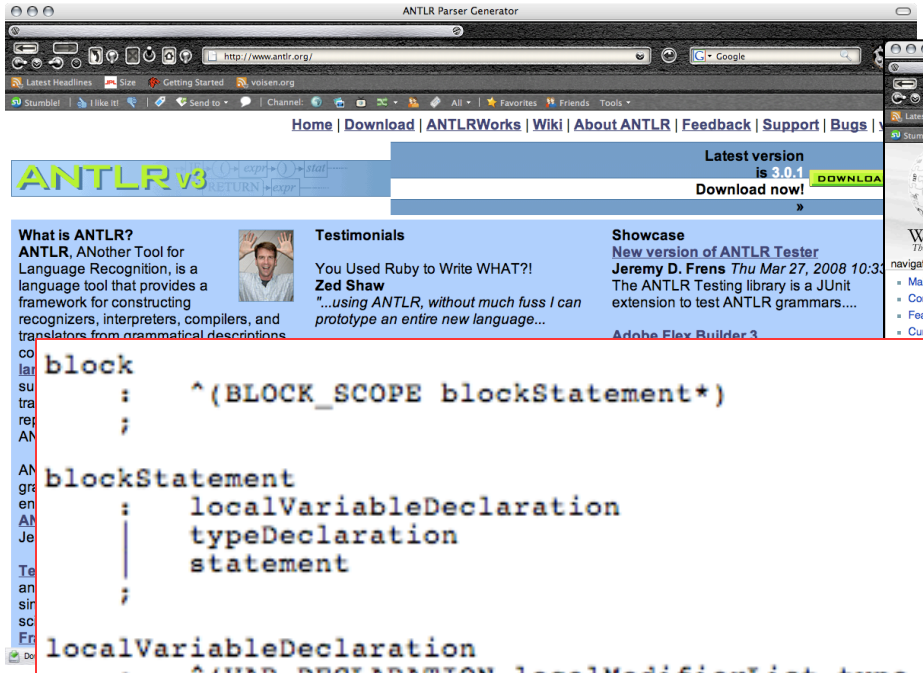
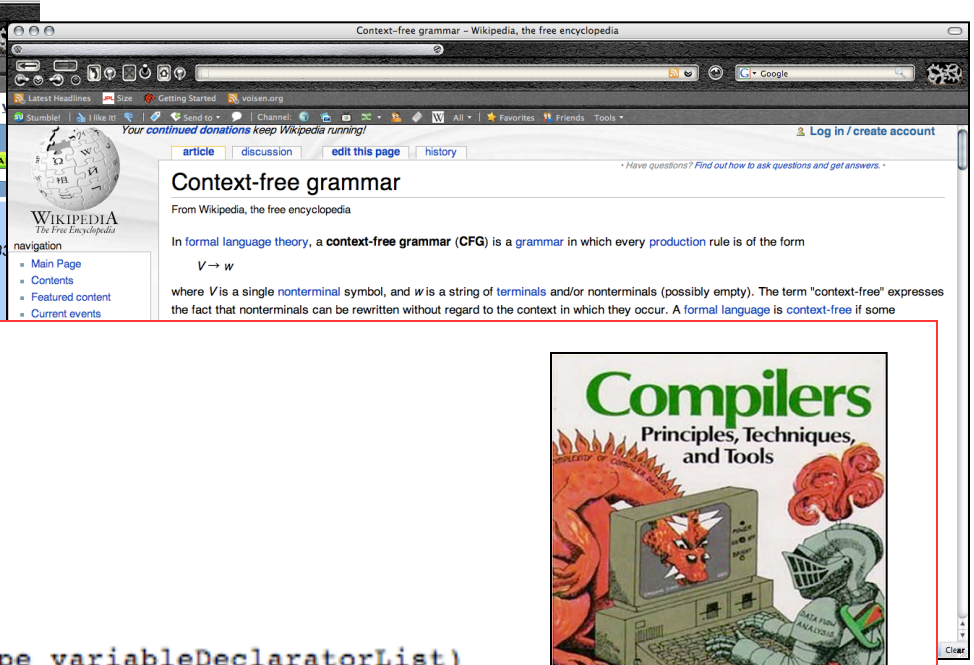
context free grammars

CS 119

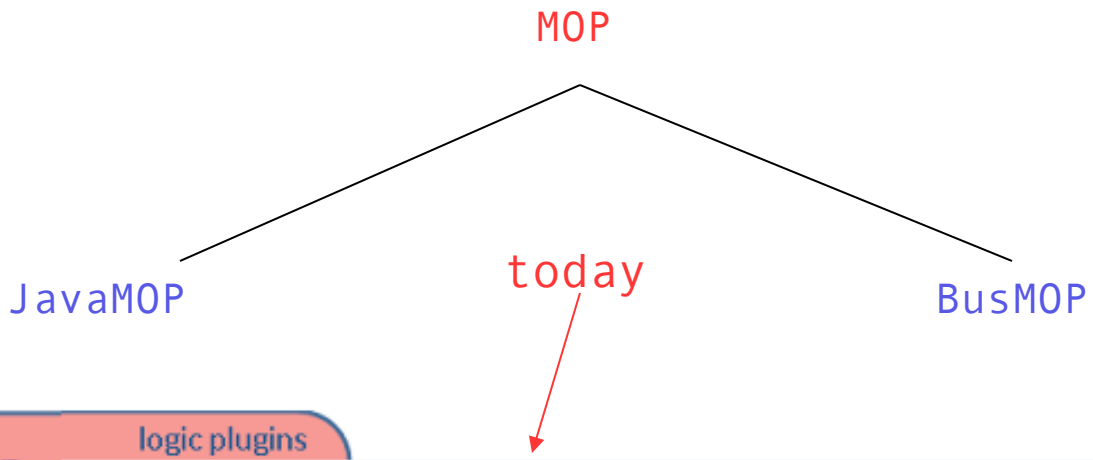
a property can be seen as a
language defined by a grammar






```
block
    :   ^(BLOCK_SCOPE blockStatement*)
    ;

blockStatement
    :   localVariableDeclaration
    |   typeDeclaration
    |   statement
    ;

localVariableDeclaration
    :   ^(VAR_DECLARATION localModifierList type variableDeclaratorList)
    ;


statement
    :   block
    |   ^(ASSERT expression expression?)
    |   ^(IF parenthesizedExpression statement statement?)
    |   ^(FOR forInit forCondition forUpdater statement)
    |   ^(FOR_EACH localModifierList type IDENT expression statement)
    |   ^(WHILE parenthesizedExpression statement)
    |   ^(DO statement parenthesizedExpression)
    |   ^(TRY block catches? block?)  // The second optional block is the optional finally block.
    |   ^(SWITCH parenthesizedExpression switchBlockLabels)
    |   ^(SYNCHRONIZED parenthesizedExpression block)
    |   ^(RETURN expression?)
    |   ^(THROW expression)
    |   ^(BREAK IDENT?)
    |   ^(CONTINUE IDENT?)
    |   ^(LABELED_STATEMENT IDENT statement)
    |   expression
    |   SEMI // Empty statement.
    ;
```

instances of MOP



MOP	logic plugins: FSM	ERE	CFG	PTLTL	FTLTL	PTCaRet	...
languages: JavaMOP	JavaFSM	JavaERE	JavaCFG	JavaPTLTL	JavaFTLTL	JavaPTCaRet	...
BusMOP	BusFSM	BusERE	...	BusPTLTL	...	...	...
...	...	...	...	...	...	...	...

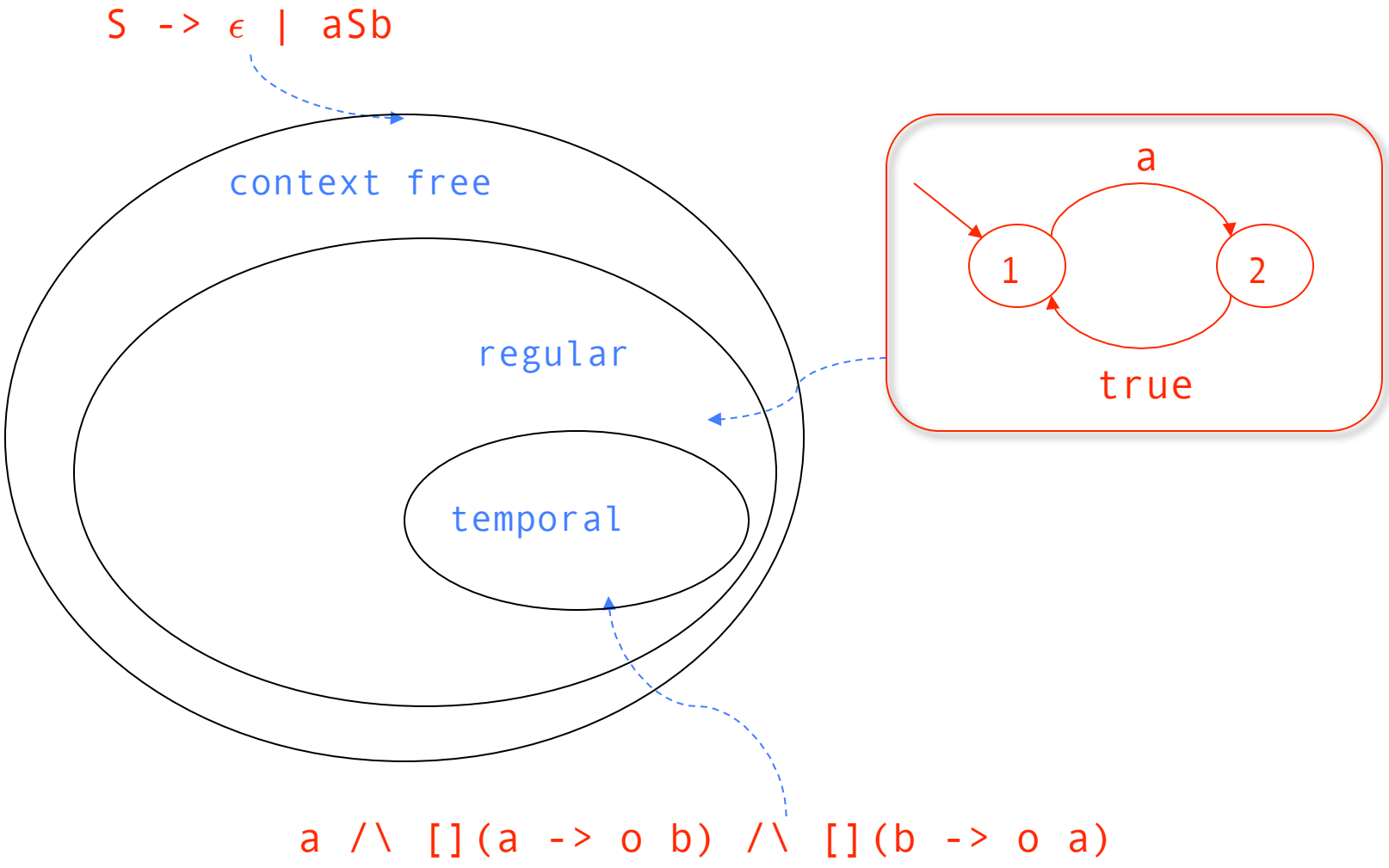
MOP Matrix: a clickable map of MOP pages.

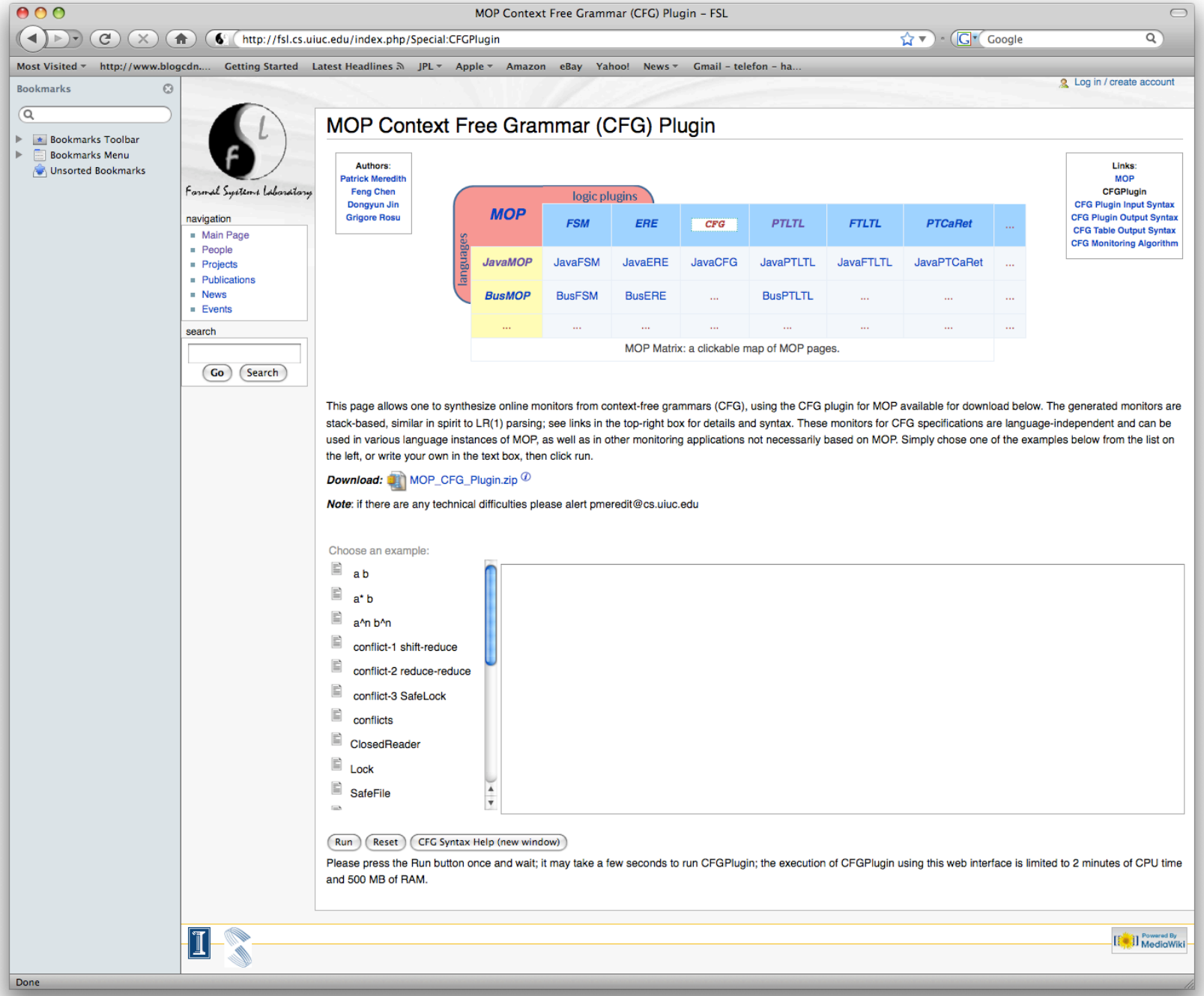
Chomsky's language hierarchy

Grammar	Languages	Automaton	Production rules (constraints)
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$

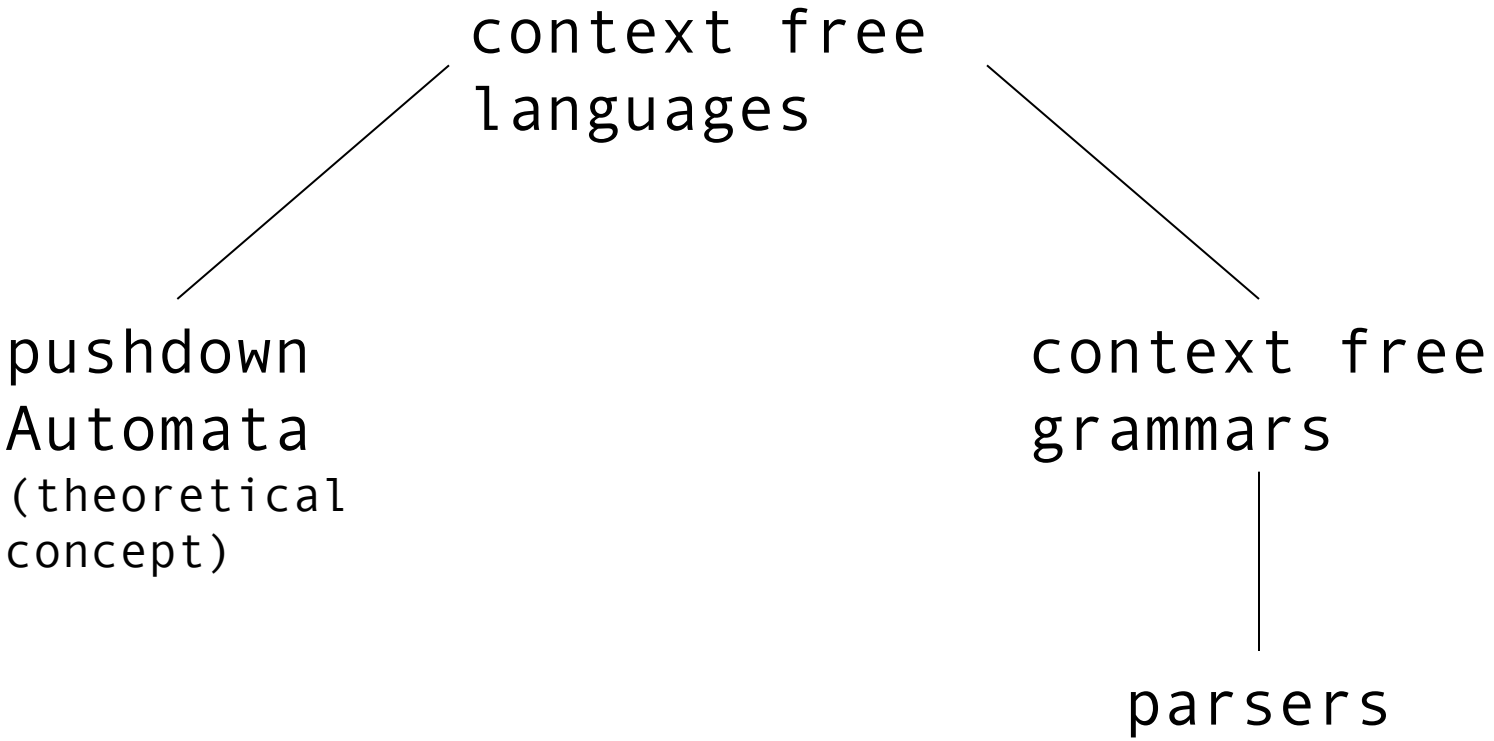
http://en.wikipedia.org/wiki/Chomsky_hierarchy

the language hierarchy



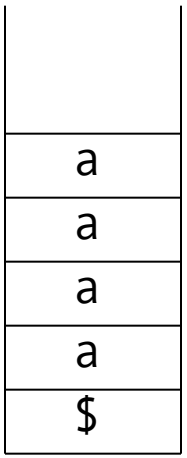

MOP Context Free Grammar (CFG) Plugin – FSL
http://fsl.cs.uiuc.edu/index.php/Special:CFGPlugin
Google
Most Visited
http://www.blogcdn....
Getting Started
Latest Headlines
JPL
Apple
Amazon
eBay
Yahoo!
News
Gmail – telefon – ha...
Bookmarks
Bookmarks Toolbar
Bookmarks Menu
Unsorted Bookmarks
Log in / create account
Formal System Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
MOP Context Free Grammar (CFG) Plugin
Authors:
Patrick Meredith
Feng Chen
Dongyun Jin
Grigore Rosu
logic plugins
MOP
languages
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
...
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
Links:
MOP
CFGPlugin
CFG Plugin Input Syntax
CFG Plugin Output Syntax
CFG Table Output Syntax
CFG Monitoring Algorithm
This page allows one to synthesize online monitors from context-free grammars (CFG), using the CFG plugin for MOP available for download below. The generated monitors are stack-based, similar in spirit to LR(1) parsing; see links in the top-right box for details and syntax. These monitors for CFG specifications are language-independent and can be used in various language instances of MOP, as well as in other monitoring applications not necessarily based on MOP. Simply chose one of the examples below from the list on the left, or write your own in the text box, then click run.
Download: MOP_CFG_Plugin.zip
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
a b
a* b
a^n b^n
conflict-1 shift-reduce
conflict-2 reduce-reduce
conflict-3 SafeLock
conflicts
ClosedReader
Lock
SafeFile
Run
Reset
CFG Syntax Help (new window)
Please press the Run button once and wait; it may take a few seconds to run CFGPlugin; the execution of CFGPlugin using this web interface is limited to 2 minutes of CPU time and 500 MB of RAM.
Powered By MediaWiki
Done

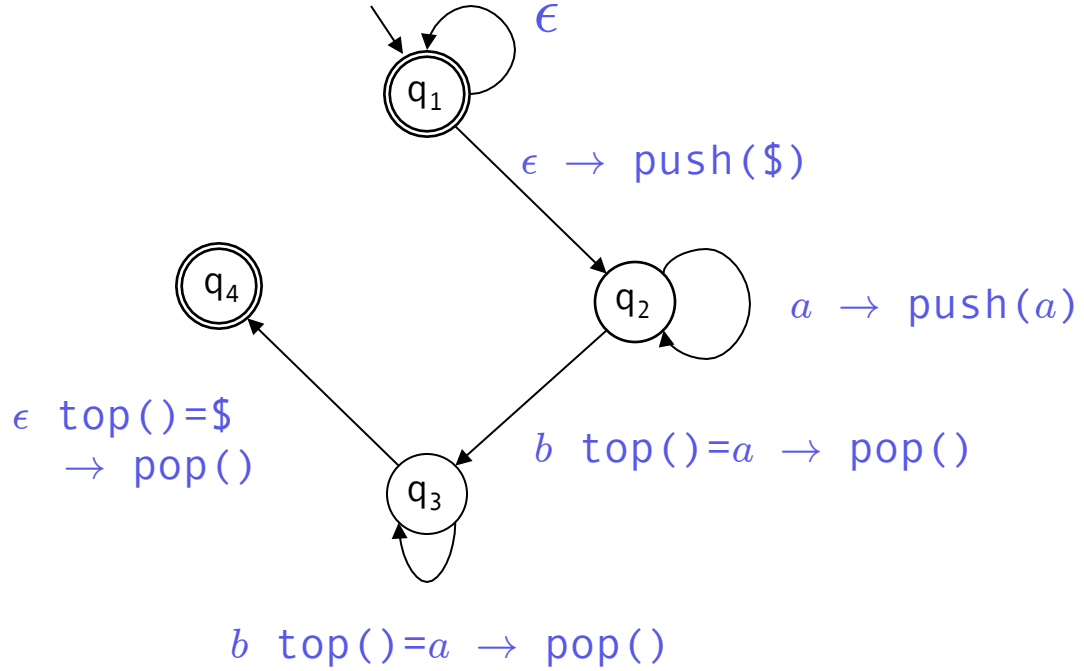
the 4 elements



context free languages and pushdown automata

$a^n\ b^n$ $S \rightarrow \epsilon \mid aSb$ input: aaaa↑bbbb





CFL parsers

Consider the term: ax

example grammar:

$$S \rightarrow Ax$$
$$A \rightarrow a$$
$$A \rightarrow b$$

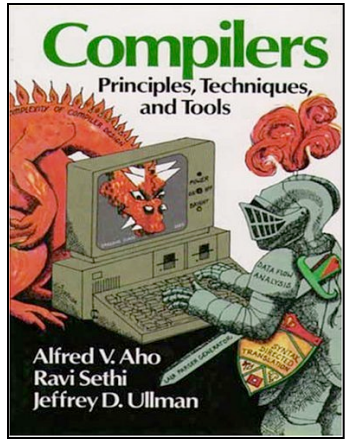
- top-down parser
  - expands non-terminals into RH-sides

$$S \rightarrow Ax \rightarrow ax$$

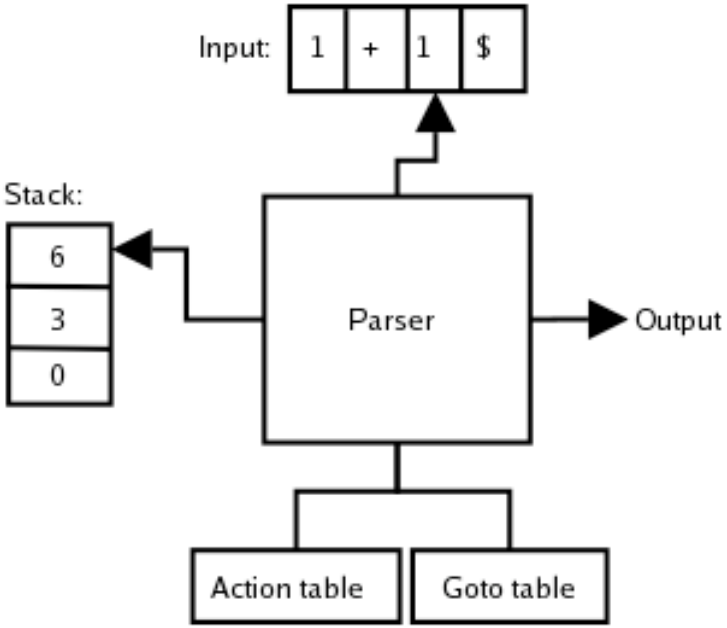
- bottom-up parser
  - reduces RH-sides to non-terminals
  - LALR(1) $\subset$ LR(1) $\subset$ DCFL $\subset$ CFL

JavaMOP

$$ax \rightarrow Ax \rightarrow S$$



structure of a table-driven bottom-up parser



- an input buffer
- a stack of states visited
- an action-table giving a grammar rule to apply given the current state and current terminal in input buffer
- a goto-table describing which new state it should go to

actions

- *Shift* - push token onto stack
- *Reduce* - remove handle from stack and push on corresponding nonterminal
- *Accept* - recognize sentence when stack contains only the distinguished symbol and input is empty
- *Error* - happens when none of the above is possible; means original input was not a sentence!

abstract algorithm

- start with an empty stack
- a "shift" action corresponds to pushing the current input symbol onto the stack
- a "reduce" action occurs when we have a handle on top of the stack. To perform the reduction, we pop the handle off the stack and replace it with the non-terminal on the LHS of the corresponding rule.

the lock/release example

specification

S → lock S release
S → epsilon

()	lock lock lock release release release	SHIFT
(lock)	lock lock release release release	SHIFT
(lock lock)	lock release release release	SHIFT
(lock lock lock)	release release release	SHIFT
(lock lock lock release)	release release	RED(S)
(lock lock S)	release release	SHIFT
(lock lock S release)	release	RED(S)
(lock S)	release	SHIFT
(lock S release)		RED(S)
(S)		ACC

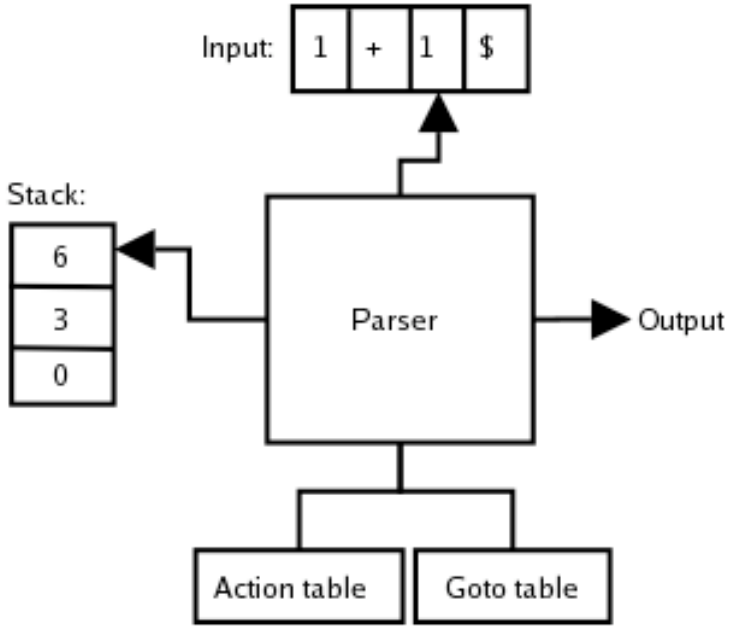
expressions example

Given the language:

```
<Expression> --> <Term> | <Term> + <Expression>
<Term>       --> <Factor> | <Factor> * <Term>
<Factor>     --> [ <Expression> ] | 0...9
```

()	(2 * [1 + 3])	SHIFT
(2)	(* [1 + 3])	REDUCE
(<Factor>)	(* [1 + 3])	SHIFT
(<Factor> *)	([1 + 3])	SHIFT
(<Factor> * [)	(1 + 3])	SHIFT
(<Factor> * [1)	(+ 3])	REDUCE (twice)
(<Factor> * [<Term>)	(+ 3])	SHIFT (twice)
(<Factor> * [<Term> + 3) ( ])		REDUCE (thrice)
(<Factor> * [<Term> + <Expression>) ( ])		REDUCE
(<Factor> * [<Expression>) ( ])		SHIFT
(<Factor> * [<Expression>]) ()		REDUCE
(<Factor> * <Factor>)	()	REDUCE
(<Factor> * <Term>)	()	REDUCE
(<Term>)	()	REDUCE
(<Expression>)	()	SUCCESS

recall structure of a table-driven bottom-up parser



JavaMOP specification

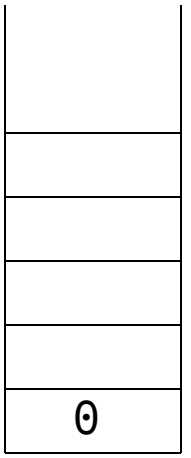
event lock
event release

productions :
  S -> lock S release | epsilon

input

lock
lock
lock
release
release
release
$

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



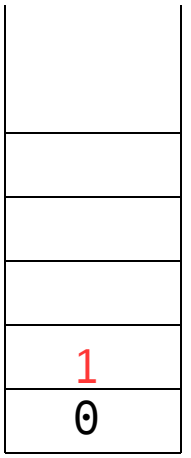
JavaMOP specification

```
event lock
event release

productions :
  S -> lock S release | epsilon
```

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

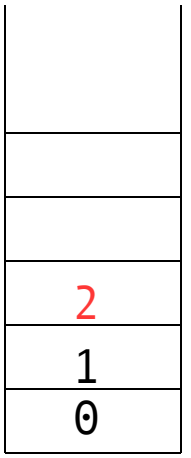
event lock
event release

productions :
  S -> lock S release | epsilon

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



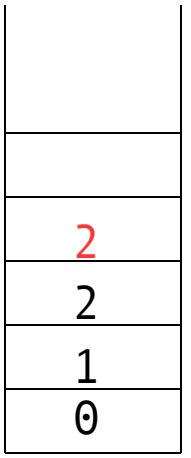
JavaMOP specification

```
event lock
event release

productions :
  S -> lock S release | epsilon
```

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

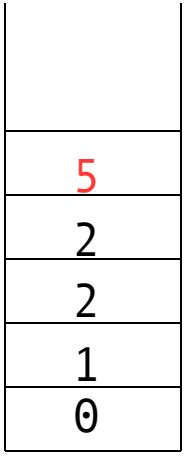
event lock
event release

productions :
  S -> lock S release | epsilon

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

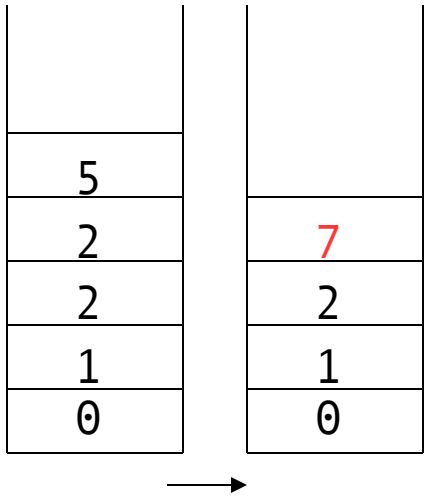
event lock
event release

productions :
  S -> lock S release | epsilon

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



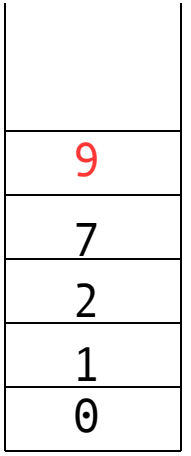
JavaMOP specification

```
event lock
event release

productions :
  S -> lock S release | epsilon
```

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

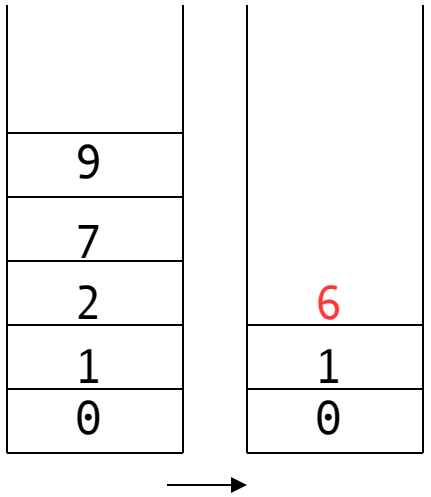
event lock
event release

productions :
  S -> lock S release | epsilon

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



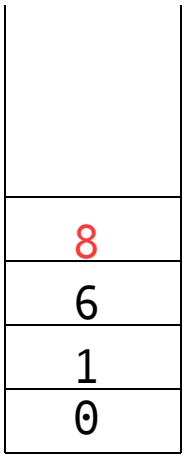
JavaMOP specification

```
event lock
event release

productions :
  S -> lock S release | epsilon
```

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

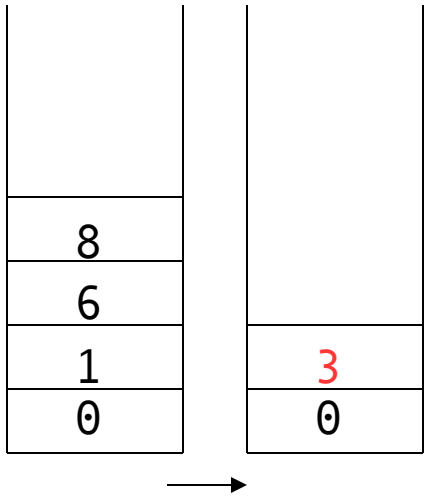
event lock
event release

productions :
  S -> lock S release | epsilon

input

```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP specification

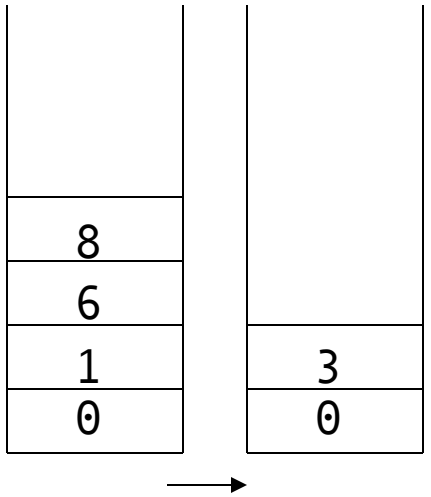
event lock
event release

productions :
  S -> lock S release | epsilon

input

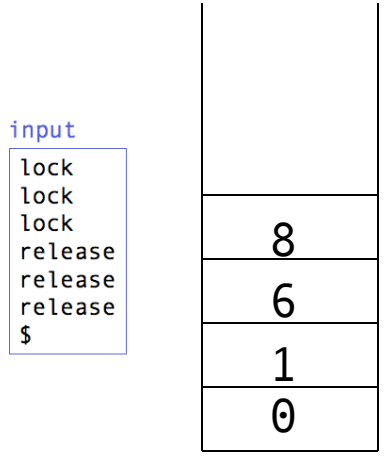
```
lock
lock
lock
release
release
release
$
```

	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error



JavaMOP algorithm

```
globals action_table, goto_table
initialize stack.push(initial_state)
procedure monitor(event, stack)
  locals state, state', stack', A
  while (true) {
    state <- stack.top()
    switch (action_table[state,event].action_type) {
    case shift :
      state' <- action_table[state, event].next_state
      if (state' = error) {
        violation
        break while
      }
      stack.push(state')
      if (action_table[state', $].action_type = reduce) {
        stack' <- stack.clone()
        monitor($, stack')
      }
      break while
    case reduce :
      stack.pop(action_table[state, event].pop)
      A <- action_table[state, event].non_terminal
      state' <- stack.top()
      stack.push(goto_table[state', A])
      continue while
    case accept :
      validation
      break while
  }
}
```



	Action			Goto	
	$	lock	release		S
0	SHIFT(Error)	SHIFT(1)	SHIFT(Error)	0	3
1	SHIFT(Error)	SHIFT(2)	SHIFT(4)	1	6
2	SHIFT(Error)	SHIFT(2)	SHIFT(5)	2	7
3	ACCEPT	SHIFT(Error)	SHIFT(Error)	3	Error
4	REDUCE(S, 2)	SHIFT(Error)	SHIFT(Error)	4	Error
5	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 2)	5	Error
6	SHIFT(Error)	SHIFT(Error)	SHIFT(8)	6	Error
7	SHIFT(Error)	SHIFT(Error)	SHIFT(9)	7	Error
8	REDUCE(S, 3)	SHIFT(Error)	SHIFT(Error)	8	Error
9	SHIFT(Error)	SHIFT(Error)	REDUCE(S, 3)	9	Error

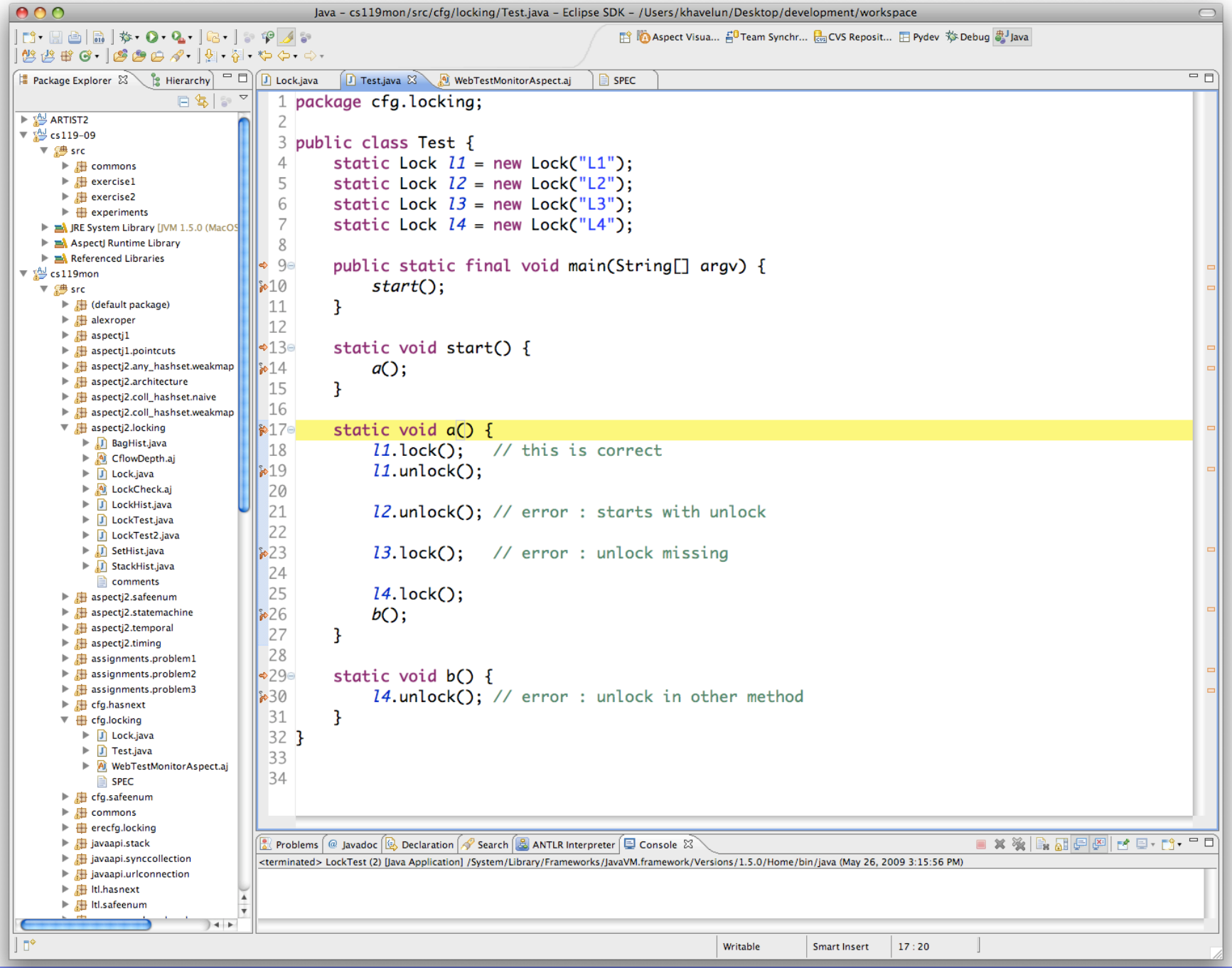
Within one method invocation, locks should be acquired and released correctly (all taken locks should be released as many times as taken)

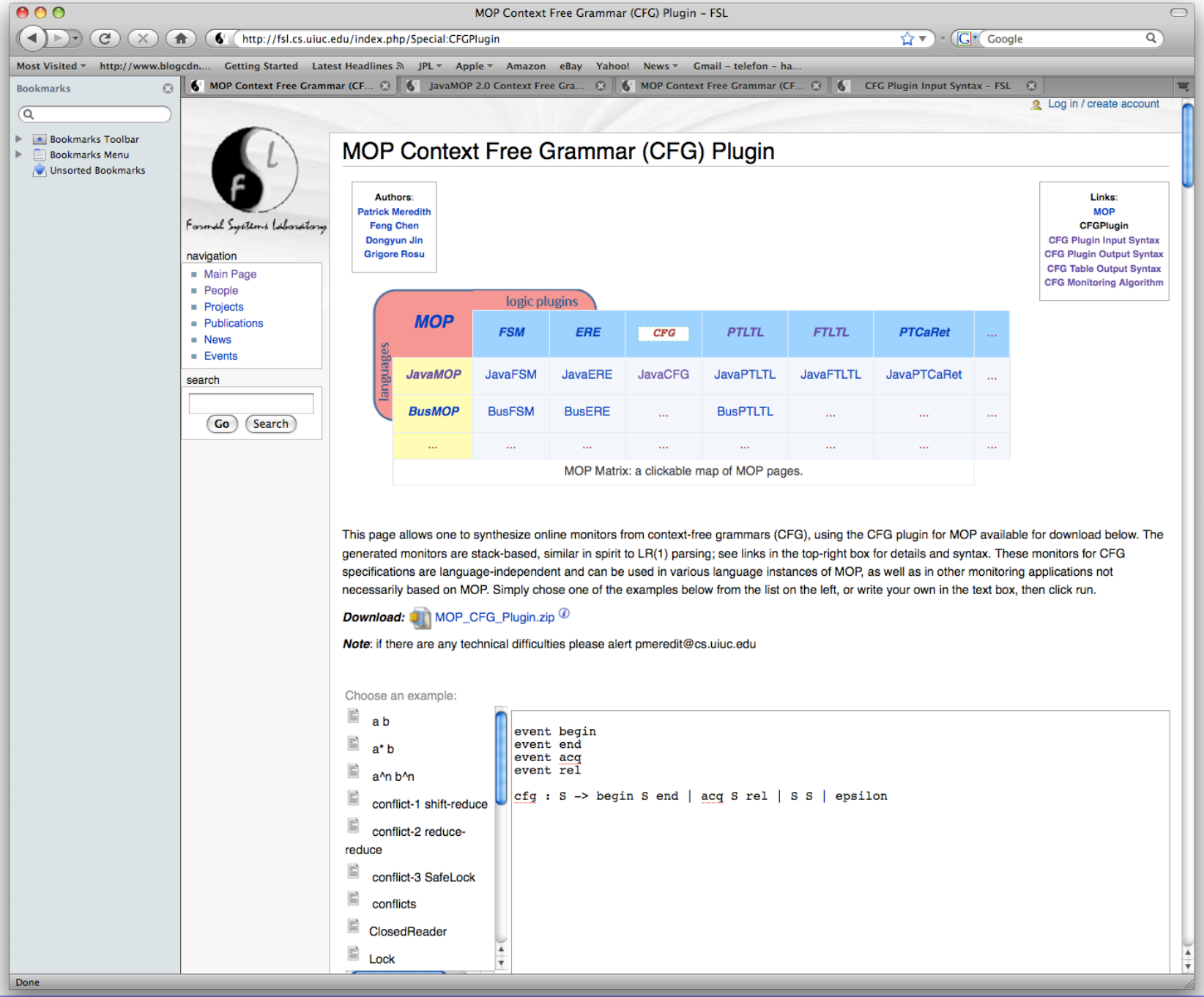
```
public class Lock {

    synchronized void lock(){
        // ...
    }

    synchronized void unlock(){
        // ...
    }

}
```

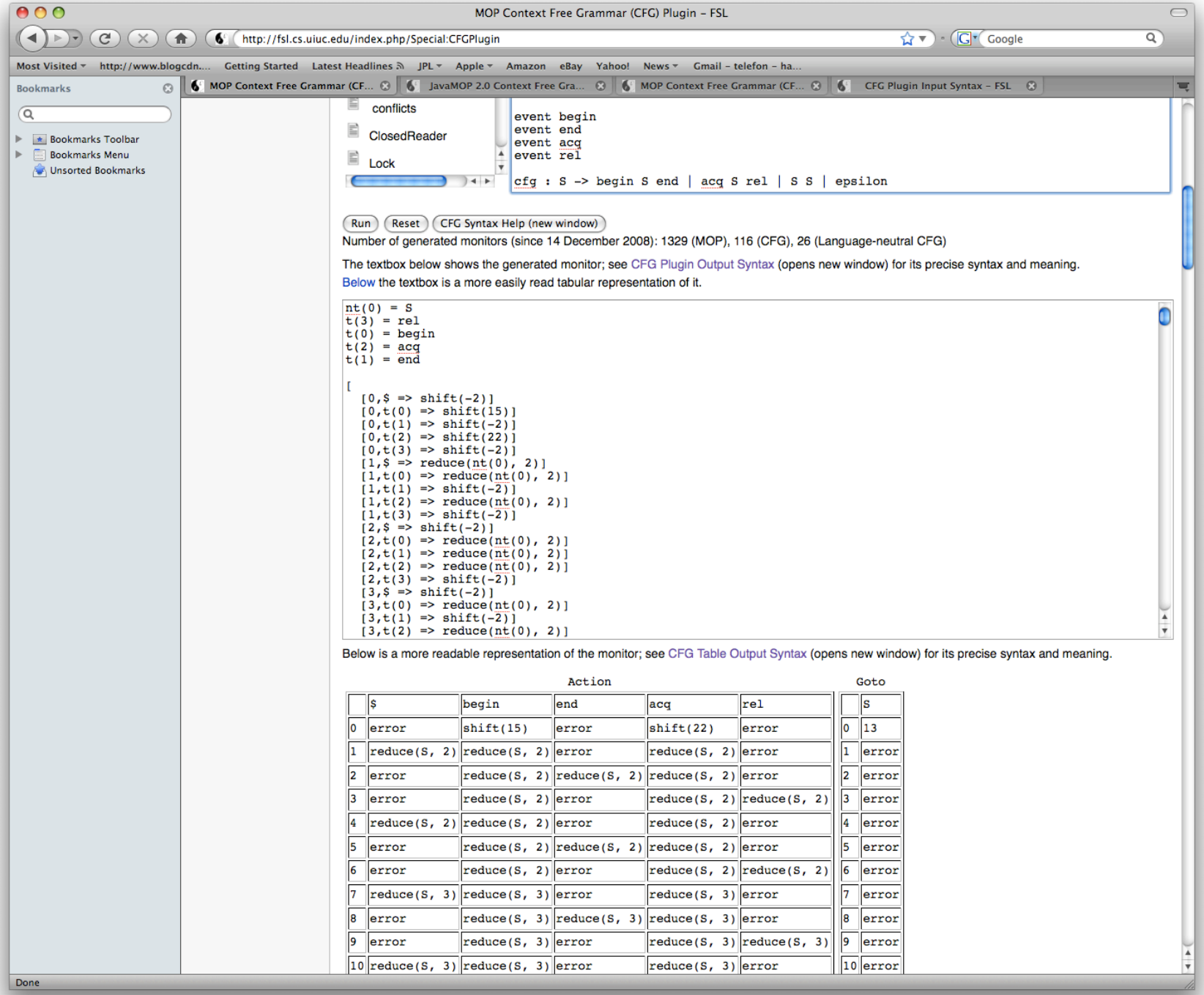
```
static void start() {
    a();
}

static void a() {
    l1.lock();   // this is correct
    l1.unlock();

    l2.unlock(); // error : starts with unlock

    l3.lock();   // error : unlock missing

    l4.lock();
    b();
}

static void b() {
    l4.unlock(); // error : unlock in other method
}
```

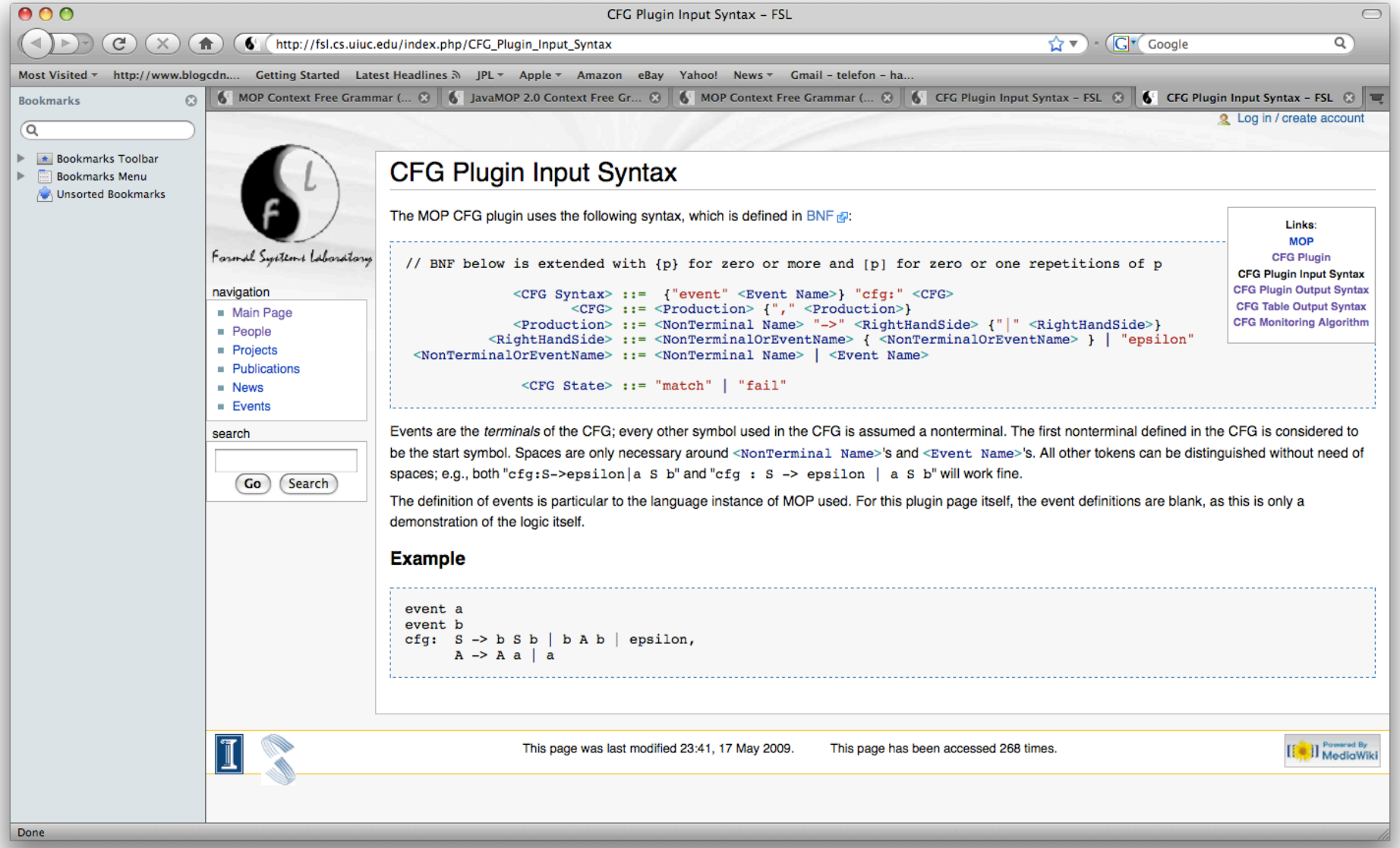
```
package cfg.locking;

import java.util.*;

SafeLock(Lock l, Thread t) {

  event acq after(Lock l, Thread t) :
    call(* Lock.lock()) && target(l) && thread(t) {}

  event rel after(Lock+ l, Thread t) :
    call(* Lock.unlock()) && target(l) && thread(t) {}

  event begin before(Thread t) :
    execution(* *.*(..)) && thread(t) {}

  event end after(Thread t) :
    execution(* *.*(..)) && thread(t) {}

  cfg : S -> begin S end | acq S rel | S S | epsilon


  @fail {
    System.out.println("Unmatched acquire/release at line number " + __LOC);
  }
}
```

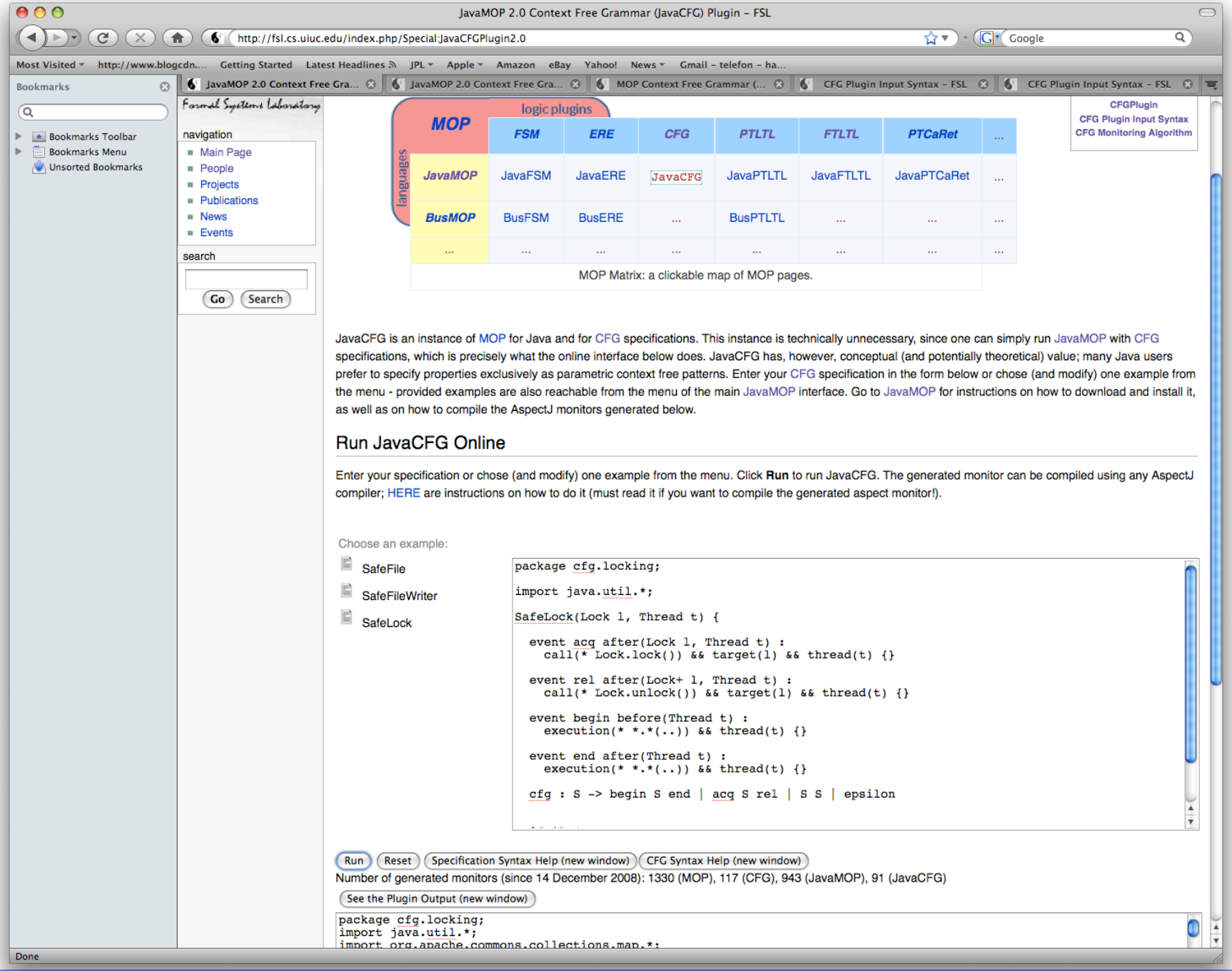
DEMO ON SLIDES


Java – cs119mon/src/cfg/locking/Test.java – Eclipse SDK – /Users/khavelun/Desktop/development/workspace
Package Explorer
Hierarchy
Lock.java
Test.java
WebTestMonitorAspect.aj
SPEC
ARTIST2
cs119-09
src
commons
exercise1
exercise2
experiments
JRE System Library [JVM 1.5.0 (MacOS
AspectJ Runtime Library
Referenced Libraries
cs119mon
src
(default package)
alexroper
aspectj1
aspectj1.pointcuts
aspectj2.any_hashset.weakmap
aspectj2.architecture
aspectj2.coll_hashset.naive
aspectj2.coll_hashset.weakmap
aspectj2.locking
BagHist.java
CflowDepth.aj
Lock.java
LockCheck.aj
LockHist.java
LockTest.java
LockTest2.java
SetHist.java
StackHist.java
comments
aspectj2.safeenum
aspectj2.statemachine
aspectj2.temporal
aspectj2.timing
assignments.problem1
assignments.problem2
assignments.problem3
cfg.hasnext
cfg.locking
Lock.java
Test.java
WebTestMonitorAspect.aj
SPEC
cfg.safeenum
commons
erecfg.locking
javaapi.stack
javaapi.synccollection
javaapi.urlconnection
ltl.hasnext
ltl.safeenum
1 package cfg.locking;
2
3 public class Test {
4 static Lock l1 = new Lock("L1");
5 static Lock l2 = new Lock("L2");
6 static Lock l3 = new Lock("L3");
7 static Lock l4 = new Lock("L4");
8
9 public static final void main(String[] argv) {
10 start();
11 }
12
13 static void start() {
14 a();
15 }
16
17 static void a() {
18 l1.lock(); // this is correct
19 l1.unlock();
20
21 l2.unlock(); // error : starts with unlock
22
23 l3.lock(); // error : unlock missing
24
25 l4.lock();
26 b();
27 }
28
29 static void b() {
30 l4.unlock(); // error : unlock in other method
31 }
32 }
33
34
Problems
Javadoc
Declaration
Search
ANTLR Interpreter
Console
<terminated> LockTest (2) [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/java (May 26, 2009 3:15:56 PM)
Writable
Smart Insert
17 : 20


MOP Context Free Grammar (CFG) Plugin – FSL
http://fsl.cs.uiuc.edu/index.php/Special:CFGPlugin
Log in / create account
MOP Context Free Grammar (CFG) Plugin
Authors:
Patrick Meredith
Feng Chen
Dongyun Jin
Grigore Rosu
Links:
MOP
CFGPlugin
CFG Plugin Input Syntax
CFG Plugin Output Syntax
CFG Table Output Syntax
CFG Monitoring Algorithm
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
logic plugins
MOP
languages
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
This page allows one to synthesize online monitors from context-free grammars (CFG), using the CFG plugin for MOP available for download below. The generated monitors are stack-based, similar in spirit to LR(1) parsing; see links in the top-right box for details and syntax. These monitors for CFG specifications are language-independent and can be used in various language instances of MOP, as well as in other monitoring applications not necessarily based on MOP. Simply chose one of the examples below from the list on the left, or write your own in the text box, then click run.
Download: MOP_CFG_Plugin.zip
Note: if there are any technical difficulties please alert pmeredit@cs.uiuc.edu
Choose an example:
a b
a* b
a^n b^n
conflict-1 shift-reduce
conflict-2 reduce-reduce
conflict-3 SafeLock
conflicts
ClosedReader
Lock
event begin
event end
event acq
event rel
cfg : S -> begin S end | acq S rel | S S | epsilon
Done

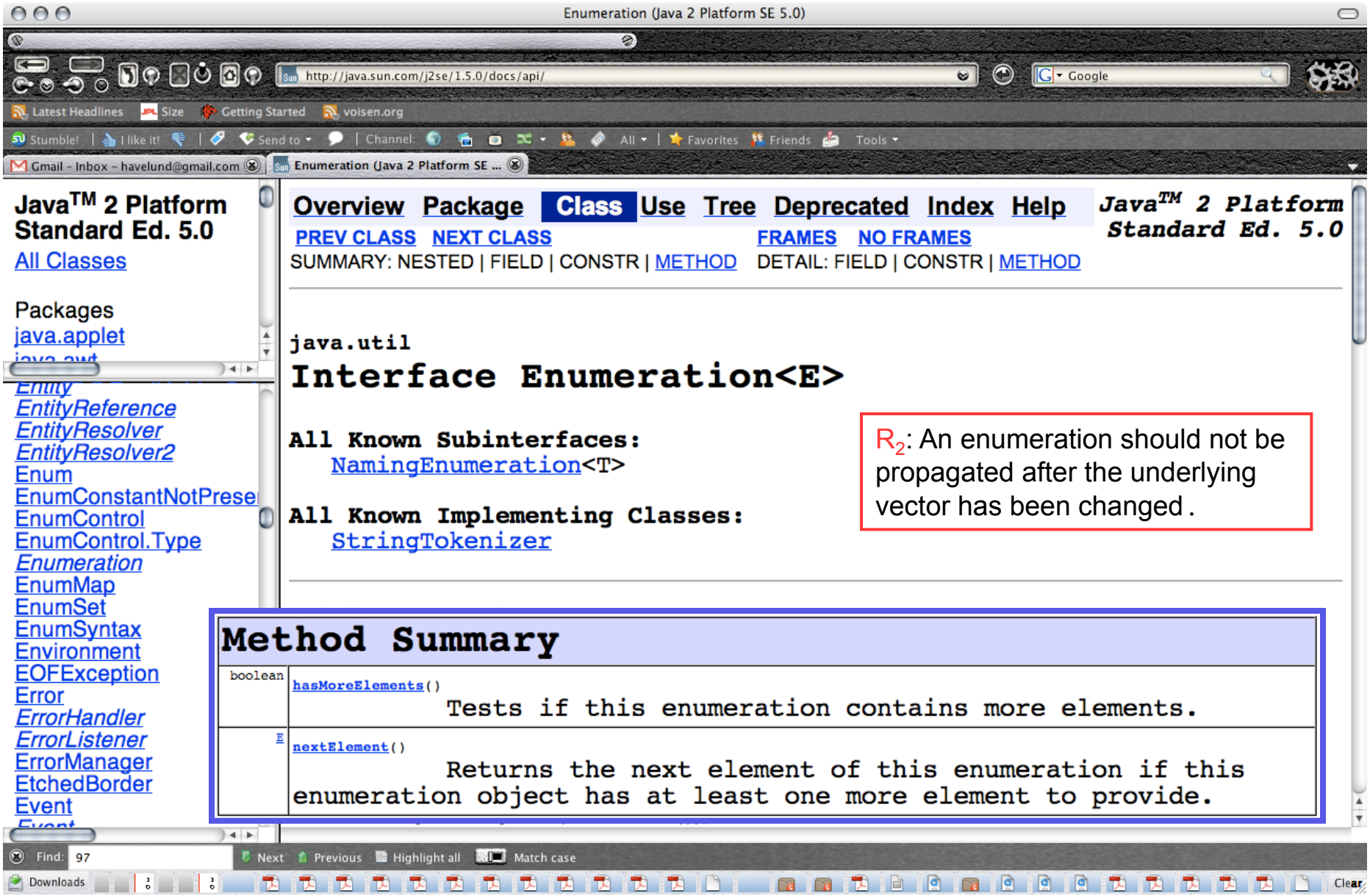

MOP Context Free Grammar (CFG) Plugin - FSL
http://fsl.cs.uiuc.edu/index.php/Special:CFGPlugin
Google
Most Visited
http://www.blogcdn....
Getting Started
Latest Headlines
JPL
Apple
Amazon
eBay
Yahoo!
News
Gmail - telefon - ha...
Bookmarks
Bookmarks Toolbar
Bookmarks Menu
Unsorted Bookmarks
MOP Context Free Grammar (CF...
JavaMOP 2.0 Context Free Gra...
MOP Context Free Grammar (CF...
CFG Plugin Input Syntax - FSL
conflicts
ClosedReader
Lock
event begin
event end
event acq
event rel
cfg : S -> begin S end | acq S rel | S S | epsilon
Run
Reset
CFG Syntax Help (new window)
Number of generated monitors (since 14 December 2008): 1329 (MOP), 116 (CFG), 26 (Language-neutral CFG)
The textbox below shows the generated monitor; see CFG Plugin Output Syntax (opens new window) for its precise syntax and meaning.
Below the textbox is a more easily read tabular representation of it.
nt(0) = S
t(3) = rel
t(0) = begin
t(2) = acq
t(1) = end
[
[0,$ => shift(-2)]
[0,t(0) => shift(15)]
[0,t(1) => shift(-2)]
[0,t(2) => shift(22)]
[0,t(3) => shift(-2)]
[1,$ => reduce(nt(0), 2)]
[1,t(0) => reduce(nt(0), 2)]
[1,t(1) => shift(-2)]
[1,t(2) => reduce(nt(0), 2)]
[1,t(3) => shift(-2)]
[2,$ => shift(-2)]
[2,t(0) => reduce(nt(0), 2)]
[2,t(1) => reduce(nt(0), 2)]
[2,t(2) => reduce(nt(0), 2)]
[2,t(3) => shift(-2)]
[3,$ => shift(-2)]
[3,t(0) => reduce(nt(0), 2)]
[3,t(1) => shift(-2)]
[3,t(2) => reduce(nt(0), 2)]
Below is a more readable representation of the monitor; see CFG Table Output Syntax (opens new window) for its precise syntax and meaning.
Action
Goto
$ begin end acq rel S
0 error shift(15) error shift(22) error 13
1 reduce(S, 2) reduce(S, 2) error reduce(S, 2) error error
2 error reduce(S, 2) reduce(S, 2) reduce(S, 2) error error
3 error reduce(S, 2) error reduce(S, 2) reduce(S, 2) error
4 reduce(S, 2) reduce(S, 2) error reduce(S, 2) error error
5 error reduce(S, 2) reduce(S, 2) reduce(S, 2) error error
6 error reduce(S, 2) error reduce(S, 2) reduce(S, 2) error
7 reduce(S, 3) reduce(S, 3) error reduce(S, 3) error error
8 error reduce(S, 3) reduce(S, 3) reduce(S, 3) error error
9 error reduce(S, 3) error reduce(S, 3) reduce(S, 3) error
10 reduce(S, 3) reduce(S, 3) error reduce(S, 3) error error
Done


CFG Plugin Input Syntax - FSL
http://fsl.cs.uiuc.edu/index.php/CFG_Plugin_Input_Syntax
Google
Most Visited
http://www.blogcdn....
Getting Started
Latest Headlines
JPL
Apple
Amazon
eBay
Yahoo!
News
Gmail - telefon - ha...
Bookmarks
Bookmarks Toolbar
Bookmarks Menu
Unsorted Bookmarks
MOP Context Free Grammar (...
JavaMOP 2.0 Context Free Gr...
MOP Context Free Grammar (...
CFG Plugin Input Syntax - FSL
CFG Plugin Input Syntax - FSL
Log in / create account
Formal System Laboratory
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
CFG Plugin Input Syntax
The MOP CFG plugin uses the following syntax, which is defined in BNF:
// BNF below is extended with {p} for zero or more and [p] for zero or one repetitions of p
<CFG Syntax> ::= {"event" <Event Name>} "cfg:" <CFG>
<CFG> ::= <Production> {"," <Production>}
<Production> ::= <NonTerminal Name> "->" <RightHandSide> {"|" <RightHandSide>}
<RightHandSide> ::= <NonTerminalOrEventName> { <NonTerminalOrEventName> } | "epsilon"
<NonTerminalOrEventName> ::= <NonTerminal Name> | <Event Name>
<CFG State> ::= "match" | "fail"
Links:
MOP
CFG Plugin
CFG Plugin Input Syntax
CFG Plugin Output Syntax
CFG Table Output Syntax
CFG Monitoring Algorithm
Events are the terminals of the CFG; every other symbol used in the CFG is assumed a nonterminal. The first nonterminal defined in the CFG is considered to be the start symbol. Spaces are only necessary around <NonTerminal Name>'s and <Event Name>'s. All other tokens can be distinguished without need of spaces; e.g., both "cfg:S->epsilon|a S b" and "cfg : S -> epsilon | a S b" will work fine.
The definition of events is particular to the language instance of MOP used. For this plugin page itself, the event definitions are blank, as this is only a demonstration of the logic itself.
Example
event a
event b
cfg: S -> b S b | b A b | epsilon,
A -> A a | a
This page was last modified 23:41, 17 May 2009.
This page has been accessed 268 times.
Powered By MediaWiki
Done


JavaMOP 2.0 Context Free Grammar (JavaCFG) Plugin - FSL
http://fsl.cs.uiuc.edu/index.php/Special:JavaCFGPlugin2.0
navigation
Main Page
People
Projects
Publications
News
Events
search
Go
Search
logic plugins
MOP
FSM
ERE
CFG
PTLTL
FTLTL
PTCaRet
languages
JavaMOP
JavaFSM
JavaERE
JavaCFG
JavaPTLTL
JavaFTLTL
JavaPTCaRet
BusMOP
BusFSM
BusERE
BusPTLTL
MOP Matrix: a clickable map of MOP pages.
CFGPlugin
CFG Plugin Input Syntax
CFG Monitoring Algorithm
JavaCFG is an instance of MOP for Java and for CFG specifications. This instance is technically unnecessary, since one can simply run JavaMOP with CFG specifications, which is precisely what the online interface below does. JavaCFG has, however, conceptual (and potentially theoretical) value; many Java users prefer to specify properties exclusively as parametric context free patterns. Enter your CFG specification in the form below or chose (and modify) one example from the menu - provided examples are also reachable from the menu of the main JavaMOP interface. Go to JavaMOP for instructions on how to download and install it, as well as on how to compile the AspectJ monitors generated below.
Run JavaCFG Online
Enter your specification or chose (and modify) one example from the menu. Click Run to run JavaCFG. The generated monitor can be compiled using any AspectJ compiler; HERE are instructions on how to do it (must read it if you want to compile the generated aspect monitor!).
Choose an example:
SafeFile
SafeFileWriter
SafeLock
package cfg.locking;
import java.util.*;
SafeLock(Lock l, Thread t) {
event acq after(Lock l, Thread t) :
call(* Lock.lock()) && target(l) && thread(t) {}
event rel after(Lock+ l, Thread t) :
call(* Lock.unlock()) && target(l) && thread(t) {}
event begin before(Thread t) :
execution(* *.*(..)) && thread(t) {}
event end after(Thread t) :
execution(* *.*(..)) && thread(t) {}
cfg : S -> begin S end | acq S rel | S S | epsilon
Run
Reset
Specification Syntax Help (new window)
CFG Syntax Help (new window)
Number of generated monitors (since 14 December 2008): 1330 (MOP), 117 (CFG), 943 (JavaMOP), 91 (JavaCFG)
See the Plugin Output (new window)
package cfg.locking;
import java.util.*;
Done

END OF
DEMO ON SLIDES

properties of Java library APIs



R_2: An enumeration should not be propagated after the underlying vector has been changed .

CFG specification

```
package cfg.safeenum;

import java.io.*;
import java.util.*;

suffix SafeEnumeration(Vector v, Enumeration e) {
  event create after(Vector v) returning(Enumeration e) :
    call(Enumeration Vector.elements()) && target(v) {}

  event updatesource before(Vector v) :
    (call(* Vector.remove*(..)) || call(* Vector.add*(..)) ) && target(v) {}

  event next before(Enumeration e) :
    call(* Enumeration.nextElement()) && target(e) {}

  cfg :
    Bad -> create NextStar updatesource UpdateStar next,
    NextStar -> next NextStar | epsilon,
    UpdateStar -> updatesource UpdateStar | epsilon

  @match {
     System.out.println("improper enumerator usage");
  }
}
```

indexing works as for regular expressions

Action

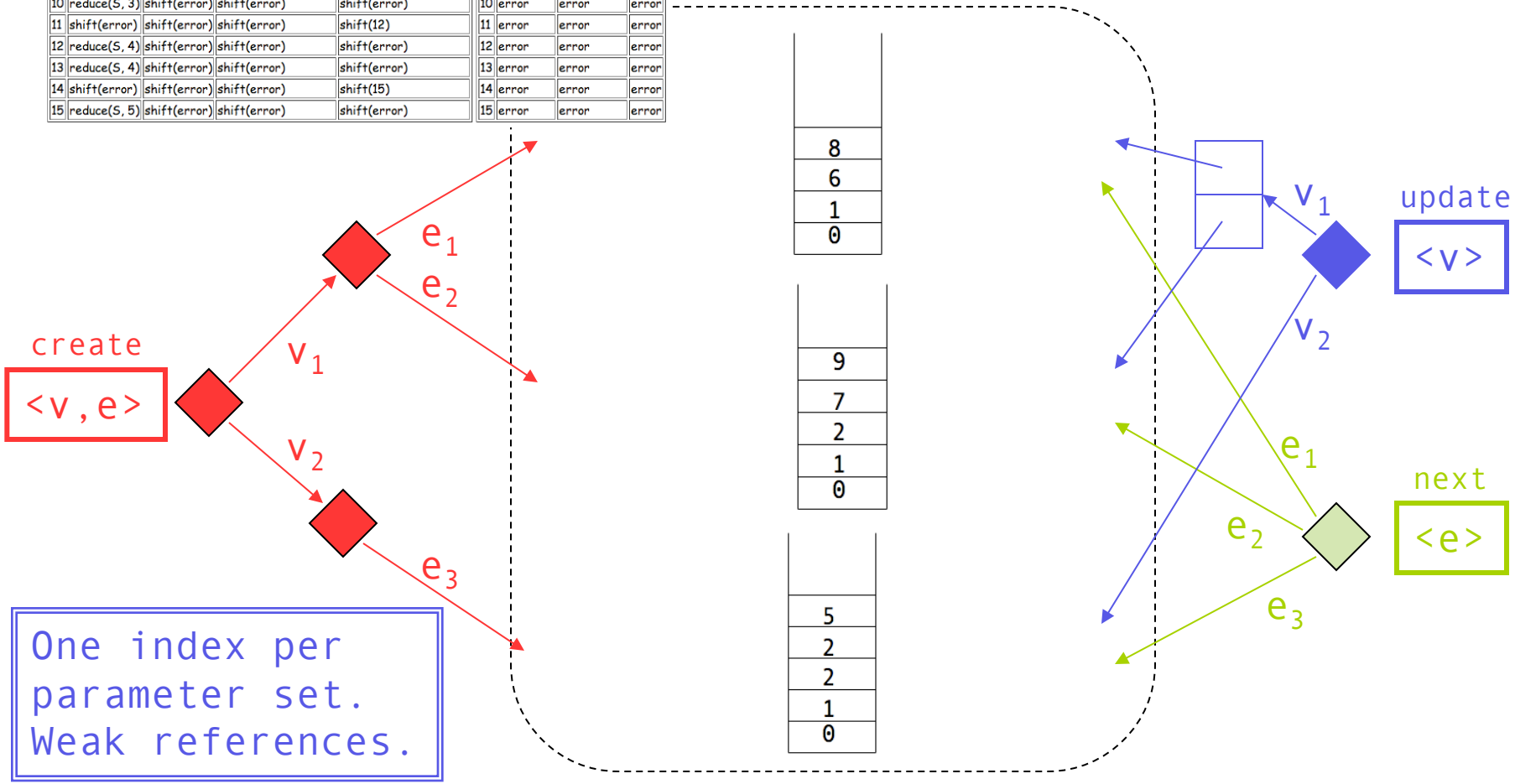
	$	create	updatesource	next
0	shift(error)	shift(6)	shift(error)	shift(error)
1	shift(error)	shift(error)	shift(5)	shift(error)
2	shift(error)	shift(error)	reduce(NextStar, 1)	shift(2)
3	shift(error)	shift(error)	shift(3)	reduce(UpdateStar, 1)
4	shift(error)	shift(error)	shift(3)	shift(10)
5	shift(error)	shift(error)	shift(3)	shift(13)
6	shift(error)	shift(error)	shift(4)	shift(2)
7	accept	shift(error)	shift(error)	shift(error)
8	shift(error)	shift(error)	reduce(NextStar, 2)	shift(error)
9	shift(error)	shift(error)	shift(error)	reduce(UpdateStar, 2)
10	reduce(S, 3)	shift(error)	shift(error)	shift(error)
11	shift(error)	shift(error)	shift(error)	shift(12)
12	reduce(S, 4)	shift(error)	shift(error)	shift(error)
13	reduce(S, 4)	shift(error)	shift(error)	shift(error)
14	shift(error)	shift(error)	shift(error)	shift(15)
15	reduce(S, 5)	shift(error)	shift(error)	shift(error)

Goto

	NextStar	UpdateStar	S
0	error	error	7
1	error	error	error
2	8	error	error
3	error	9	error
4	error	11	error
5	error	14	error
6	1	error	error
7	error	error	error
8	error	error	error
9	error	error	error
10	error	error	error
11	error	error	error
12	error	error	error
13	error	error	error
14	error	error	error
15	error	error	error

events:
create<v,e>
update<v>
next<e>

monitor creation event



One index per parameter set. Weak references.

end