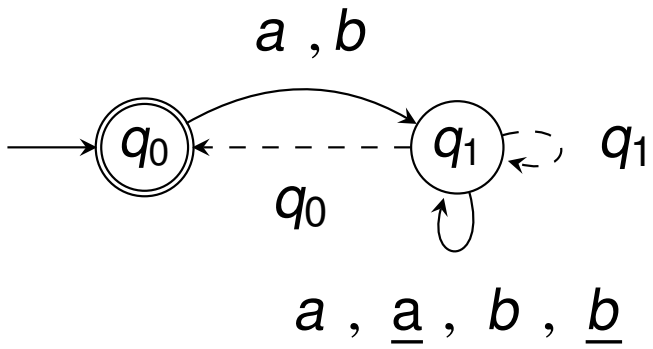
$abaa\underline{b}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][b' \; q_1]$

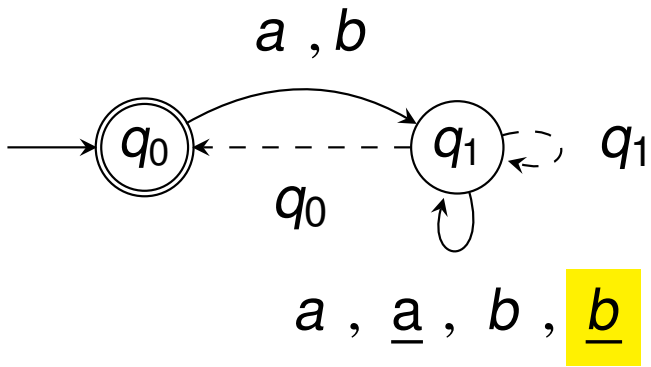
$abaa\underline{b}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



push

$[\# \; q_0][a' \; q_1][b' \; q_1]$

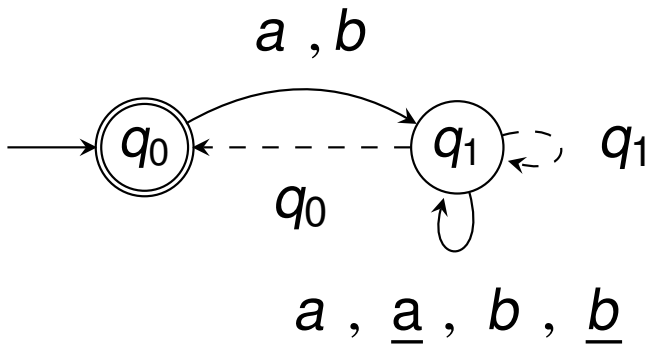
$abaa\underline{b}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][b' \; q_1][\underline{b} \; q_1]$

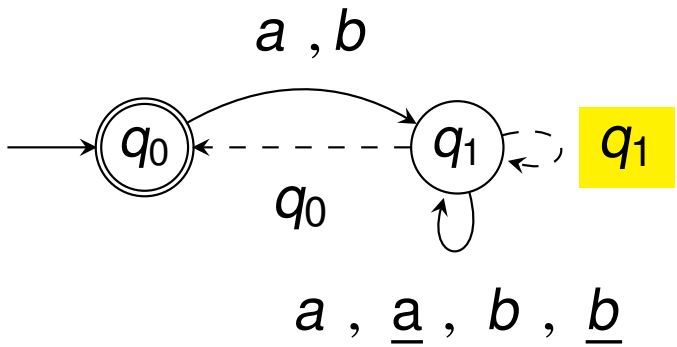
$abaab\underline{a}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



flush

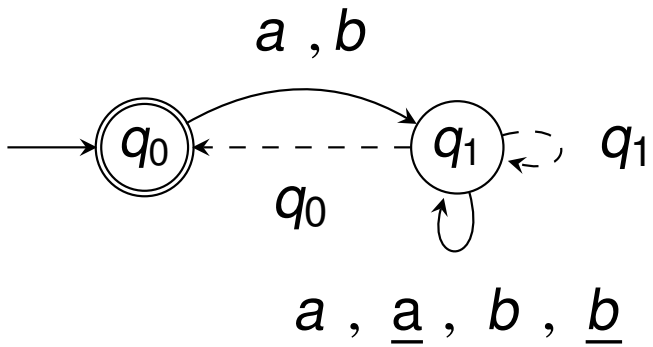
$[\# \; q_0][a' \; q_1][b' \; q_1][\underline{b} \; q_1]$

$abaab\underline{a}a\underline{a}\#$

An automata model for Floyd languages - example

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1]$

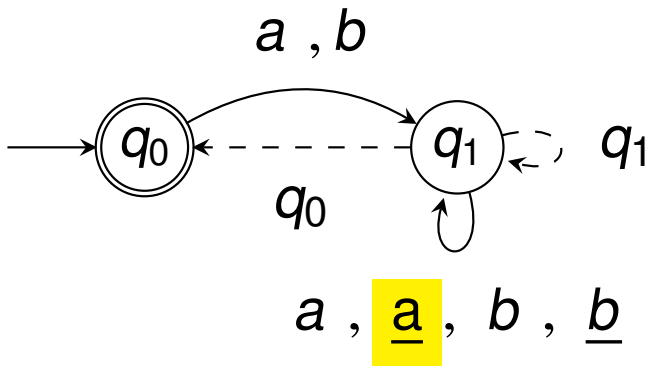
$abaab\underline{a}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



push $[\# \; q_0][a' \; q_1]$

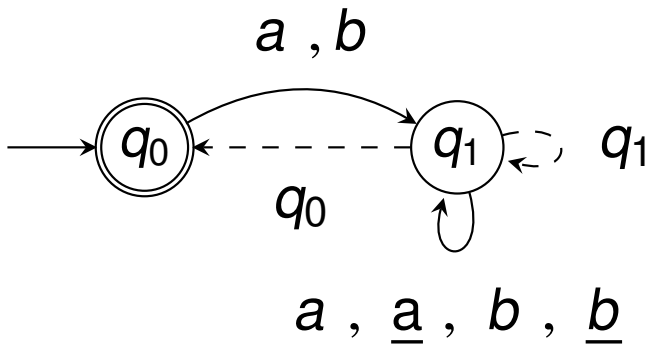
$abaab\underline{a}a\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][\underline{a} \; q_1]$

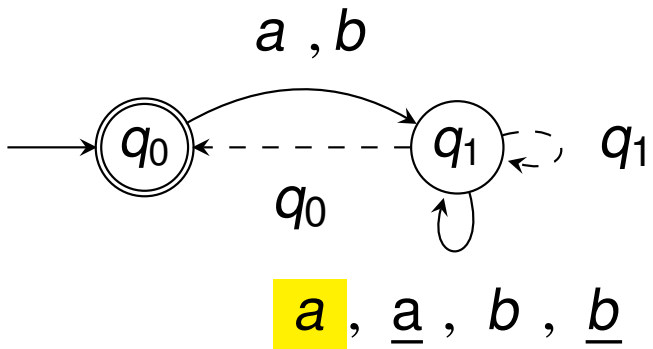
$abaaba\underline{a}\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



mark

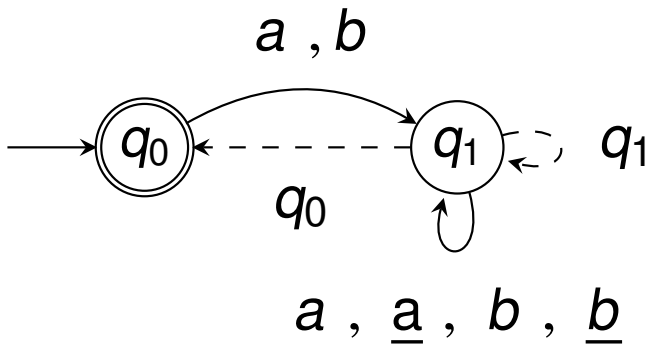
$[\# \; q_0][a' \; q_1][\underline{a} \; q_1]$

$abaa\underline{ba}a\underline{a}\#$

An automata model for Floyd languages - example

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][\underline{a} \; q_1][a' \; q_1]$

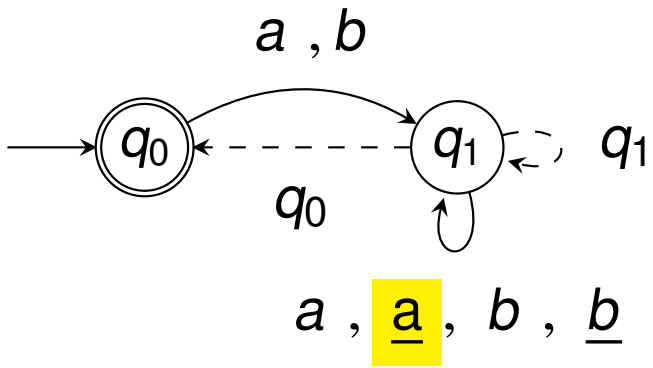
$abaa\underline{aba}\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



push

$[\# \; q_0][a' \; q_1][\underline{a} \; q_1][a' \; q_1]$

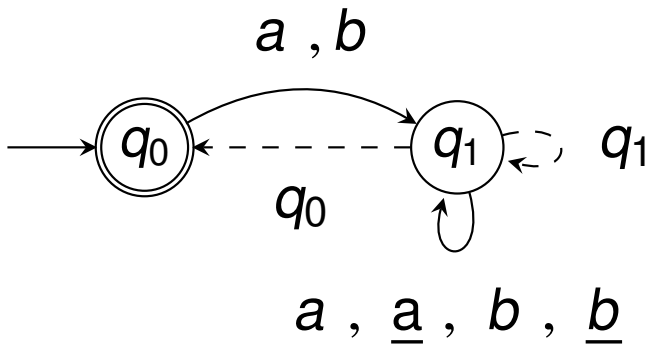
$abaa\underline{baa}\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



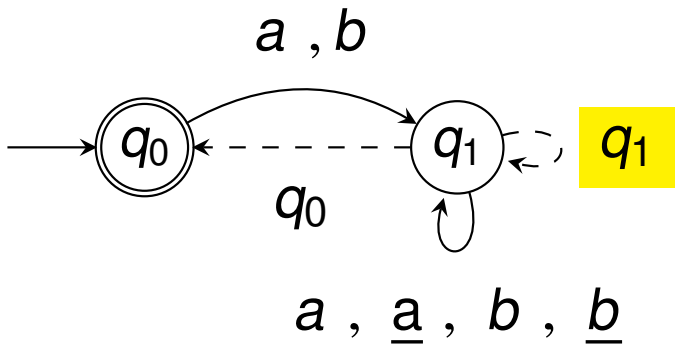
$[\# \; q_0][a' \; q_1][\underline{a} \; q_1][a' \; q_1][\underline{a} \; q_1]$ $abaabaa\underline{a}\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



flush

$[\# \; q_0][a' \; q_1][\underline{a} \; q_1][a' \; q_1][\underline{a} \; q_1]$

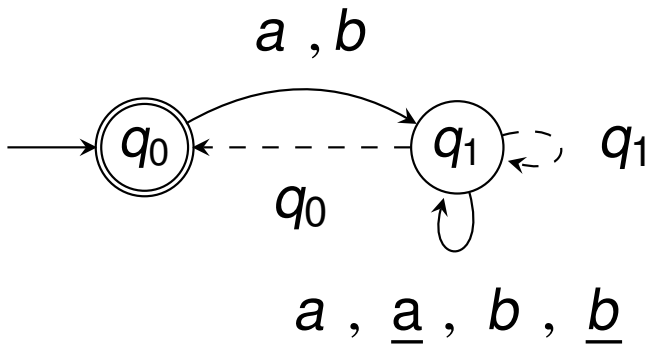
$abaabaaa\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0][a' \; q_1][\underline{a} \; q_1]$

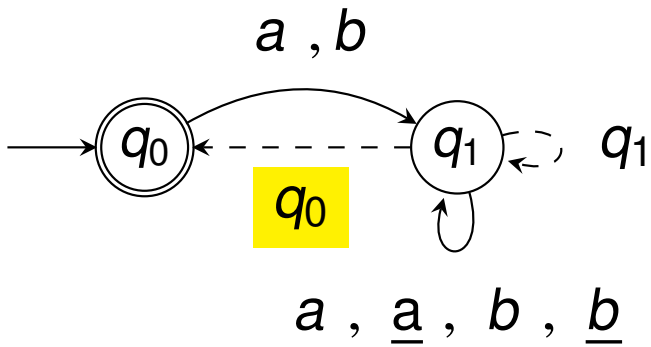
$abaabaaa\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



flush

$[\# \; q_0][a' \; q_1][\underline{a} \; q_1]$

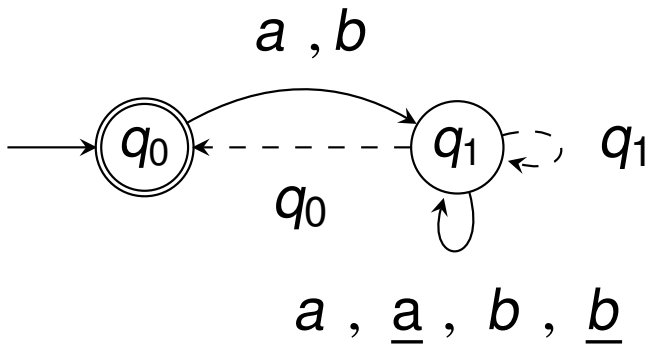
$abaabaaa\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



$[\# \; q_0]$

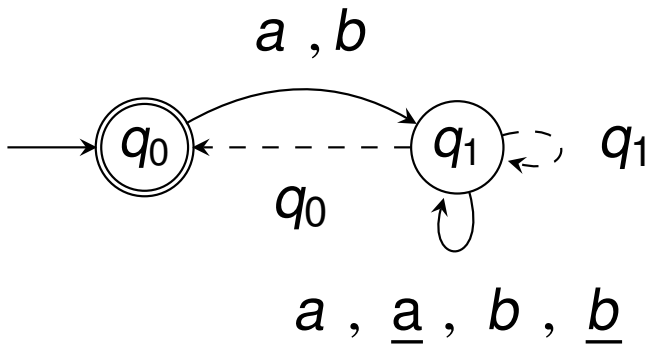
$abaabaaa\#$

An automata model for Floyd languages - example

restart skip

Dyck language of balanced string of parenthesis,
with 2 pairs $a, \underline{a}$ and $b, \underline{b}$.

	a	$\underline{a}$	b	$\underline{b}$	$\#$
a	$\lessdot$	$\doteq$	$\lessdot$		
$\underline{a}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
b	$\lessdot$		$\lessdot$	$\doteq$	
$\underline{b}$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
$\#$	$\lessdot$		$\lessdot$		$\doteq$



accept!

$[\# \, q_0]$

$abaabaaa\#$

Equivalence theorems

Theorem

It is easy to define a deterministic version of Floyd automata (as usual: require that the transition function is deterministic). Deterministic Floyd automata are equivalent to nondeterministic ones.

Theorem

Any language generated by a Floyd grammar can be recognized by a Floyd automaton.

Theorem

Any language recognized by a Floyd automaton can be generated by a Floyd grammar.

Outline

Ω-languages

Having an operational model that defines FLs, it is now easy to introduce extensions to infinite words, i.e. to **ω-languages**.

The classical **Büchi condition of acceptance** can be adapted to FAs: an infinite word x is accepted if and only if configurations with stack $[a\ q_f]$, where q_f is final, occur infinitely often in the infinite computation on x.

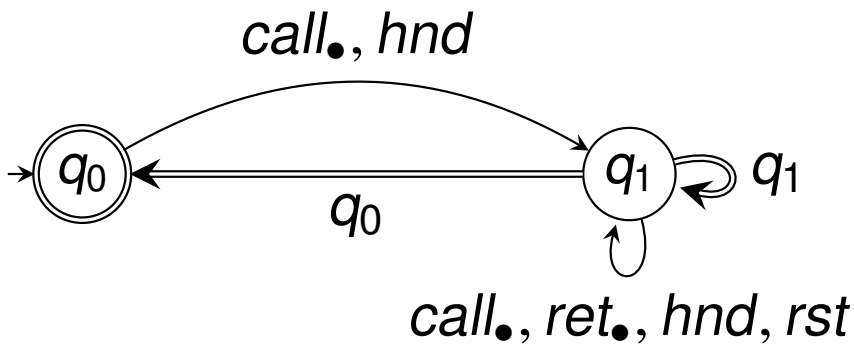
Example

We can model the stack management of a simple programming language that is able to handle nested exceptions

Ω-language - Example

- there are only two procedures, called *a* and *b*
- calls and returns are denoted by $call_a$, $call_b$, ret_a, ret_b
- during execution, it is possible to install an exception handler *hnd*
- signal *rst* is issued when an exception occur, or after a correct execution to uninstall the handler (with a *rst* the stack is "flushed", restoring the state right before the last *hnd*).

	$call_a$	ret_a	$call_b$	ret_b	*hnd*	*rst*
$call_a$	$\lessdot$	$\doteq$	$\lessdot$		$\lessdot$	$\gtrdot$
ret_a	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$		$\gtrdot$
$call_b$	$\lessdot$		$\lessdot$	$\doteq$	$\lessdot$	$\gtrdot$
ret_b	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$		$\gtrdot$
hnd	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		$\doteq$
rst	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$		
#	$\lessdot$		$\lessdot$		$\lessdot$	



Conclusions

Floyd grammars and languages

deserve renewed attention in the realm of formal languages

- FL family properly includes VPL
- FL family enjoys all closure properties of VPL and regular languages

Automaton

that perfectly matches the generative power of FGs

- extension to Ω-languages

Prototypical tool called *Flup*

`http://home.dei.polimi.it/pradella`

- interpreter for non-deterministic Floyd Automata
- FG to Floyd Automata translator

Further research

Parallel parsing

FLs can be parsed without applying a strictly left-to-right order [Google grant]

Logic characterization

In order to apply strong model checking techniques to this class of languages