

# *Syntactic Analysis*

## *Implementing a Parser* *LR parsing tables*

Copyright 2009, Pedro C. Diniz, all rights reserved.  
Students enrolled in the Compilers class at Instituto Superior Técnico (IST/UTL)  
have explicit permission to make copies of these materials for their personal use.

## Outline

- Implementing a Parser
- Shift-Reduce Parser Example
- Why is it hard to build a Parser Engine?
- LR(k) Parser Tables

## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques

## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques

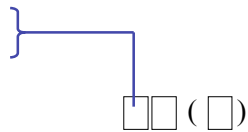
□ □ ( □ )



## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques

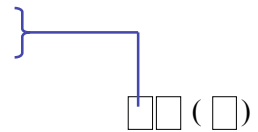
- **L** - parse from left to right
- **R** - parse from right to left



## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques

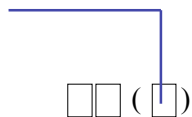
- **L** - leftmost derivation
- **R** - rightmost derivation



## Implementing a Parser

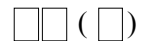
- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques

- Number of lookahead characters



## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques
  - Examples: LL(0), LR(1)





## Implementing a Parser

- Different Parsing Techniques
  - Each can handle some set of CFGs
  - Categorization of techniques
  - Examples: LL(0), LR(1)
- We will be studying LR(k) parsers

**LR (k)**



## Outline

- Implementing a Parser
- Shift-Reduce Parser Example
- Why is it hard to build a parser engine?
- LR(k) parser tables
- Constructing a LR(0) Parser Engine



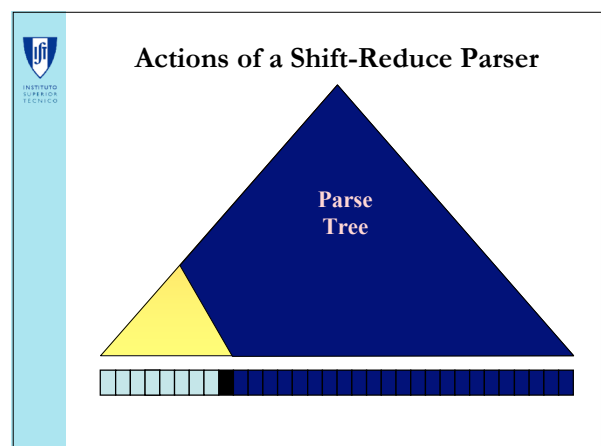
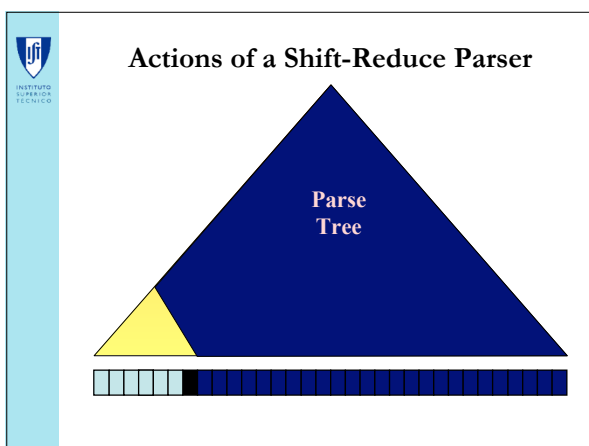
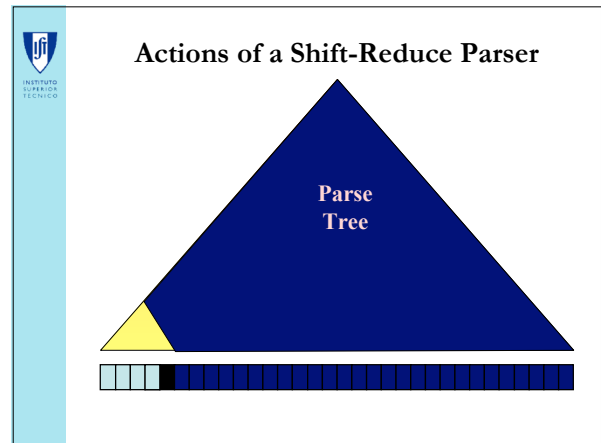
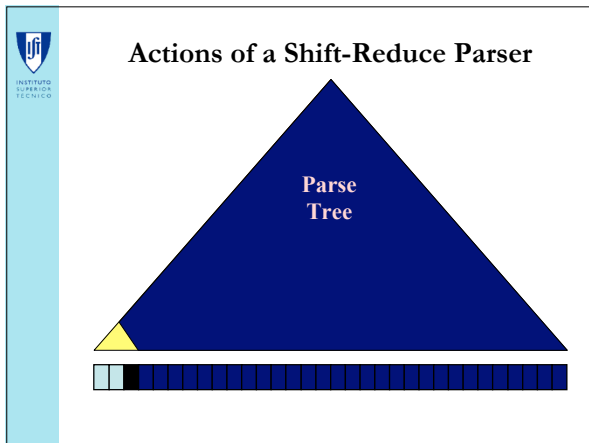
## Why use a LR(k) parser

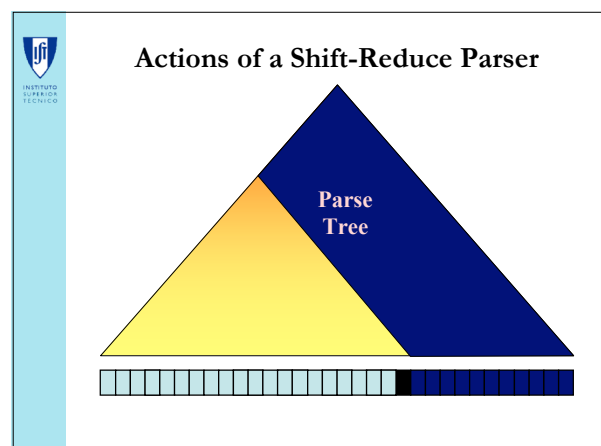
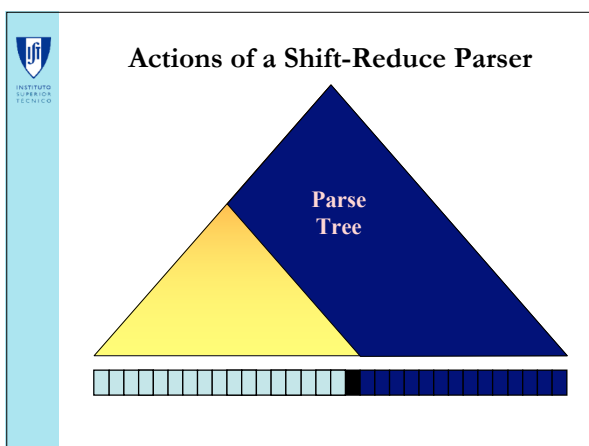
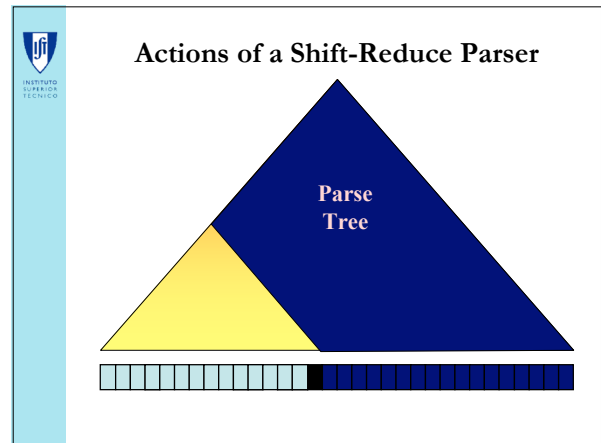
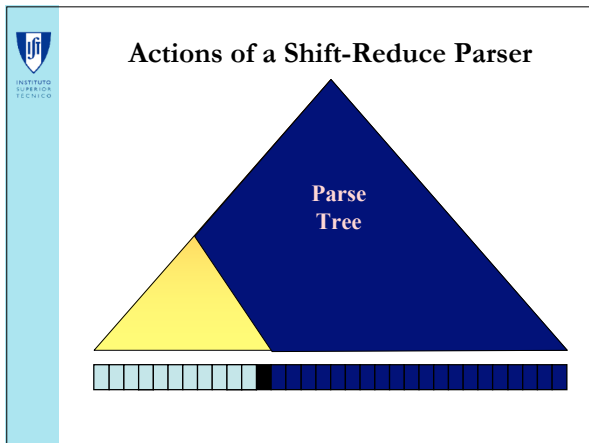
- Can be construct to recognize a very large class of CFGs
  - virtually all programming language constructs
- Most general non-backtracking parsing method
- Can build a very efficient very parser engine
- Can detect a syntactic error as soon as it is possible to do so

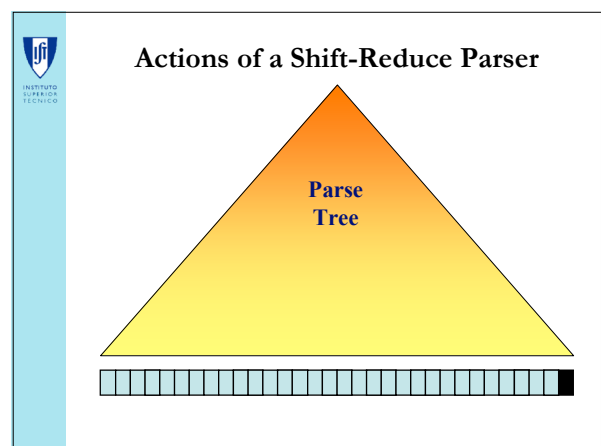
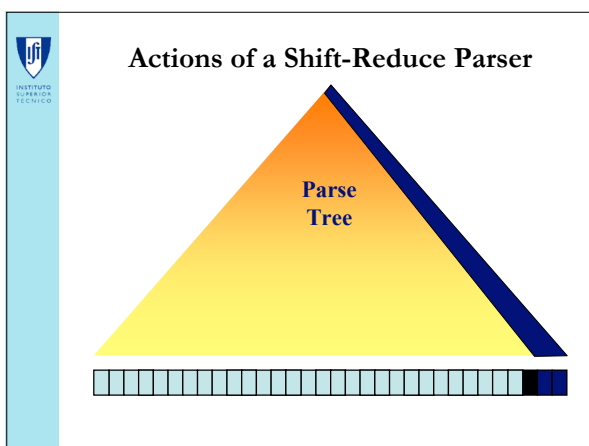
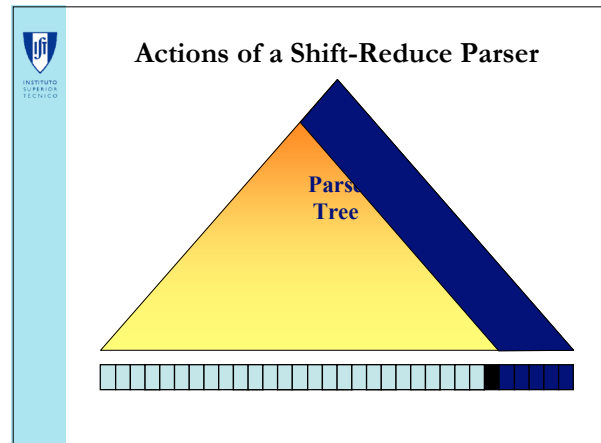
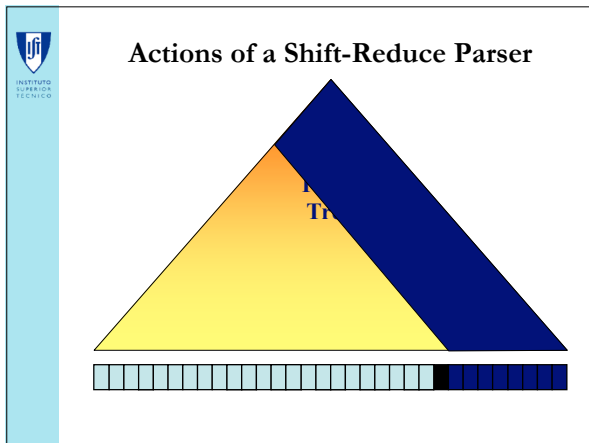


## Let's look at a Parser Implementation

- Workings of a LR(k) parser
- Parse from Left to Right
- Rightmost Derivation
  - start with the entire string
  - ends with the start symbol

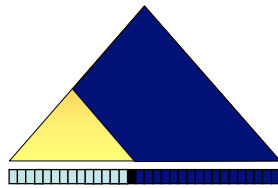






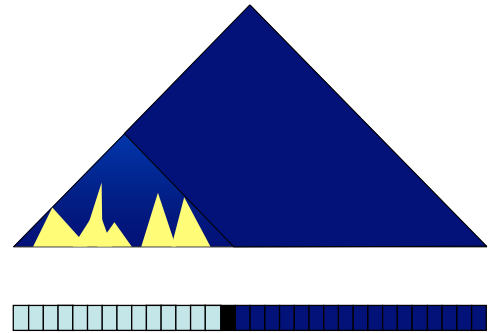
## Actions of a Shift-Reduce Parser

- Cannot create a full sub parse tree
- Need the look ahead information
- Thus, keep some state

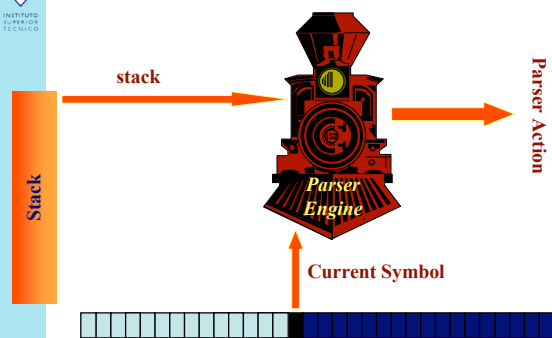


## Actions of a Shift-Reduce Parser

Current State



## Actions of a Shift-Reduce Parser



## Actions of a Shift-Reduce Parser

- Shift
  - Shift the current element into top of the stack
  - Move the current pointer
- Reduce
  - Apply a production
  - Top of the stack should match the RHS
  - Remove those symbols from the stack
  - Add the LHS non-terminal
- Accept
  - End of stream reached &
  - stack only has the start symbol
- Reject
  - End of stream reached but
  - stack has more than the start symbol



## Shift-Reduce Parser Example

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---



## Shift-Reduce Parser Example

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---



## Shift-Reduce Parser Example

$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$   
 $\langle \text{expr} \rangle \rightarrow ( \langle \text{expr} \rangle )$   
 $\langle \text{expr} \rangle \rightarrow - \langle \text{expr} \rangle$   
 $\langle \text{expr} \rangle \rightarrow \text{num}$   
 $\langle \text{op} \rangle \rightarrow +$   
 $\langle \text{op} \rangle \rightarrow -$   
 $\langle \text{op} \rangle \rightarrow *$

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---




## Shift-Reduce Parser Example

$\langle \text{expr} \rangle \rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle$   
 $\langle \text{expr} \rangle \rightarrow ( \langle \text{expr} \rangle )$   
 $\langle \text{expr} \rangle \rightarrow - \langle \text{expr} \rangle$   
 $\langle \text{expr} \rangle \rightarrow \text{num}$   
 $\langle \text{op} \rangle \rightarrow +$   
 $\langle \text{op} \rangle \rightarrow -$   
 $\langle \text{op} \rangle \rightarrow *$

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---






### Shift-Reduce Parser Example

$$\begin{aligned} \langle \text{expr} \rangle &\rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow ( \langle \text{expr} \rangle ) \\ \langle \text{expr} \rangle &\rightarrow - \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow \text{num} \\ \langle \text{op} \rangle &\rightarrow + \\ \langle \text{op} \rangle &\rightarrow - \\ \langle \text{op} \rangle &\rightarrow * \end{aligned}$$


num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---



### Shift-Reduce Parser Example

$$\begin{aligned} \langle \text{expr} \rangle &\rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow ( \langle \text{expr} \rangle ) \\ \langle \text{expr} \rangle &\rightarrow - \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow \text{num} \\ \langle \text{op} \rangle &\rightarrow + \\ \langle \text{op} \rangle &\rightarrow - \\ \langle \text{op} \rangle &\rightarrow * \end{aligned}$$

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---




### Shift-Reduce Parser Example

$$\begin{aligned} \langle \text{expr} \rangle &\rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow ( \langle \text{expr} \rangle ) \\ \langle \text{expr} \rangle &\rightarrow - \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow \text{num} \\ \langle \text{op} \rangle &\rightarrow + \\ \langle \text{op} \rangle &\rightarrow - \\ \langle \text{op} \rangle &\rightarrow * \end{aligned}$$

num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---

num

SHIFT



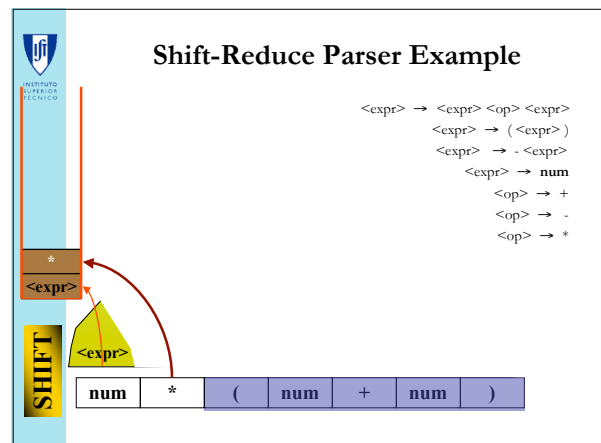
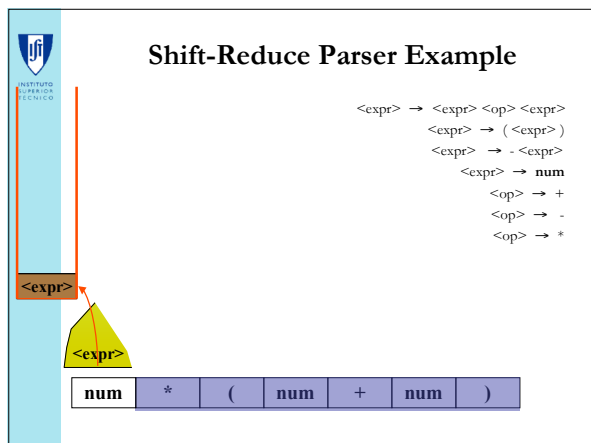
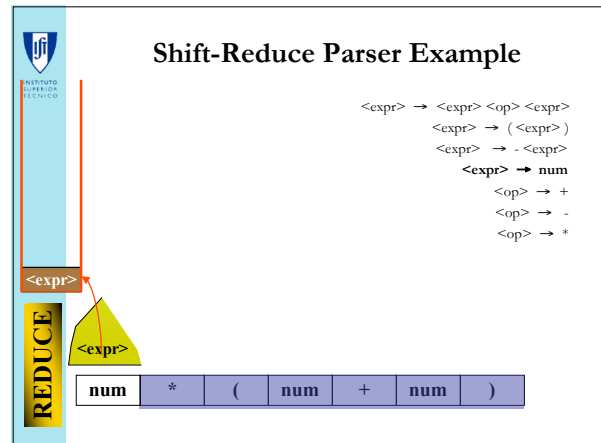
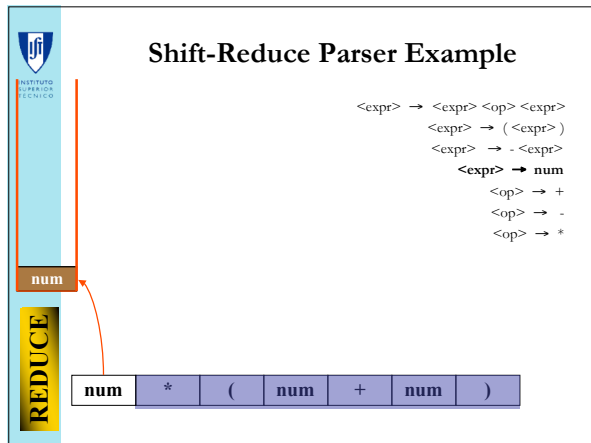
### Shift-Reduce Parser Example

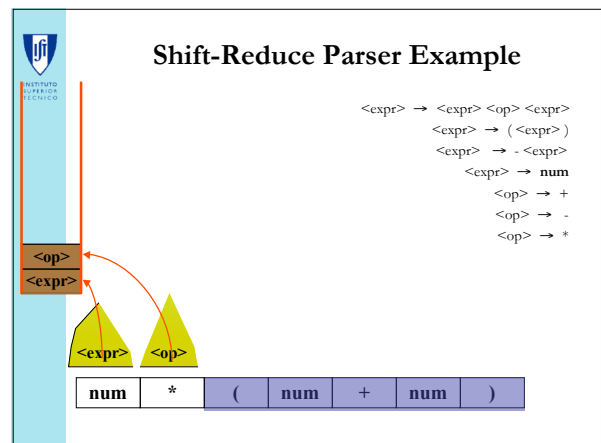
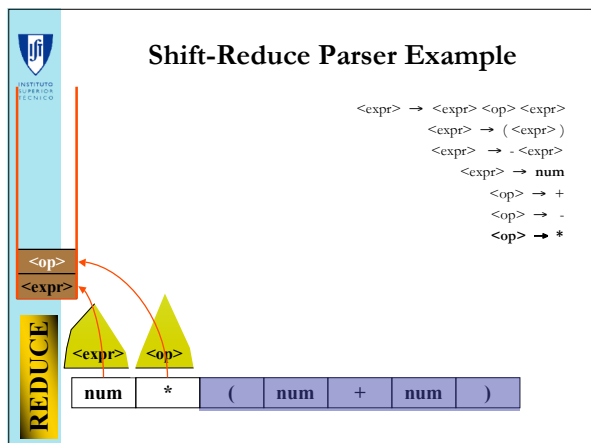
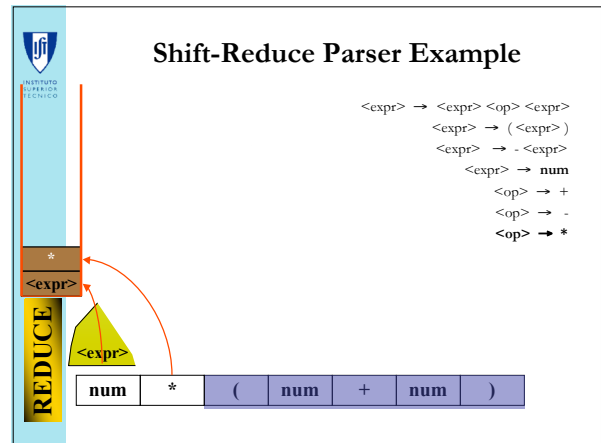
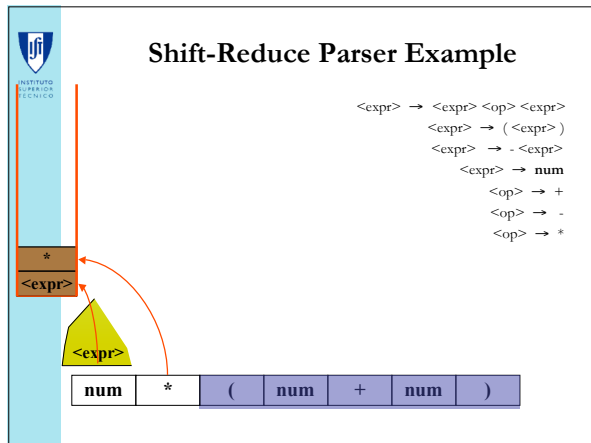
$$\begin{aligned} \langle \text{expr} \rangle &\rightarrow \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow ( \langle \text{expr} \rangle ) \\ \langle \text{expr} \rangle &\rightarrow - \langle \text{expr} \rangle \\ \langle \text{expr} \rangle &\rightarrow \text{num} \\ \langle \text{op} \rangle &\rightarrow + \\ \langle \text{op} \rangle &\rightarrow - \\ \langle \text{op} \rangle &\rightarrow * \end{aligned}$$

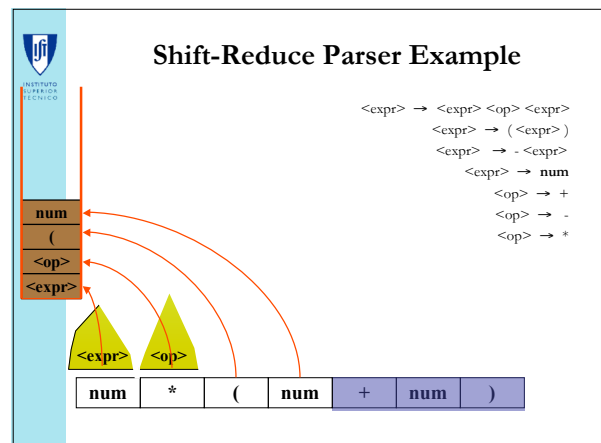
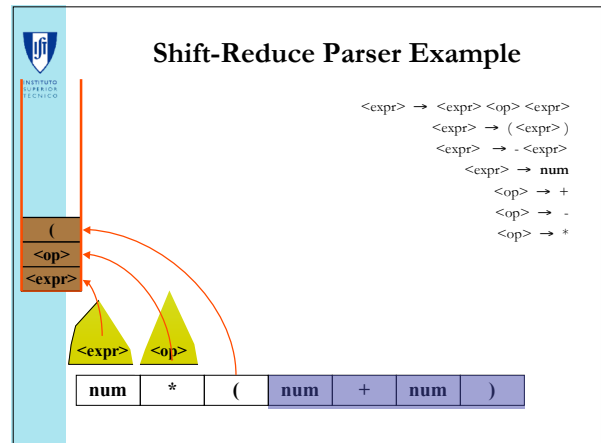
num	*	(	num	+	num	)
-----	---	---	-----	---	-----	---

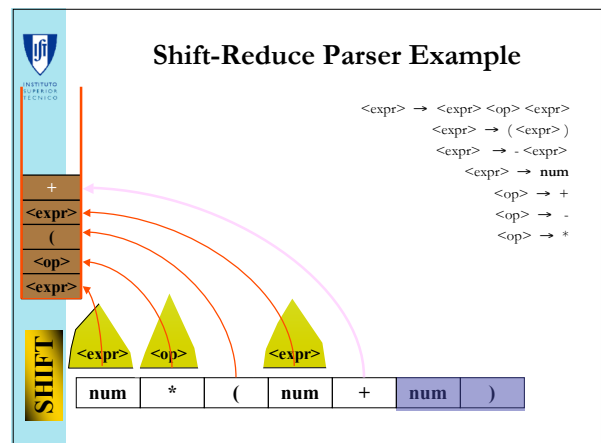
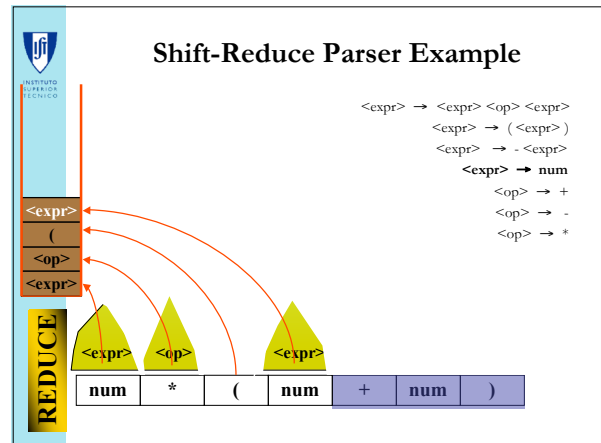
num

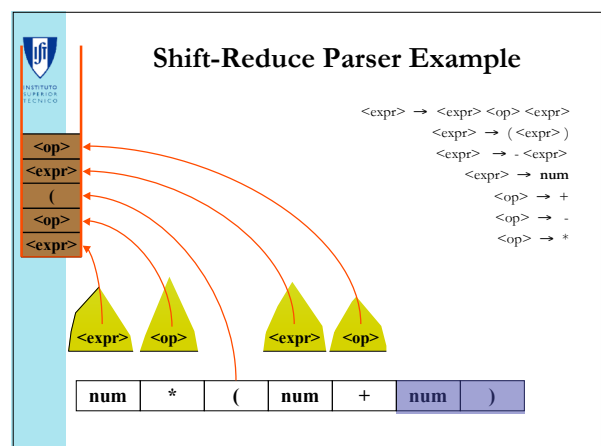
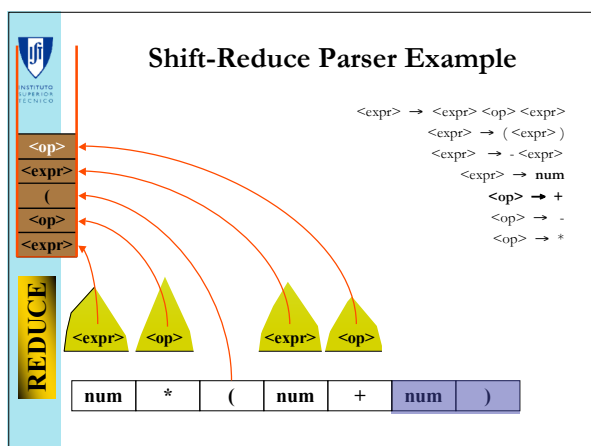
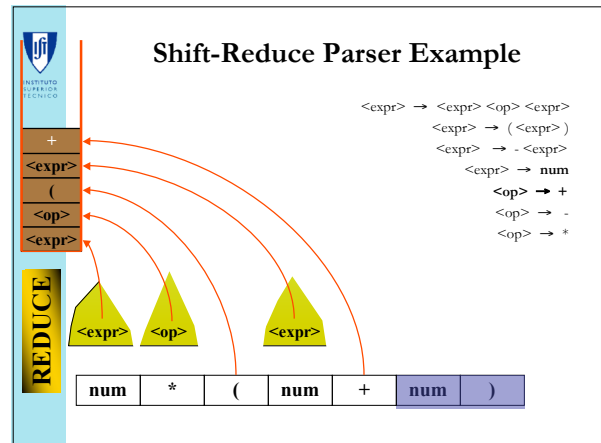
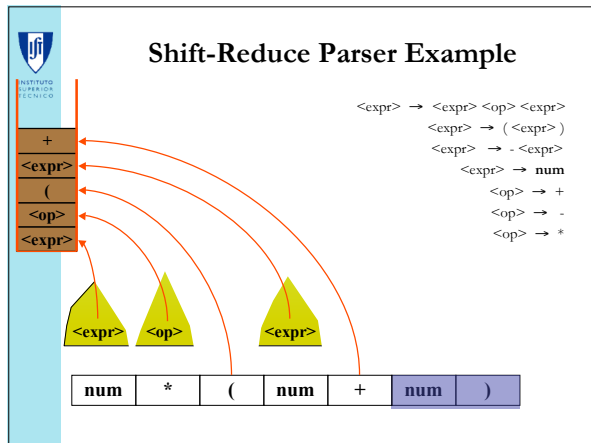
SHIFT

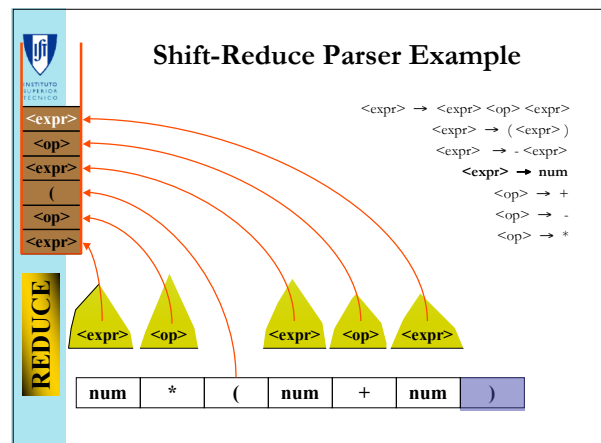
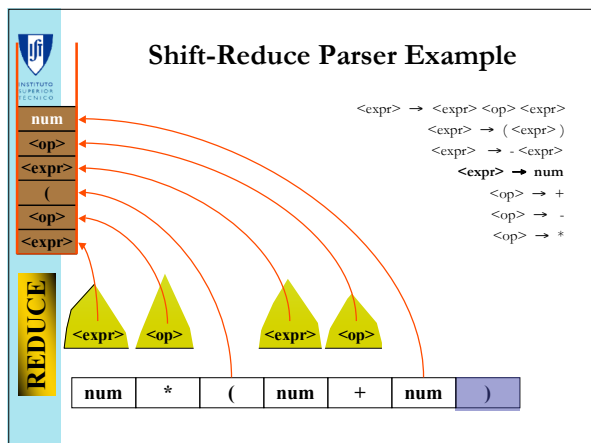
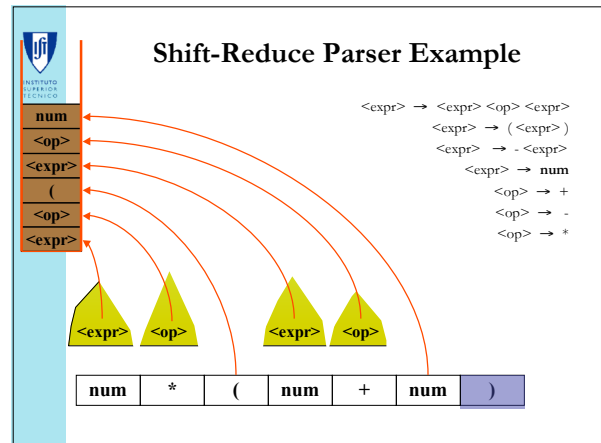
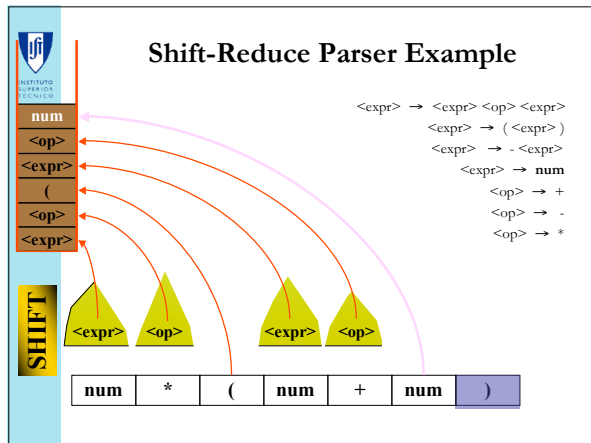


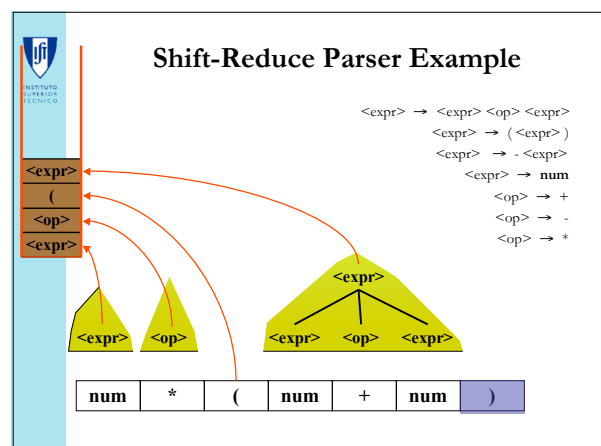
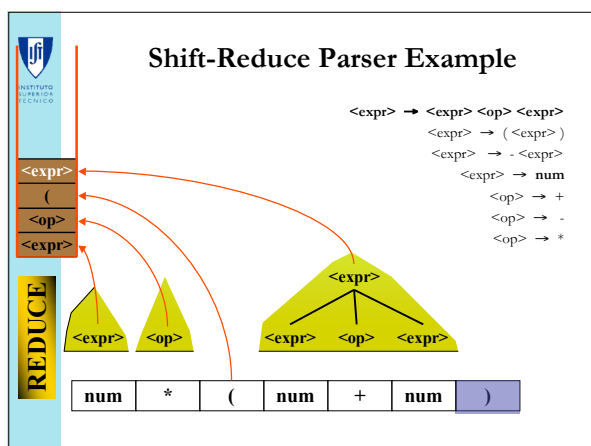
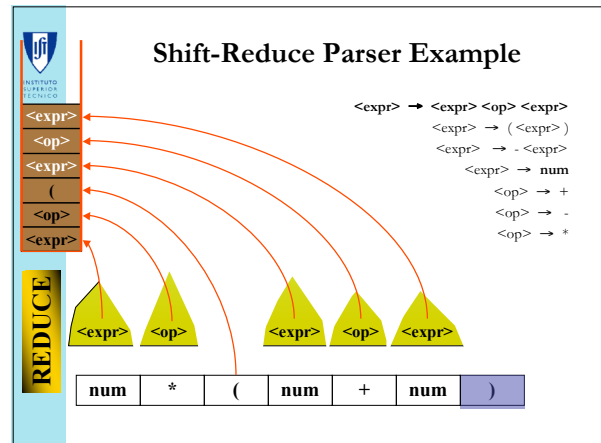
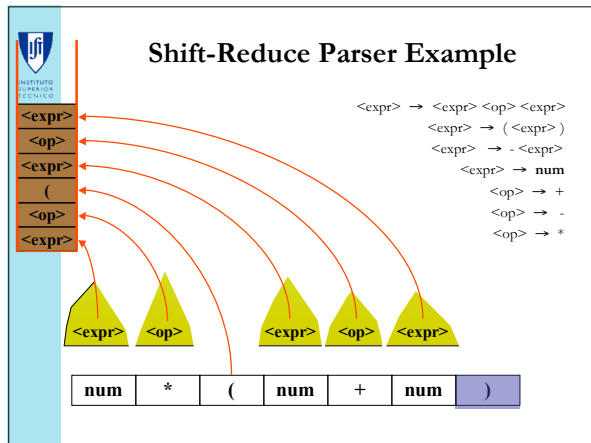




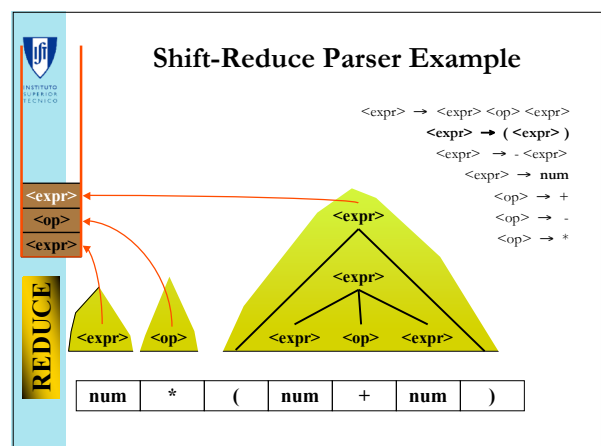
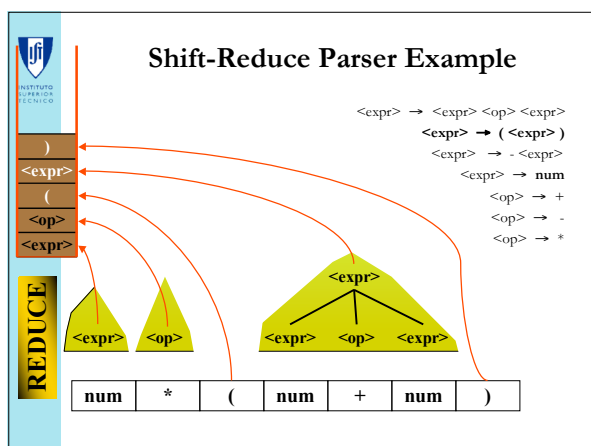
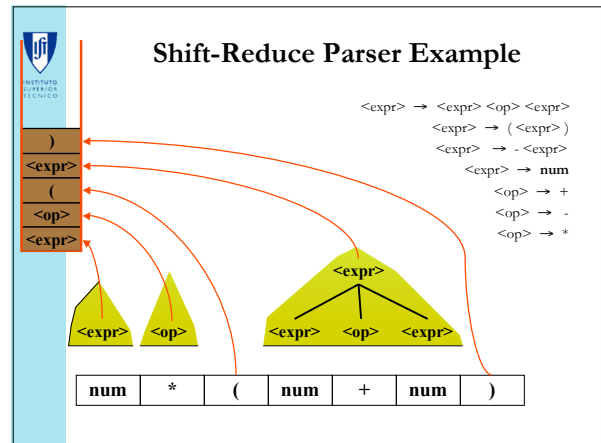
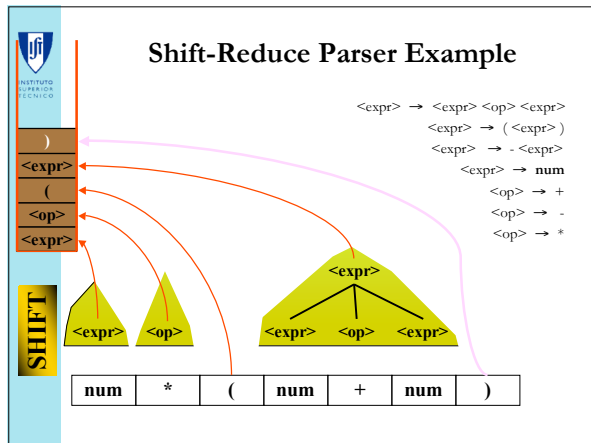


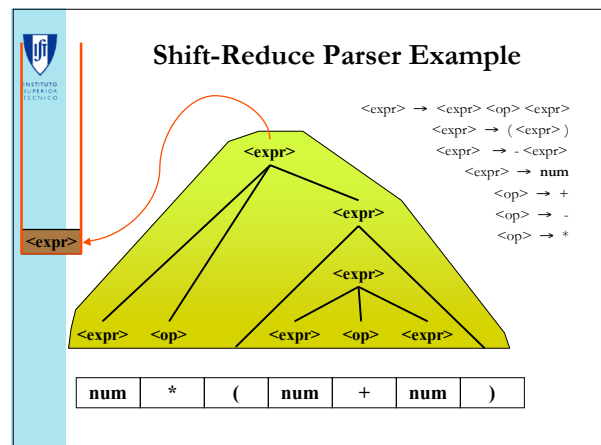
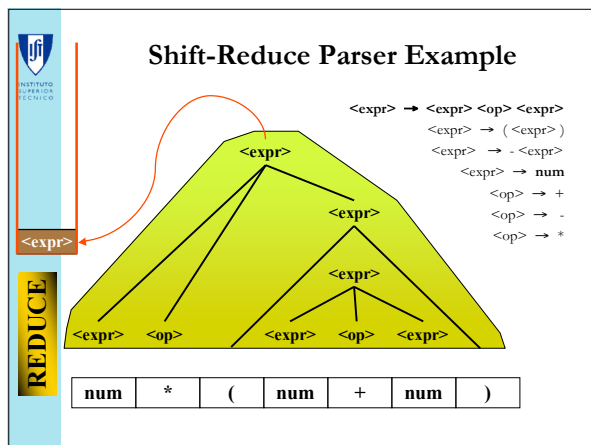
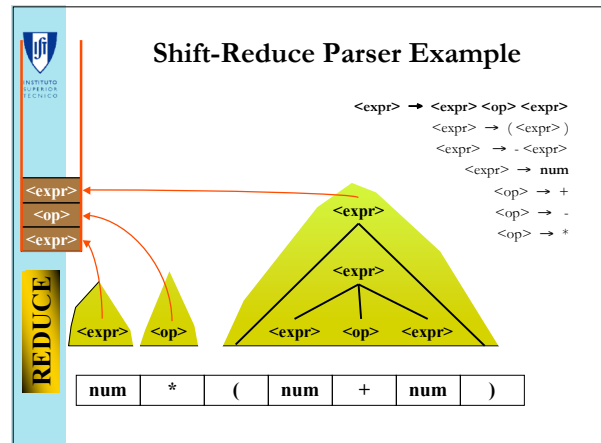
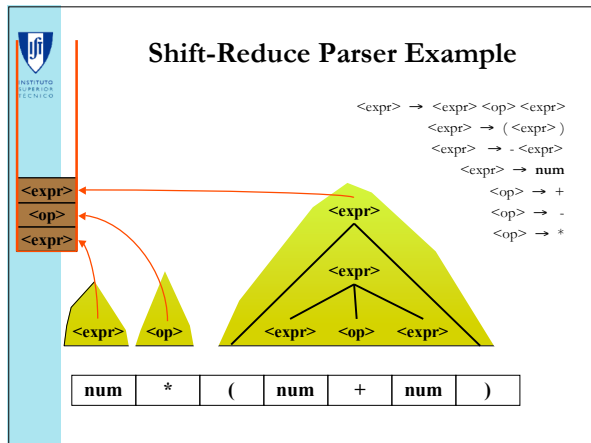


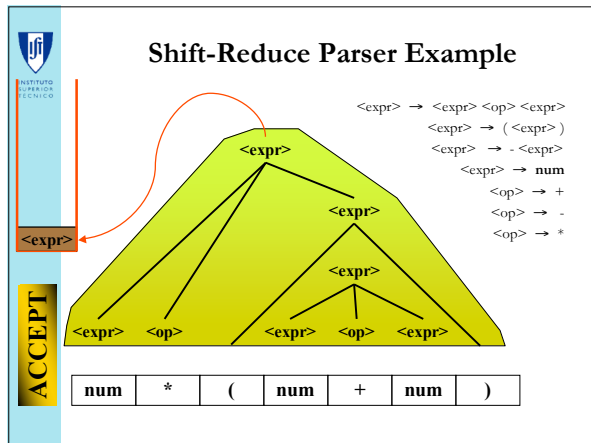












**What does the parser engine do?**

- If the top symbols of the stack match an RHS of a production do the reduction
  - Pop the RHS from the top of the stack
  - push the LHS symbol onto the stack
- If no production is found do the shift
  - push the current input into the stack
- If the input is empty
  - accept if only the start symbol is on the stack
  - reject otherwise

**Outline**

- Implementing a Parser
- Shift-Reduce Parser Example
- Why is it hard to build a parser engine?
- LR(k) parser tables
- Constructing a LR(0) Parser Engine

**What does the parser engine do?**

- If the top symbols of the stack match an RHS of a production do the reduction
  - Pop the RHS from the top of the stack
  - push the LHS symbol onto the stack
- If no production is found do the shift
  - push the current input into the stack
- If the input is empty
  - accept if only the start symbol is on the stack
  - reject otherwise



## This is not that simple!

- Many Choices of Reductions
  - Matches multiple RHS
- Choice between Shift and Reduce
  - Stack matches a RHS
  - But that may not be the right match
  - May need to shift an input and later find a different reduction



## Shift-Reduce Parser Example

### • Change in the Grammar

```
<expr> → <expr> <op> <expr>
<expr> → ( <expr> )
<expr> → - <expr>
<expr> → num
<op> → +
<op> → -
<op> → *
```



## Shift-Reduce Parser Example

### • Change in the Grammar

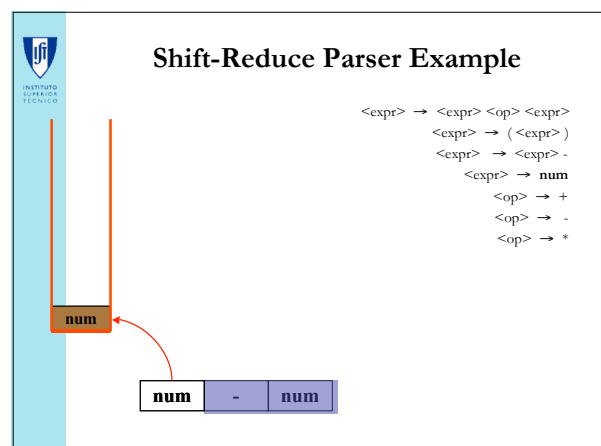
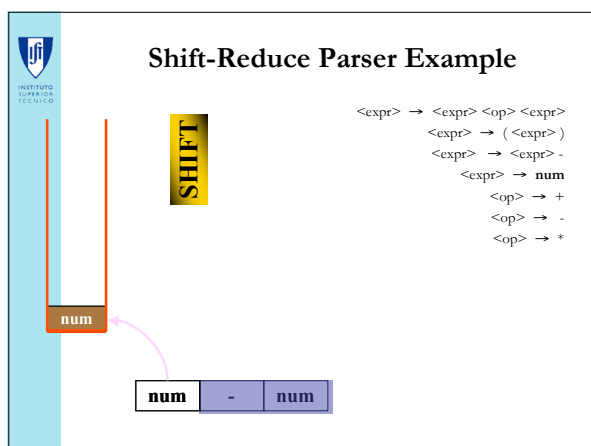
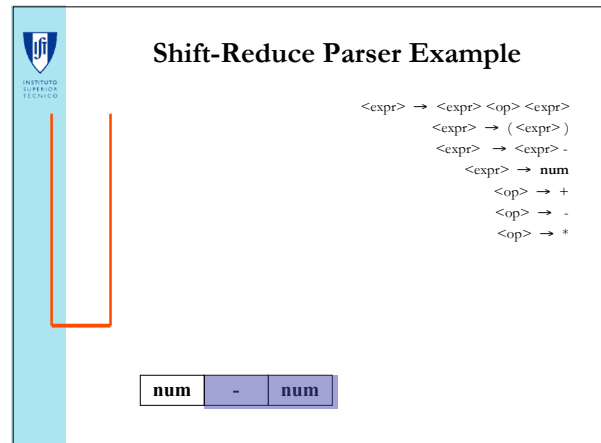
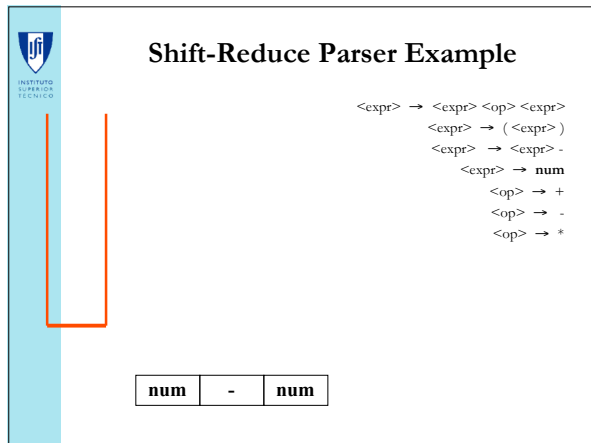
```
<expr> → <expr> <op> <expr>
<expr> → ( <expr> )
<expr> → - <expr>
<expr> → num
<op> → +
<op> → -
<op> → *
```



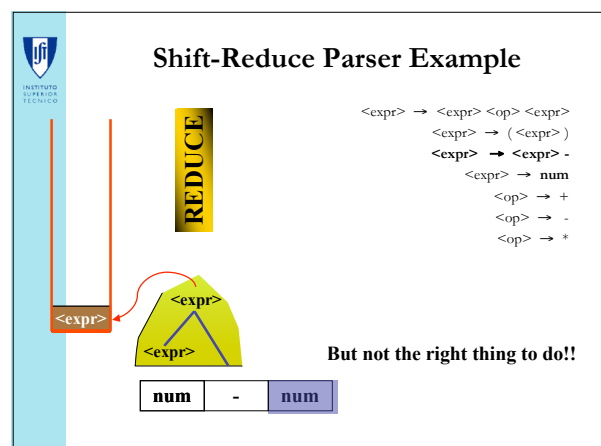
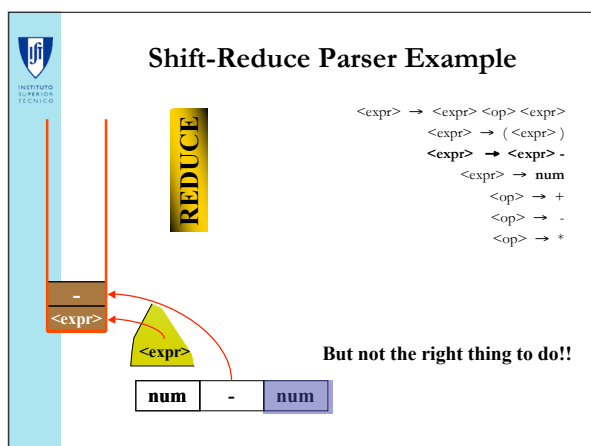
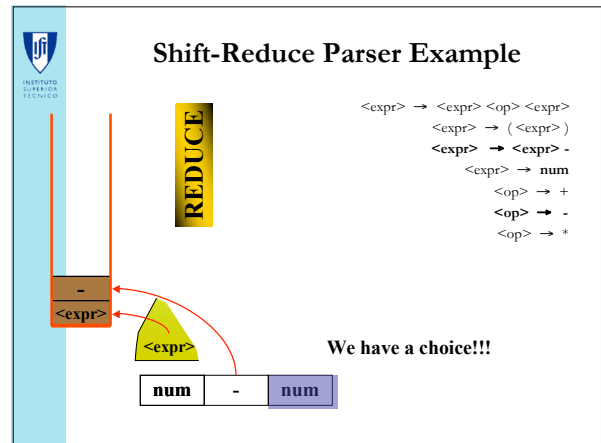
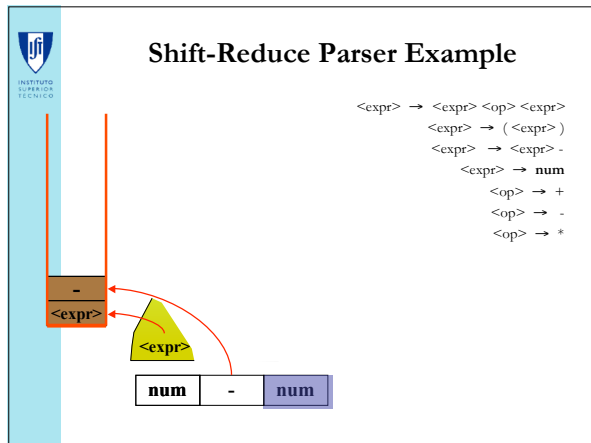
## Shift-Reduce Parser Example

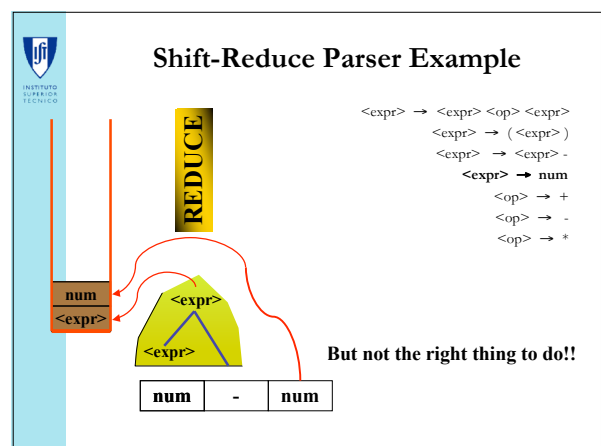
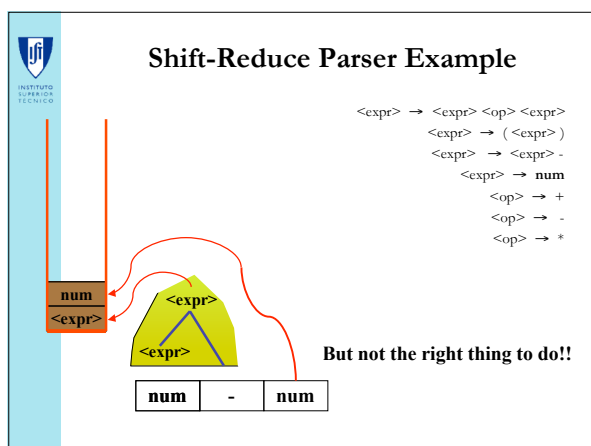
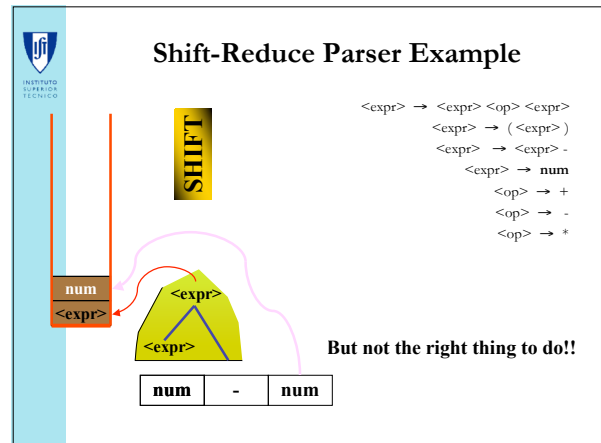
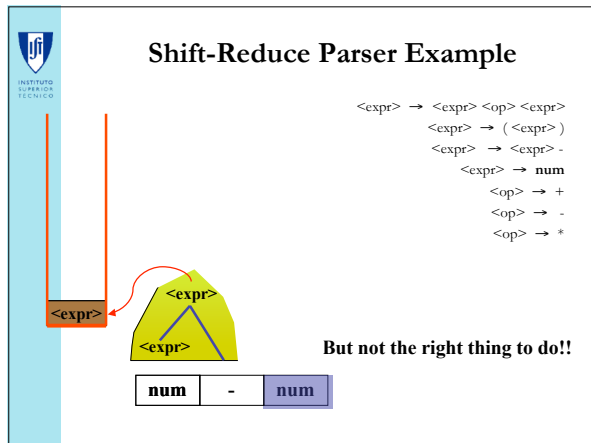
### • Change in the Grammar

```
<expr> → <expr> <op> <expr>
<expr> → ( <expr> )
<expr> → <expr> -
<expr> → num
<op> → +
<op> → -
<op> → *
```

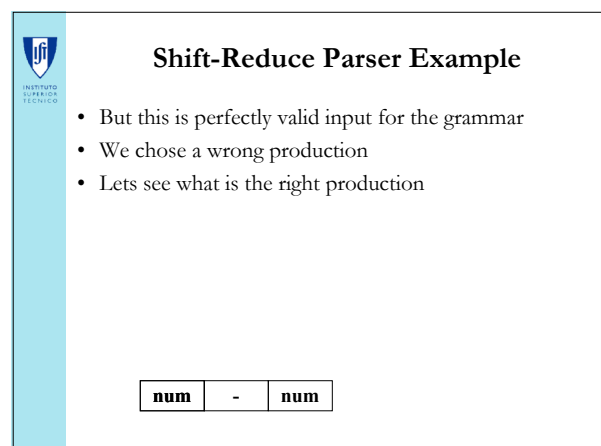
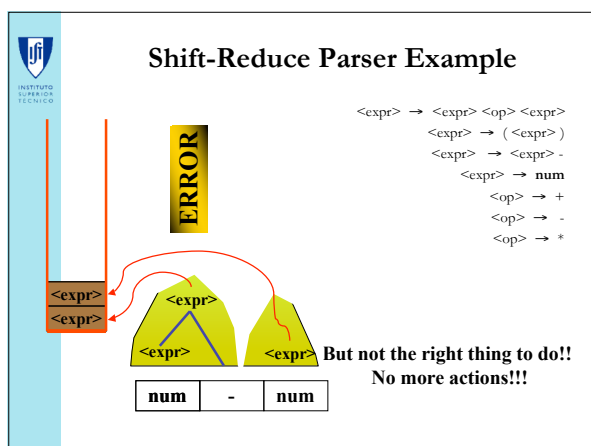
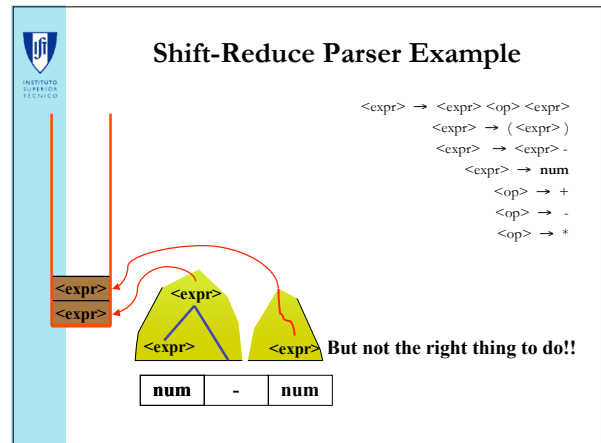
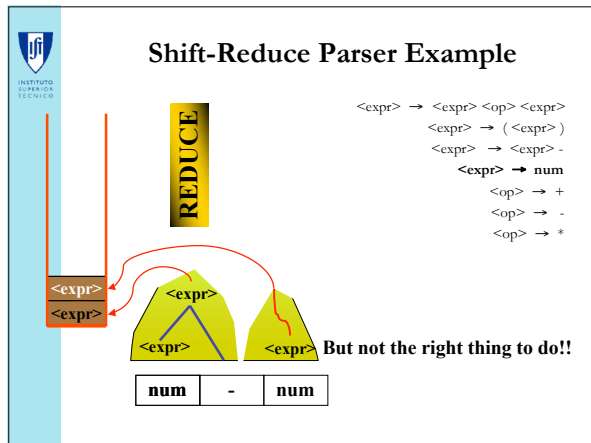


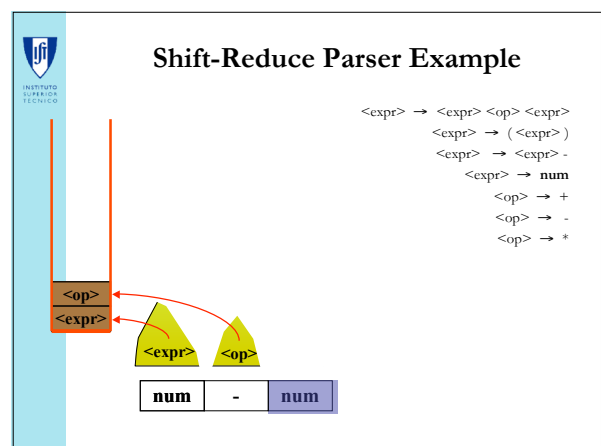
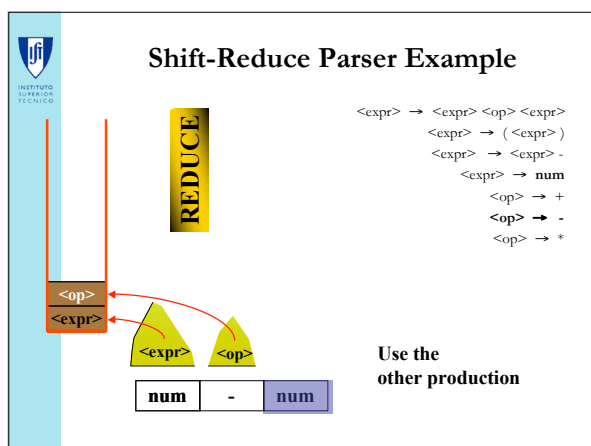
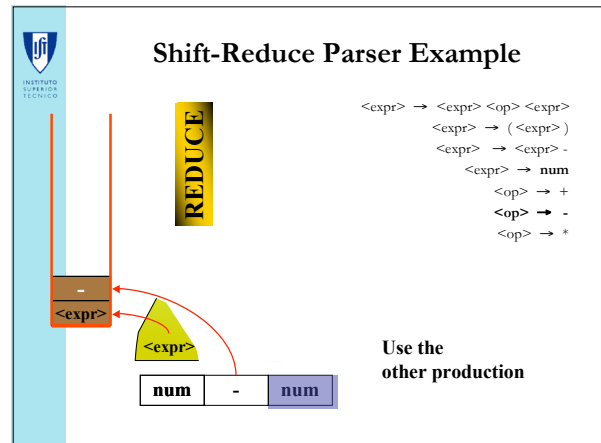
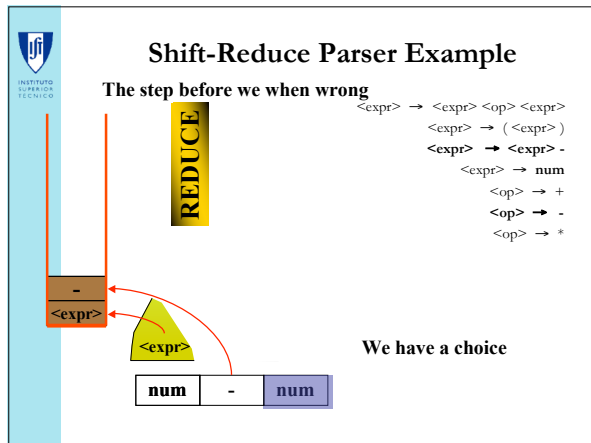


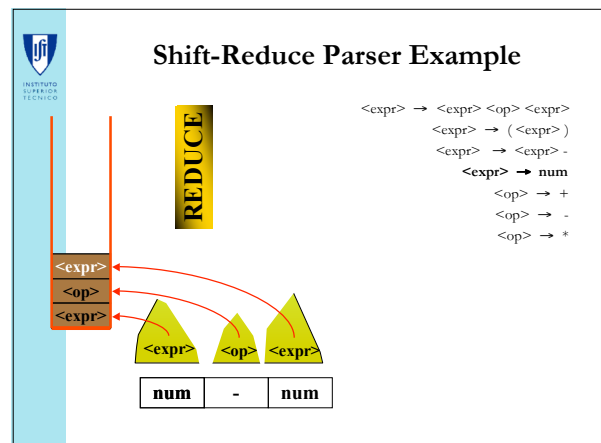
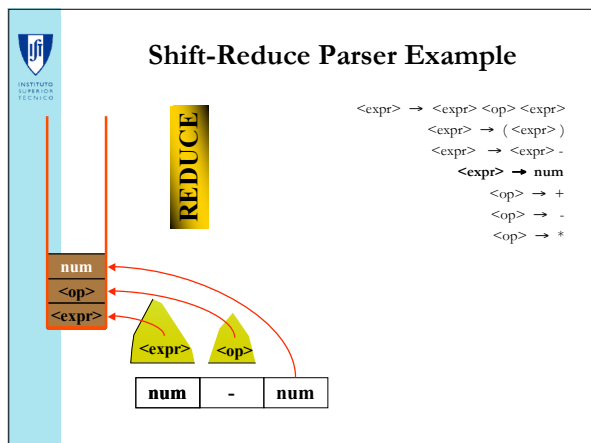
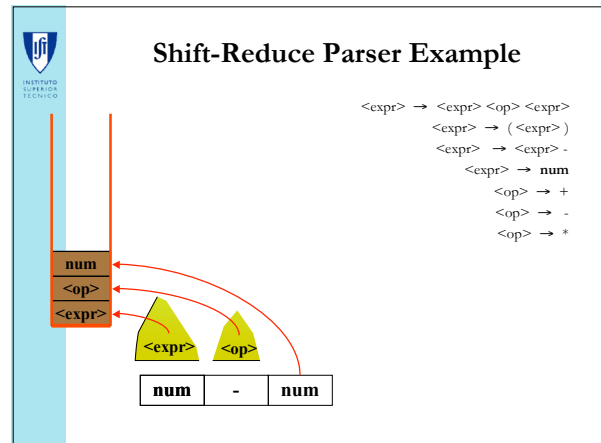
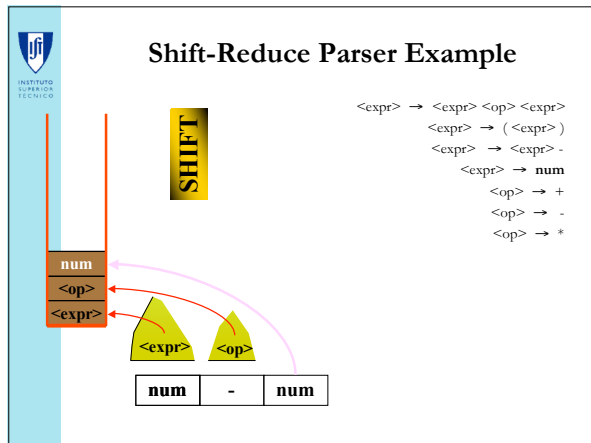


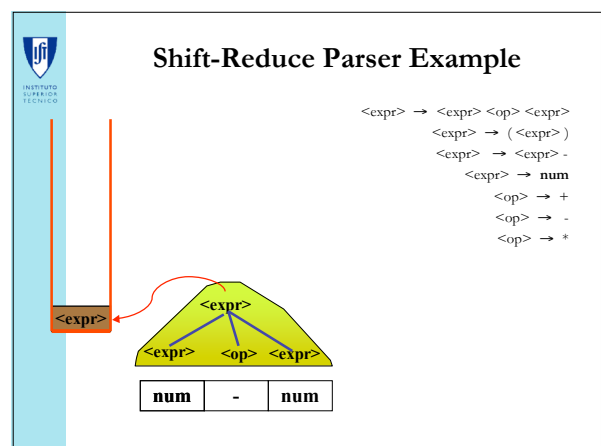
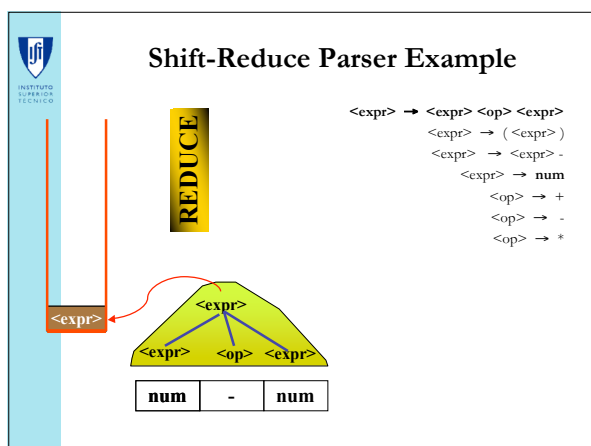
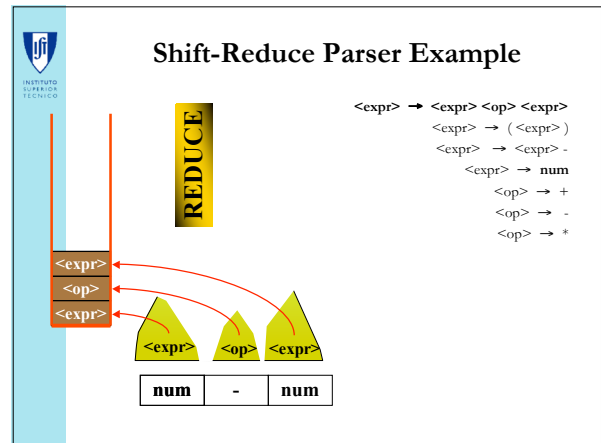
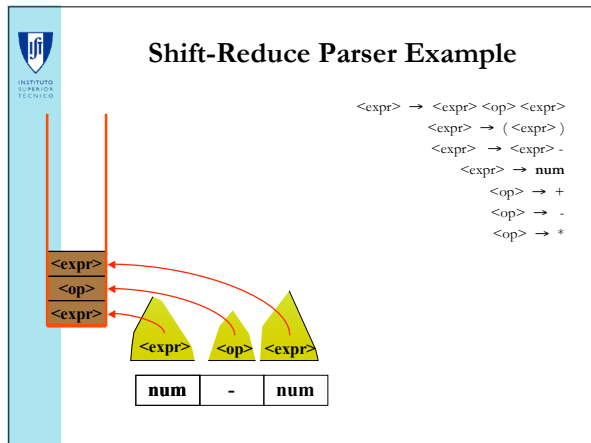


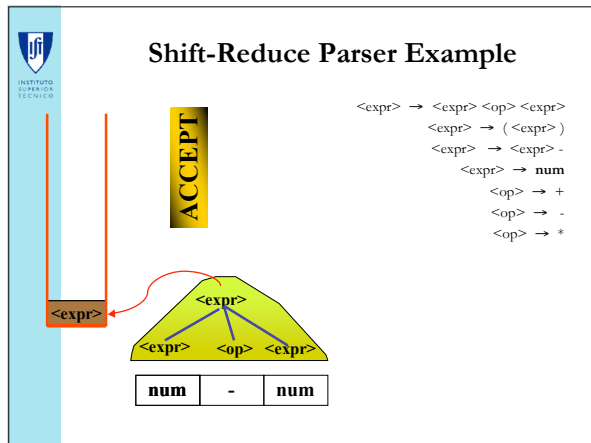












- This is not that simple!**
- Many choices of reductions
    - Matches multiple RHS
  - Choice between shift and reduce
    - Stack matches a RHS
    - But that may not be the right match
    - May need to shift an input and later find a different reduction
  - Keep additional information

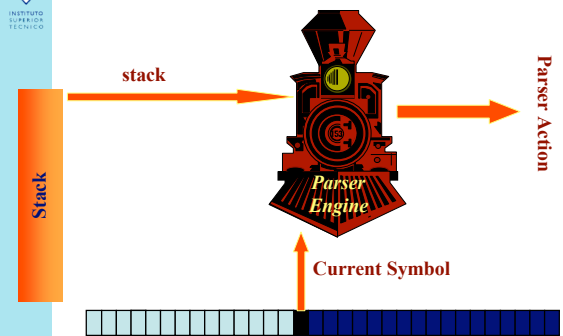
- Outline**
- Implementing a Parser
  - Shift-Reduce Parser Example
  - Why is it hard to build a parser engine?
  - LR(k) parser tables
  - Constructing a LR(0) Parser Engine

- Constructing a LR(k) Parser**
- We will construct few LR(k) parsers
    - LR(0),
    - SLR (or simple LR)
    - LR(1)

## Constructing a LR(k) Parser

- We will construct few LR(k) parsers
  - LR(0),
  - SLR (or simple LR)
  - LR(1)
- We followed the parsing actions
- What is in the parse engine
  - decide between shift and reduce
  - decide on the right reduction

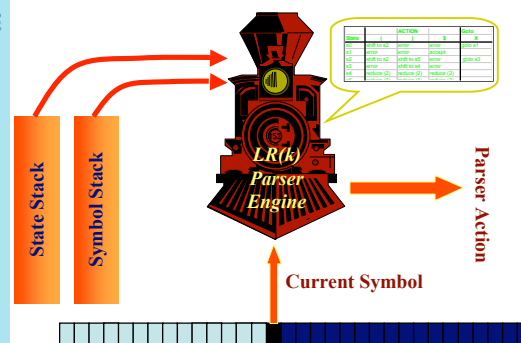
## Actions of a Shift-Reduce Parser



## Constructing a LR(k) Parser

- Create a DFA
  - encodes all the possible states that the parser can be in
  - DFA state transition occurs on terminals and non-terminals
- Create an Parser table
  - stores what action should be taken for the current state and current input character
- Maintain a stack of states
  - in parallel with the stack of symbols

## LR(k) Parser Engine



## Parser Tables

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

- Look-up [top of state stack] [input symbol] in the parser table
- Carry-out the described action

## Parser Tables

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

- Shift to  $s\#$ 
  - Push input token into the symbol stack
  - Push  $s\#$  into state stack
  - Advance to next input symbol

## Parser Tables

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

- Reduce ( $n$ )
  - Pop both stacks as many times as the number of symbols on the RHS of rule  $n$
  - Push LHS of rule  $n$  into symbol stack
  - Lookup [top of the state stack][top of symbol stack]
  - Push that state (in goto  $k$ ) into state stack

## Parser Tables

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

- Accept
  - Stop parsing and report success

## Parser Tables

	ACTION			Goto
State	(	)	\$	X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

- Error
  - Stop parsing and report failure

## LR example

- The grammar
  - $\langle S \rangle \rightarrow \langle X \rangle \$$  (1)
  - $\langle X \rangle \rightarrow ( \langle X \rangle )$  (2)
  - $\langle X \rangle \rightarrow ( )$  (3)

## Question

- The grammar
  - $\langle S \rangle \rightarrow \langle X \rangle \$$  (1)
  - $\langle X \rangle \rightarrow ( \langle X \rangle )$  (2)
  - $\langle X \rangle \rightarrow ( )$  (3)
- What is the language accepted by this CFG?

## Parser Table in Action

- The grammar
  - $\langle S \rangle \rightarrow \langle X \rangle \$$  (1)
  - $\langle X \rangle \rightarrow ( \langle X \rangle )$  (2)
  - $\langle X \rangle \rightarrow ( )$  (3)





## Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

### Action Table

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



## Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

### Action Table

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



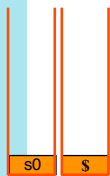
## Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

### Action Table

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



( ( ( ) ) ) \$



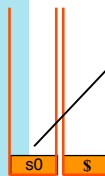
## Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

### Action Table

State	ACTION			Goto
	(	)	\$	
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



( ( ( ) ) ) \$

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	)	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	)	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	)	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	)	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action )	\$	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action )	\$	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action )	\$	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action )	\$	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

Stack: s5, s2, s2, s0

Input: ( ( ( ) ) \$

Current state: s5, Current input: (

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

Stack: s5, s2, s2, s0

Input: ( ( ( ) ) \$

Current state: s5, Current input: (

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

Stack: s5, s2, s2, s0

Input: ( ( ( ) ) \$

Current state: s5, Current input: (

**Parser Table in Action**

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow (\langle X \rangle)$
- (3)  $\langle X \rangle \rightarrow ( )$

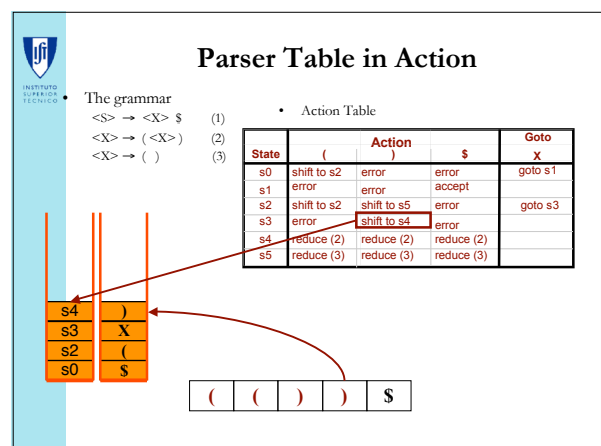
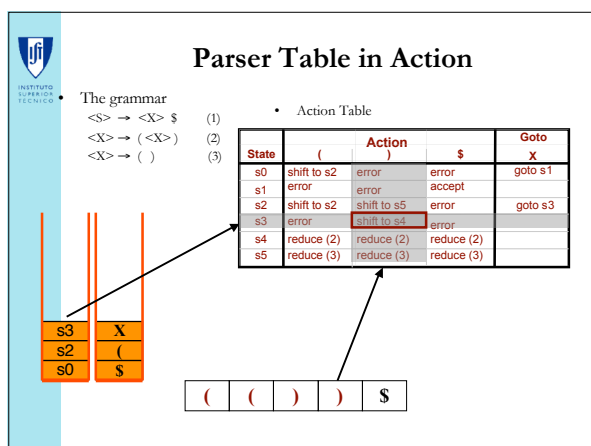
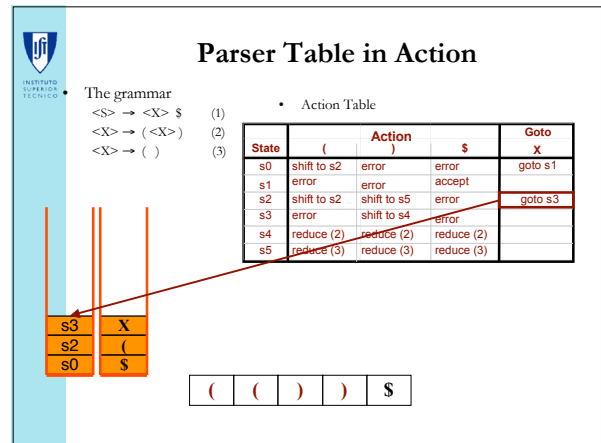
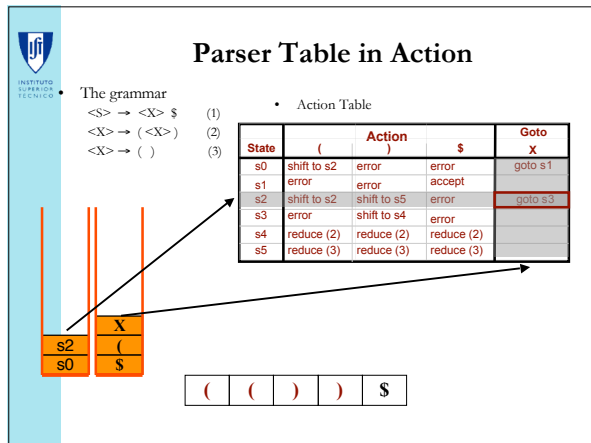
Action Table


State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

Stack: s2, s0

Input: ( ( ( ) ) \$

Current state: s2, Current input: (





INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

s4

)

s3

X


s2

(

s0

\$

( ( ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

s4

)

s3

X


s2

(

s0

\$

( ( ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

s4

)

s3

X


s2

(

s0

\$

( ( ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table


State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

s0

\$

X

( ( ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	


s1

s0

X

\$

( ( ( ) ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	


s1

s0

X

\$

( ( ( ) ) ) \$



INSTITUTO SUPERIOR TÉCNICO

### Parser Table in Action

The grammar

- (1)  $\langle S \rangle \rightarrow \langle X \rangle \$$
- (2)  $\langle X \rangle \rightarrow ( \langle X \rangle )$
- (3)  $\langle X \rangle \rightarrow ( )$

Action Table

State	(	Action	\$	Goto
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	

s1


s0

X

\$

( ( ( ) ) ) \$

Accept



INSTITUTO SUPERIOR TÉCNICO

### Summary

- Parser Implementation
- Shift-Reduce Parsing
- Examples