Expression evaluation

Xinchen Yan

Mar. 12, 2012

Outline

- Introduction
- Scanner & tokens
- Parse tree
- Operator precedence & associativity
- An example
- Parsing table

Introduction

- Examples:
  - 1 + 2 + 3 = 6
  - 1 * 2 + 3 = 5
  - 1 + 2 * 3 = 7
  - (1 + 2) * 3 = 9
  - 1 – 2 + 3 = 2
  - 1 – (2 + 3) = -4
- Do computers know which to calculate first?
- Mush “teach” computers to read and understand expressions.

Tokens

- One more complicated example:

  - (123 * (456 – 938) – 12) / 5

- Tokens

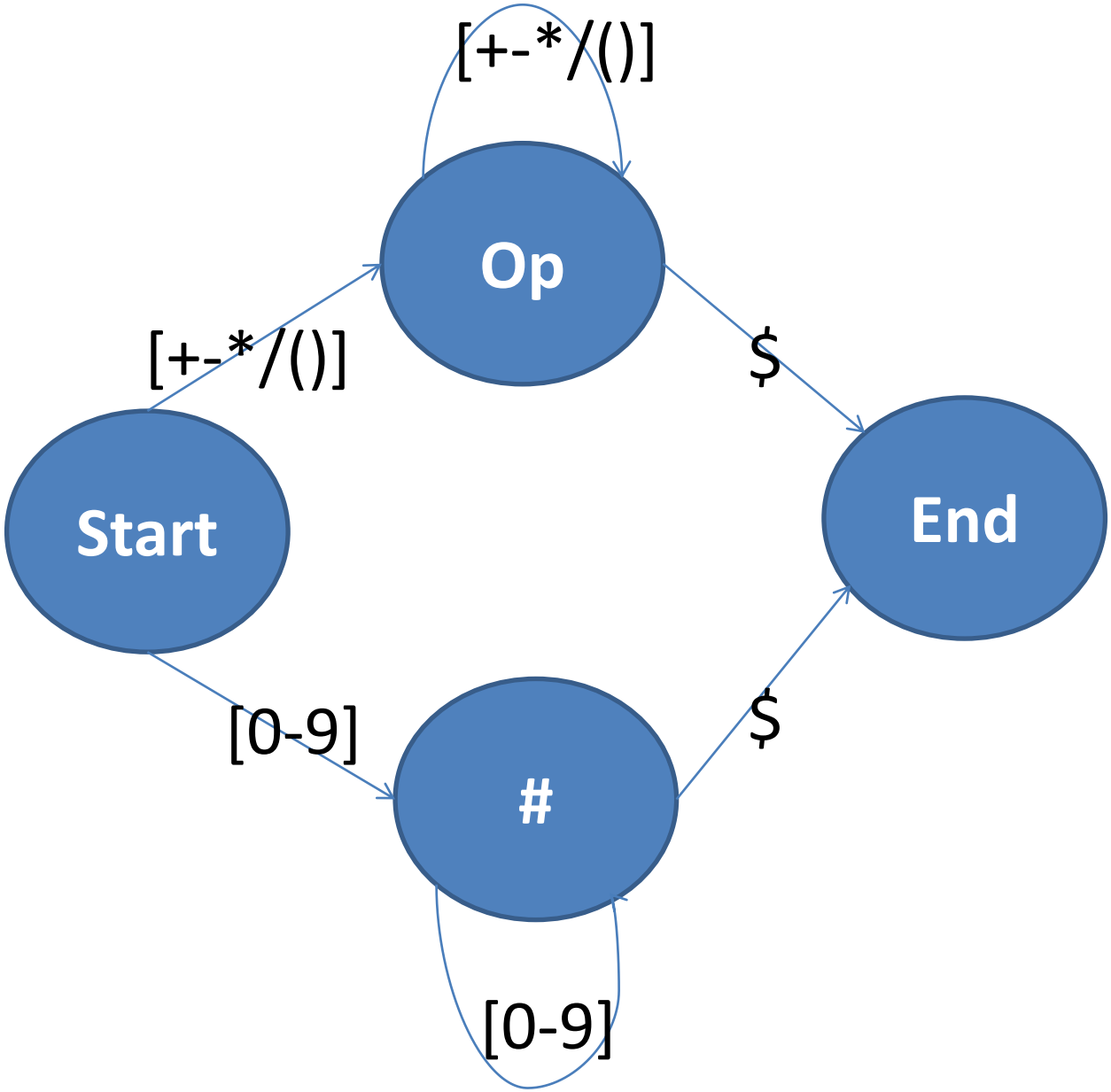
  - A **token** consists of two components, a token name and an attribute value.

  - tokens in the example above:

(	123	*	(	456	-	938	)	-	12	)	/	5
op	#	op	op	#	op	#	op	op	#	op	op	#

Scanner

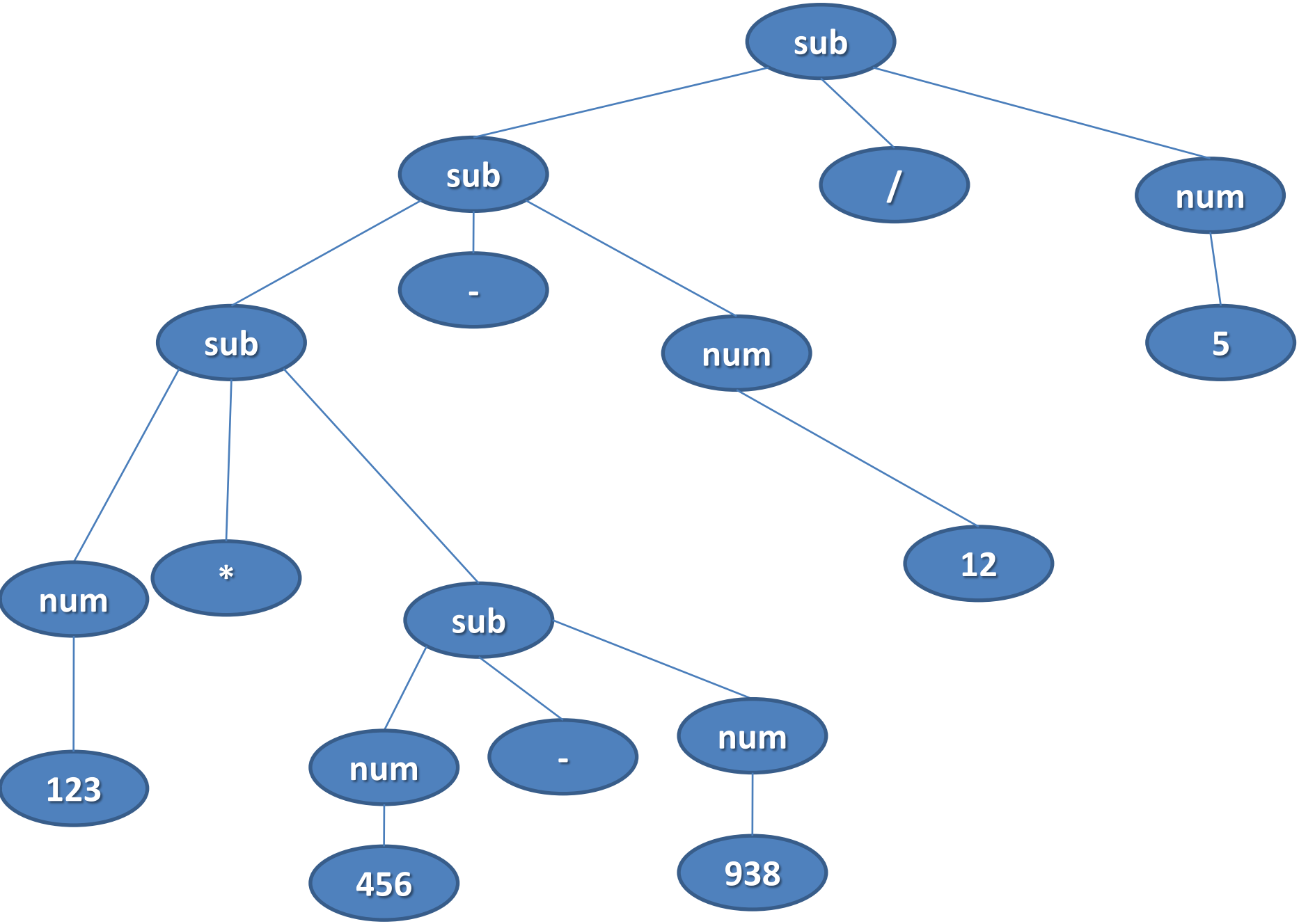
- How to extract tokens from expressions(strings)?
- Adding an ending symbol $
- Four states
  - op state
  - # state
  - Start state
  - End state


[+-*/()]
Op
[+-*/()]
$
Start
End
[0-9]
#
$
[0-9]

Parse tree

- Then, how to calculate from these tokens?
- Recall the introduction examples
- The underscored expression(sub-expr)

- Use sub-exprs to build a parse tree
- Parse tree of an expression illustrates the order of calculation.


sub
sub
/
num
sub
-
num
5
num
*
sub
12
num
-
num
123
456
938

Parse tree

- The structure of parse tree depends on two factors of expression.

  - operator precedence

  - operator associativity

Precedence & associativity

- Precedence
- + -
- * / %
- ^
- Associativity
- Left associative
  1 + 2 + 3 +4
- Right associative
  2 ^ 2 ^ 2

LR parsing

- Bottom-up shift-reduce parsing
- Stack is needed
- ***Shift.*** Shift the next input symbol onto the top of the stack
- ***Reduce.*** The right end of the string to be reduced must be at the top of the stack. Locate the left end of the string within the stack and decide with what nonterminal to replace the string.
- ***Accept.*** Annonce successful completion of parsing.
- ***Error.*** Discover a syntax error and call and call an error recovery routine.

An example

Step	Op stack	Num stack	Input	Action
1	$	$	4 * 2 + 1 $	Shift
2	$	$ 4	* 2 + 1 $	Shift
3	$ *	$ 4	2 + 1 $	Shift
4	$ *	$ 4 2	+ 1 $	Reduce
5	$	$ 8	+ 1 $	Shift
6	$ +	$ 8	1 $	Shift
7	$ +	$ 8 1	$	Reduce
8	$	$ 9	$	Accept

Parsing table

Top of op stack	Input		
	+	*	$
+	Reduce	Shift	Reduce
*	Reduce	Reduce	Reduce
$	Shift	Shift	Accept

Extended parsing table

	+	-	*	/	(	)	$
+	r	r	s	s	s	r	r
-	r	r	s	s	s	r	r
*	r	r	r	r	s	r	r
/	r	r	r	r	s	r	r
(	s	s	s	s	s	s	Err
)	r	r	r	r	Err	r	r
$	s	s	s	s	s	Err	Acc

OJ Problem

- 1033 表达式求值
- 2012 Hex表达式