

**1. Copyright.**

Copyright © Dave Bone 1998 - 2015

**2. *eval\_phrases* Grammar.**

Make sure all the grammar phases are properly parsed. This is due to my not programming a grammar sequencer to explicitly define the syntax of a grammar. I wanted to parse each phrase separately by keyword triggering a descent procedure.

Question: What source GPS do u align the error against? i just place it to the beginning of the grammar file if there is no previous phase seen. A previous phase becomes the reference point to the source file. If i was energetic, i should get the last token of that phase to be more accurate but today i'm lazy. The message is adequate to correct the problem.

**3. Fsm Ceval\_phrases class.****4. Ceval\_phrases op directive.**

```
< Ceval_phrases op directive 4 > ≡
    gps_ = 0;
```

**5. Ceval\_phrases user-declaration directive.**

```
< Ceval_phrases user-declaration directive 5 > ≡
public: CAbs_lr1_sym * gps_;
    void post_error(CAbs_lr1_sym * Err);
    void post_gps(CAbs_lr1_sym * Sym);
```

**6. Ceval\_phrases user-implementation directive.**

```
< Ceval_phrases user-implementation directive 6 > ≡
    void Ceval_phrases::post_error(CAbs_lr1_sym * Err)
    {
        using namespace NS_yacco2_T_enum;
        Err->set_rc(*gps_, __FILE__, __LINE__);
        if (gps_-enumerated_id_ ≡ T_Enum::T_LR1_eog_) {
            Err->set_line_no_and_pos_in_line(1, 1);
        }
        parser_>>add_token_to_error_queue(*Err);
        parser_>>set_abort_parse(true);
    }
```

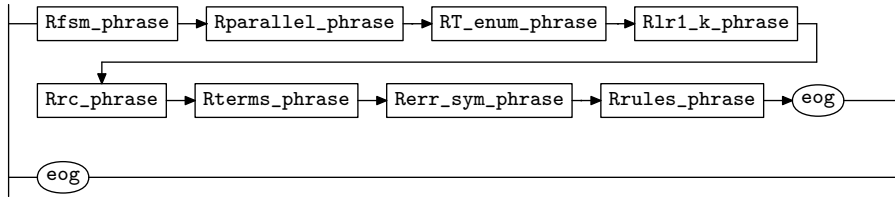
**7. *post\_gps*.**

```
< More code 7 > ≡
    void Ceval_phrases::post_gps(CAbs_lr1_sym * Sym)
    {
        gps_ = Sym;
    }
```

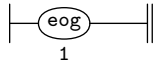
**8. Reval\_phrases rule.**

Use the property of Rule’s sequencing within a production to determine whether the parsed phrases were missed or out of sequence. Report them as errors.

Reval\_phrases



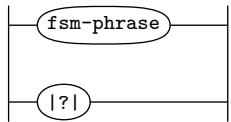
**9. Reval\_phrases’s subrule 2.**



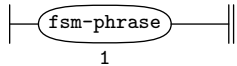
⟨ Reval\_phrases subrule 2 op directive 9 ⟩ ≡  
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser--fsm\_tbl...*;  
*fsm-post\_error*(**new** *Err\_empty\_file*);  
*fsm-post\_gps*(*sf-p1...*);

**10. Rfsm\_phrase rule.**

Rfsm\_phrase

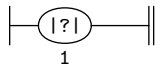


**11. Rfsm\_phrase’s subrule 1.**



⟨ Rfsm\_phrase subrule 1 op directive 11 ⟩ ≡  
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser--fsm\_tbl...*;  
*fsm-post\_gps*(*sf-p1...*);

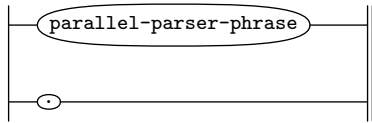
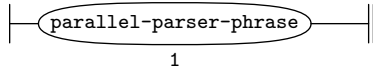
**12. Rfsm\_phrase’s subrule 2.**



⟨ Rfsm\_phrase subrule 2 op directive 12 ⟩ ≡  
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser--fsm\_tbl...*;  
*fsm-post\_gps*(*sf-p1...*);  
*fsm-post\_error*(**new** *ERR\_no\_fsm\_phrase*);

13. *Rparallel\_phrase* rule.

Rparallel\_phrase

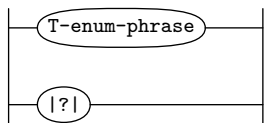
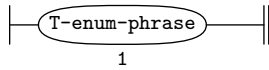
14. *Rparallel\_phrase*'s subrule 1.

$\langle$  Rparallel\_phrase subrule 1 op directive 14  $\rangle \equiv$

$Ceval\_phrases * fsm = ( Ceval\_phrases * ) rule\_info\_parser\_fsm\_tbl\_;$   
 $fsm\_post\_gps(sf-p1\_);$

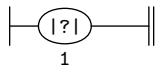
15. *RT\_enum\_phrase* rule.

RT\_enum\_phrase

16. *RT\_enum\_phrase*'s subrule 1.

$\langle$  RT\_enum\_phrase subrule 1 op directive 16  $\rangle \equiv$

$Ceval\_phrases * fsm = ( Ceval\_phrases * ) rule\_info\_parser\_fsm\_tbl\_;$   
 $fsm\_post\_gps(sf-p1\_);$

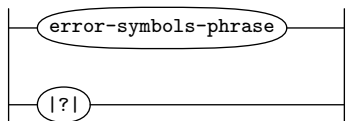
17. *RT\_enum\_phrase*'s subrule 2.

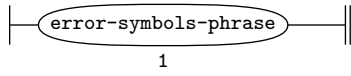
$\langle$  RT\_enum\_phrase subrule 2 op directive 17  $\rangle \equiv$

$Ceval\_phrases * fsm = ( Ceval\_phrases * ) rule\_info\_parser\_fsm\_tbl\_;$   
 $fsm\_post\_gps(sf-p1\_);$   
 $fsm\_post\_error(\mathbf{new} ERR\_no\_T\_enum\_phrase);$

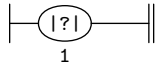
18. *Rerr\_sym\_phrase* rule.

Rerr\_sym\_phrase



**19. Rerr\_sym\_phrase's subrule 1.**

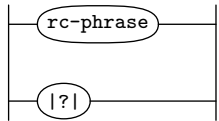
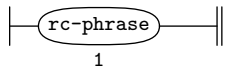
$\langle \text{Rerr\_sym\_phrase subrule 1 op directive 19} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;

**20. Rerr\_sym\_phrase's subrule 2.**

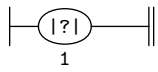
$\langle \text{Rerr\_sym\_phrase subrule 2 op directive 20} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;  
*fsm-post\_error(new ERR\_no\_errors\_phrase)*;

**21. Rrc\_phrase rule.**

Rrc\_phrase

**22. Rrc\_phrase's subrule 1.**

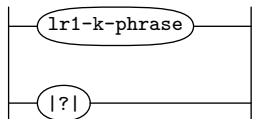
$\langle \text{Rrc\_phrase subrule 1 op directive 22} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;

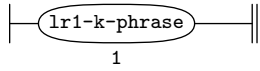
**23. Rrc\_phrase's subrule 2.**

$\langle \text{Rrc\_phrase subrule 2 op directive 23} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;  
*fsm-post\_error(new ERR\_no\_rc\_phrase)*;

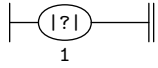
**24. Rlr1\_k\_phrase rule.**

Rlr1\_k\_phrase



**25. *Rlr1\_k\_phrase*'s subrule 1.**

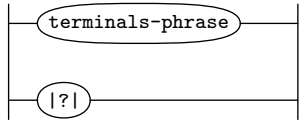
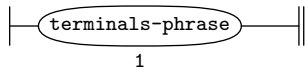
$\langle \text{Rlr1\_k\_phrase subrule 1 op directive 25} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;

**26. *Rlr1\_k\_phrase*'s subrule 2.**

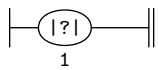
$\langle \text{Rlr1\_k\_phrase subrule 2 op directive 26} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;  
*fsm-post\_error(new ERR\_no\_lrk\_phrase)*;

**27. *Rterms\_phrase* rule.**

*Rterms\_phrase*

**28. *Rterms\_phrase*'s subrule 1.**

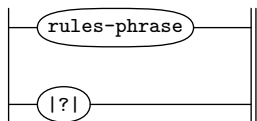
$\langle \text{Rterms\_phrase subrule 1 op directive 28} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;

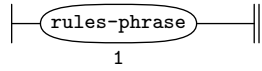
**29. *Rterms\_phrase*'s subrule 2.**

$\langle \text{Rterms\_phrase subrule 2 op directive 29} \rangle \equiv$   
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm\_tbl...*;  
*fsm-post\_gps(sf-p1...)*;  
*fsm-post\_error(new ERR\_no\_terminals\_phrase)*;

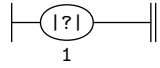
**30. *Rrules\_phrase* rule.**

*Rrules\_phrase*



**31.** *Rrules\_phrase's subrule 1.*

⟨ Rrules\_phrase subrule 1 op directive 31 ⟩ ≡  
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm.tbl...*;  
*fsm-post\_gps(sf-p1...)*;

**32.** *Rrules\_phrase's subrule 2.*

⟨ Rrules\_phrase subrule 2 op directive 32 ⟩ ≡  
*Ceval\_phrases* \* *fsm* = ( *Ceval\_phrases* \* ) *rule\_info...parser...fsm.tbl...*;  
*fsm-post\_gps(sf-p1...)*;  
*fsm-post\_error(new ERR\_no\_rules\_phrase)*;

**33. First Set Language for  $O_2^{linker}$ .**

```
/*
  File: eval_phrases.fsc
  Date and Time: Fri Jan  2 15:33:35 2015
*/
transitive      n
grammar-name    "eval_phrases"
name-space     "NS_eval_phrases"
thread-name     "Ceval_phrases"
monolithic      y
file-name      "eval_phrases.fsc"
no-of-T        569
list-of-native-first-set-terminals 3
  LR1_questionable_shift_operator
  LR1_eog
  T_fsm_phrase
end-list-of-native-first-set-terminals
list-of-transitive-threads 0
end-list-of-transitive-threads
list-of-used-threads 0
end-list-of-used-threads
fsm-comments
"Evaluate parse phrase sequencer: \n as i use a top / down approach to dispatching the
various phrases."
```



## 34. Lr1 State Network.

$\Rightarrow$					State: 1 state type: $s$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	Rfsm_phrase		2 2 1	?			1 2 2	
c	Reval_phrases		1 2 1	eog			1 3 3	
c	Rfsm_phrase		2 1 1	fsm-phrase			1 4 4	
c	Reval_phrases		1 1 1	Rfsm_phrase	<u>Rparallel_phrase<math>^{\epsilon}</math></u> <u>RT_enum_phrase</u>		1 5 13	
$\Rightarrow$	?				State: 2 state type: $r$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
t	Rfsm_phrase		2 2 2				1 0 2 1	
$\Rightarrow$	eog				State: 3 state type: $r$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
t	Reval_phrases		1 2 2				1 0 3 2	
$\Rightarrow$	fsm-phrase				State: 4 state type: $r$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
t	Rfsm_phrase		2 1 2				1 0 4 1	
$\Rightarrow$	Rfsm_phrase				State: 5 state type: $s/r$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	Rparallel_phrase		3 2 1	$\epsilon$			5 0 5 3	
c	Rparallel_phrase		3 1 1	parallel-parser-phrase			5 14 14	
t	Reval_phrases		1 1 2	Rparallel_phrase	<u>RT_enum_phrase</u>		1 6 13	
$\Rightarrow$	Rparallel_phrase				State: 6 state type: $s$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	RT_enum_phrase		4 2 1	?			6 15 15	
c	RT_enum_phrase		4 1 1	T-enum-phrase			6 16 16	
t	Reval_phrases		1 1 3	RT_enum_phrase	<u>Rlr1_k_phrase</u>		1 7 13	
$\Rightarrow$	RT_enum_phrase				State: 7 state type: $s$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	Rlr1_k_phrase		7 2 1	?			7 17 17	
c	Rlr1_k_phrase		7 1 1	lr1-k-phrase			7 18 18	
t	Reval_phrases		1 1 4	Rlr1_k_phrase	<u>Rrc_phrase</u>		1 8 13	
$\Rightarrow$	Rlr1_k_phrase				State: 8 state type: $s$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	Rrc_phrase		6 2 1	?			8 19 19	
c	Rrc_phrase		6 1 1	rc-phrase			8 20 20	
t	Reval_phrases		1 1 5	Rrc_phrase	<u>Rterms_phrase</u>		1 9 13	
$\Rightarrow$	Rrc_phrase				State: 9 state type: $s$			
$\leftarrow$	rule	$\rightarrow$	R# sr# Po	$\leftarrow$	subrule element	$\rightarrow$	Brn Gto Red LA	
c	Rterms_phrase		8 2 1	?			9 21 21	
c	Rterms_phrase		8 1 1	terminals-phrase			9 22 22	
t	Reval_phrases		1 1 6	Rterms_phrase	<u>Rerr_sym_phrase</u>		1 10 13	
$\Rightarrow$	Rterms_phrase				State: 10 state type: $s$			

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rerr_sym_phrase		5	2	1	←	?		10	23	23	
c	Rerr_sym_phrase		5	1	1	←	error-symbols-phrase		10	24	24	
t	Reval_phrases		1	1	7	←	Rerr_sym_phrase <u>Rrules_phrase</u>		1	11	13	
⇒ <i>Rerr_sym_phrase</i> State: 11 state type: <i>s</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rrules_phrase		9	2	1	←	?		11	25	25	
c	Rrules_phrase		9	1	1	←	rules-phrase		11	26	26	
t	Reval_phrases		1	1	8	←	Rrules_phrase <u>eog</u>		1	12	13	
⇒ <i>Rrules_phrase</i> State: 12 state type: <i>s</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Reval_phrases		1	1	9	←	eog		1	13	13	
⇒ <i>eog</i> State: 13 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Reval_phrases		1	1	10	←			1	0	13	2
⇒ <i>parallel-parser-phrase</i> State: 14 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rparallel_phrase		3	1	2	←			5	0	14	3
⇒ <i> ? </i> State: 15 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	RT_enum_phrase		4	2	2	←			6	0	15	4
⇒ <i>T-enum-phrase</i> State: 16 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	RT_enum_phrase		4	1	2	←			6	0	16	4
⇒ <i> ? </i> State: 17 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rlr1_k_phrase		7	2	2	←			7	0	17	5
⇒ <i>lr1-k-phrase</i> State: 18 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rlr1_k_phrase		7	1	2	←			7	0	18	5
⇒ <i> ? </i> State: 19 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rrc_phrase		6	2	2	←			8	0	19	6
⇒ <i>rc-phrase</i> State: 20 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rrc_phrase		6	1	2	←			8	0	20	6
⇒ <i> ? </i> State: 21 state type: <i>r</i>												
←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rterms_phrase		8	2	2	←			9	0	21	7
⇒ <i>terminals-phrase</i> State: 22 state type: <i>r</i>												

← rule	→ R# sr# Po ←	subrule element	→ Brn Gto Red LA
t Rterms_phrase	8 1 2		9 0 22 7
⇒ ?			
← rule	→ R# sr# Po ←	State: 23 state type: <i>r</i>	subrule element
t Rerr_sym_phrase	5 2 2		→ Brn Gto Red LA
			10 0 23 8
⇒ <i>error-symbols-phrase</i>			
← rule	→ R# sr# Po ←	State: 24 state type: <i>r</i>	subrule element
t Rerr_sym_phrase	5 1 2		→ Brn Gto Red LA
			10 0 24 8
⇒ ?			
← rule	→ R# sr# Po ←	State: 25 state type: <i>r</i>	subrule element
t Rrules_phrase	9 2 2		→ Brn Gto Red LA
			11 0 25 9
⇒ <i>rules-phrase</i>			
← rule	→ R# sr# Po ←	State: 26 state type: <i>r</i>	subrule element
t Rrules_phrase	9 1 2		→ Brn Gto Red LA
			11 0 26 9

**35. Index.**

$\epsilon$  : 13.  
 |?| : 10, 15, 18, 21, 24, 27, 30.  
 \_\_FILE\_\_: 6.  
 \_\_LINE\_\_: 6.  
 add\_token\_to\_error\_queue: 6.  
 CAbs\_lr1\_sym: 5, 6, 7.  
 Ceval\_phrases: 6, 7, 9, 11, 12, 14, 16, 17, 19, 20,  
 22, 23, 25, 26, 28, 29, 31, 32.  
 enumerated\_id\_: 6.  
 eog: 8.  
 Err: 5, 6.  
 Err\_empty\_file: 9.  
 ERR\_no\_errors\_phrase: 20.  
 ERR\_no\_fsm\_phrase: 12.  
 ERR\_no\_lrk\_phrase: 26.  
 ERR\_no\_rc\_phrase: 23.  
 ERR\_no\_rules\_phrase: 32.  
 ERR\_no\_T\_enum\_phrase: 17.  
 ERR\_no\_terminals\_phrase: 29.  
 error-symbols-phrase: 18.  
 eval\_phrases: 2.  
 fsm: 9, 11, 12, 14, 16, 17, 19, 20, 22, 23, 25,  
 26, 28, 29, 31, 32.  
 fsm-phrase: 10.  
 fsm\_tbl\_: 9, 11, 12, 14, 16, 17, 19, 20, 22, 23,  
 25, 26, 28, 29, 31, 32.  
 gps\_: 4, 5, 6, 7.  
 lr1-k-phrase: 24.  
 NS\_yacco2\_T\_enum: 6.  
 parallel-parser-phrase: 13.  
 parser\_: 6, 9, 11, 12, 14, 16, 17, 19, 20, 22, 23,  
 25, 26, 28, 29, 31, 32.  
 post\_error: 5, 6, 9, 12, 17, 20, 23, 26, 29, 32.  
 post\_gps: 5, 7, 9, 11, 12, 14, 16, 17, 19, 20, 22,  
 23, 25, 26, 28, 29, 31, 32.  
 p1\_: 9, 11, 12, 14, 16, 17, 19, 20, 22, 23, 25,  
 26, 28, 29, 31, 32.  
 rc-phrase: 21.  
 Rerr\_sym\_phrase: 8.  
 Rerr\_sym\_phrase: 18, 19, 20.  
 Reval\_phrases: 8, 9.  
 Rfsm\_phrase: 8.  
 Rfsm\_phrase: 10, 11, 12.  
 Rlr1\_k\_phrase: 8.  
 Rlr1\_k\_phrase: 24, 25, 26.  
 Rparallel\_phrase: 8.  
 Rparallel\_phrase: 13, 14.  
 Rrc\_phrase: 8.  
 Rrc\_phrase: 21, 22, 23.  
 Rrules\_phrase: 8.  
 Rrules\_phrase: 30, 31, 32.  
 RT\_enum\_phrase: 8.  
 RT\_enum\_phrase: 15, 16, 17.  
 Rterms\_phrase: 8.  
 Rterms\_phrase: 27, 28, 29.  
 rule\_info\_: 9, 11, 12, 14, 16, 17, 19, 20, 22, 23,  
 25, 26, 28, 29, 31, 32.  
 rules-phrase: 30.  
 set\_abort\_parse: 6.  
 set\_line\_no\_and\_pos\_in\_line: 6.  
 set\_rc: 6.  
 sf: 9, 11, 12, 14, 16, 17, 19, 20, 22, 23, 25,  
 26, 28, 29, 31, 32.  
 Sym: 5, 7.  
 T-enum-phrase: 15.  
 T\_Enum: 6.  
 T\_LR1\_eog\_: 6.  
 terminals-phrase: 27.  
 true: 6.

< Ceval\_phrases op directive 4 >  
< Ceval\_phrases user-declaration directive 5 >  
< Ceval\_phrases user-implementation directive 6 >  
< More code 7 >  
< RT\_enum\_phrase subrule 1 op directive 16 >  
< RT\_enum\_phrase subrule 2 op directive 17 >  
< Rerr\_sym\_phrase subrule 1 op directive 19 >  
< Rerr\_sym\_phrase subrule 2 op directive 20 >  
< Reval\_phrases subrule 2 op directive 9 >  
< Rfsm\_phrase subrule 1 op directive 11 >  
< Rfsm\_phrase subrule 2 op directive 12 >  
< Rlr1\_k\_phrase subrule 1 op directive 25 >  
< Rlr1\_k\_phrase subrule 2 op directive 26 >  
< Rparallel\_phrase subrule 1 op directive 14 >  
< Rrc\_phrase subrule 1 op directive 22 >  
< Rrc\_phrase subrule 2 op directive 23 >  
< Rrules\_phrase subrule 1 op directive 31 >  
< Rrules\_phrase subrule 2 op directive 32 >  
< Rterms\_phrase subrule 1 op directive 28 >  
< Rterms\_phrase subrule 2 op directive 29 >

# eval\_phrases Grammar

Date: January 2, 2015 at 15:35

File: eval\_phrases.lex

Ns: NS\_eval\_phrases

Version: 1.0

Debug: true

Grammar Comments:

Type: Monolithic

Evaluate parse phrase sequencer: as i use a top / down approach to dispatching the various phrases.

	Section	Page
<b>Copyright</b> .....	1	1
<i>eval_phrases</i> <b>Grammar</b> .....	2	2
Fsm Ceval_phrases class .....	3	2
Ceval_phrases op directive .....	4	2
Ceval_phrases user-declaration directive .....	5	2
Ceval_phrases user-implementation directive .....	6	2
<i>post_gps</i> .....	7	2
<i>Reval_phrases</i> rule .....	8	3
<i>Reval_phrases</i> 's subrule 2 .....	9	3
<i>Rfsm_phrase</i> rule .....	10	3
<i>Rfsm_phrase</i> 's subrule 1 .....	11	3
<i>Rfsm_phrase</i> 's subrule 2 .....	12	3
<i>Rparallel_phrase</i> rule .....	13	4
<i>Rparallel_phrase</i> 's subrule 1 .....	14	4
<i>RT_enum_phrase</i> rule .....	15	4
<i>RT_enum_phrase</i> 's subrule 1 .....	16	4
<i>RT_enum_phrase</i> 's subrule 2 .....	17	4
<i>Rerr_sym_phrase</i> rule .....	18	4
<i>Rerr_sym_phrase</i> 's subrule 1 .....	19	5
<i>Rerr_sym_phrase</i> 's subrule 2 .....	20	5
<i>Rrc_phrase</i> rule .....	21	5
<i>Rrc_phrase</i> 's subrule 1 .....	22	5
<i>Rrc_phrase</i> 's subrule 2 .....	23	5
<i>Rlr1_k_phrase</i> rule .....	24	5
<i>Rlr1_k_phrase</i> 's subrule 1 .....	25	6
<i>Rlr1_k_phrase</i> 's subrule 2 .....	26	6
<i>Rterms_phrase</i> rule .....	27	6
<i>Rterms_phrase</i> 's subrule 1 .....	28	6
<i>Rterms_phrase</i> 's subrule 2 .....	29	6
<i>Rrules_phrase</i> rule .....	30	6
<i>Rrules_phrase</i> 's subrule 1 .....	31	7
<i>Rrules_phrase</i> 's subrule 2 .....	32	7
<b>First Set Language for <math>O_2^{linker}</math></b> .....	33	8
<b>Lr1 State Network</b> .....	34	9

eval\_phrases Grammar

TABLE OF CONTENTS 1

**Index** ..... 35 12