1. *c_k_of_cweb_or_cpp* **Thread.** Eliminate the non-determinism between these 2 threads due to their common prefix. Why? Speed and stop the confusion across the threads address space when called by procedure instead of by thread. The thread gets called singlely and hence the speed versus parallel threads competing where only one will win. Some interesting debuging issues where found when using Sun's superb mdb and dtrace tools.

2. Cpp comments.
Handles both flavours of c++ comments.

1) // single line type comment
2) /* ... */ type comment

To speed things up, i'm using a finite-state approach to lower the pushdown deterministic rule push / pop cycles.

3. Cweb type comments.
Recognize *cweb* comments buried within a grammar. These comments start with "/@" followed by the comments and closed off by "@" and a forward slash — can't declare it as this will prematurely end this comment. The enclosing comment prefix / suffix are not appended to the data as only its contents that are outputed in the various documents.

Use of the `|.|`symbol is used to lower the Lookahead set members.

4. Fsm Cc_k_of_cweb_or_cpp class.

5. Cc_k_of_cweb_or_cpp constructor directive.

⟨ Cc_k_of_cweb_or_cpp constructor directive 5 ⟩ ≡
```
data_.clear( );
```

6. Cc_k_of_cweb_or_cpp op directive.

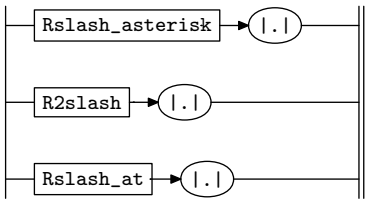
⟨ Cc_k_of_cweb_or_cpp op directive 6 ⟩ ≡
```
data_.clear( );
```

7. Cc_k_of_cweb_or_cpp user-declaration directive.

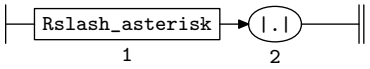
⟨ Cc_k_of_cweb_or_cpp user-declaration directive 7 ⟩ ≡
```
public: std::string data_;
```

8. *Rc_k_of_cweb_or_cpp* **rule.**

Rc_k_of_cweb_or_cpp

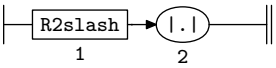


9. *Rc_k_of_cweb_or_cpp***'s subrule 1.**



⟨ Rc_k_of_cweb_or_cpp subrule 1 op directive 9 ⟩ ≡
```
Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__->fsm_tbl__;
T_comment * com = new T_comment(fsm->data_);
com->set_rc(*rule_info__.parser__->start_token__, *rule_info__.parser__, __FILE__, __LINE__);
com->set_line_no_and_pos_in_line(*rule_info__.parser__->start_token__);
RSVP(com);
```

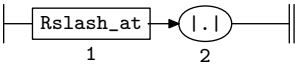
10. *Rc_k_of_cweb_or_cpp*'**s subrule 2.**



⟨ Rc_k_of_cweb_or_cpp subrule 2 op directive 10 ⟩ ≡

```
Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__→fsm_tbl__;
T_comment * com = new T_comment(fsm→data_);
com→set_rc(*rule_info__.parser__→start_token__, *rule_info__.parser__, __FILE__, __LINE__);
com→set_line_no_and_pos_in_line(*rule_info__.parser__→start_token__);
RSVP(com);
```

11. *Rc_k_of_cweb_or_cpp*'**s subrule 3.**

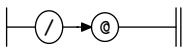


⟨ Rc_k_of_cweb_or_cpp subrule 3 op directive 11 ⟩ ≡

```
Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__→fsm_tbl__;
T_cweb_comment * com = new T_cweb_comment(fsm→data_);
com→set_rc(*rule_info__.parser__→start_token__, *rule_info__.parser__, __FILE__, __LINE__);
  /* file marker */
com→set_line_no_and_pos_in_line(*rule_info__.parser__→start_token__);
RSVP(com);
```

12. *Rslash_at* **rule.**

```
Rslash_at
```

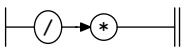


⟨ Rslash_at subrule 1 op directive 12 ⟩ ≡

```
  Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__→fsm_tbl__;
loop:
  switch (rule_info__.parser__→current_token( )→enumerated_id__) {
  case T_Enum :: T_raw_lf_: goto other;
  case T_Enum :: T_raw_cr_: goto cr;
  case T_Enum :: T_LR1_eof_: goto overrun;
  case T_Enum :: T_LR1_eog_: goto overrun;
  case T_Enum :: T_raw_at_sign_: goto atsign;
  default: goto other;
  }
cr:
  {     /* cr lf? */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    if (rule_info__.parser__→current_token( )→enumerated_id__ ≠ T_Enum :: T_raw_lf_) goto loop;
        /* not cr lf */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    goto loop;
  }
  ;
atsign:
  {     /* end of k? */
    rule_info__.parser__→get_next_token( );
    if (rule_info__.parser__→current_token( )→enumerated_id__ ≠ T_Enum :: T_raw_slash_) {
      fsm→data_ += "@";
      goto loop;      /* false eok */
    }
    rule_info__.parser__→get_next_token( );
    return;
  }
overrun:
  {
    CAbs_lr1_sym * sym = new LR1_err_comment_overrun;
    sym→set_rc(*rule_info__.parser__→start_token__, *rule_info__.parser__, __FILE__, __LINE__);
    sym→set_line_no_and_pos_in_line(*rule_info__.parser__→start_token__);
    RSVP(sym);
    rule_info__.parser__→set_stop_parse(true);
    return;
  }
other:
  {
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    goto loop;
  }
```

13. *Rslash_asterisk* **rule.**

Rslash_asterisk

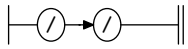


```
⟨ Rslash_asterisk subrule 1 op directive 13 ⟩ ≡
  Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__→fsm_tbl__;
  fsm→data_ += '/';      /* due to lex scanner */
  fsm→data_ += "*";
loop:
  switch (rule_info__.parser__→current_token( )→enumerated_id__) {
  case T_Enum :: T_raw_lf_: goto other;
  case T_Enum :: T_raw_cr_: goto cr;
  case T_Enum :: T_LR1_eof_: goto overrun;
  case T_Enum :: T_LR1_eog_: goto overrun;
  case T_Enum :: T_raw_asteric_: goto aster;
  default: goto other;
  }
cr:
  {     /* cr lf? */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    if (rule_info__.parser__→current_token( )→enumerated_id__ ≠ T_Enum :: T_raw_lf_) goto loop;
        /* not cr lf */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    goto loop;
  }
  ;
aster:
  {     /* end of k? */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    if (rule_info__.parser__→current_token( )→enumerated_id__ ≠ T_Enum :: T_raw_slash_) goto loop;
        /* false eok */
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    return;
  }
overrun:
  {
    CAbs_lr1_sym * sym = new LR1_err_comment_overrun;
    sym→set_rc(*rule_info__.parser__→start_token__, *rule_info__.parser__, __FILE__, __LINE__);
    sym→set_line_no_and_pos_in_line(*rule_info__.parser__→start_token__);
    RSVP(sym);
    rule_info__.parser__→set_stop_parse(true);
    return;
  }
other:
  {
    fsm→data_ += rule_info__.parser__→current_token( )→id__;
    rule_info__.parser__→get_next_token( );
    goto loop;
```

}

14. *R2slash* **rule.**

`R2slash`



⟨ R2slash subrule 1 op directive 14 ⟩ ≡

```
Cc_k_of_cweb_or_cpp * fsm = ( Cc_k_of_cweb_or_cpp * ) rule_info__.parser__→fsm_tbl__;
fsm→data_ += '/';
fsm→data_ += '/';
for ( ; ; ) {
  switch (rule_info__.parser__→current_token( )→enumerated_id__) {
  case T_Enum :: T_raw_lf_:
    {
      return;
    }
  case T_Enum :: T_raw_cr_:
    {
      return;
    }
  case T_Enum :: T_LR1_eof_: return;
  case T_Enum :: T_LR1_eog_: return;
  }
  fsm→data_ += rule_info__.parser__→current_token( )→id__;
  rule_info__.parser__→get_next_token( );
}
```

15. First Set Language for O_2^{linker}.

```
/*
 File: c_k_of_cweb_or_cpp.fsc
 Date and Time: Sun Oct 30 13:39:14 2011
*/
transitive   n
grammar-name "c_k_of_cweb_or_cpp"
name-space   "NS_c_k_of_cweb_or_cpp"
thread-name  "TH_c_k_of_cweb_or_cpp"
monolithic   n
file-name    "c_k_of_cweb_or_cpp.fsc"
no-of-T      569
list-of-native-first-set-terminals 1
   raw_slash
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
"C++ or cweb type comments lexer."
```

16. Lr1 State Network.

⇒ State: 1 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rslash_at		2	1	1	/			1	2	5	
c	Rslash_asterisk		3	1	1	/			1	2	3	
c	R2slash		4	1	1	/			1	2	4	
c	Rc_k_of_cweb_or_cpp		1	3	1	Rslash_at |.|			1	6	7	
c	Rc_k_of_cweb_or_cpp		1	1	1	Rslash_asterisk |.|			1	8	9	
c	Rc_k_of_cweb_or_cpp		1	2	1	R2slash |.|			1	10	11	

⇒/ State: 2 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rslash_asterisk		3	1	2	*			1	3	3	
t	R2slash		4	1	2	/			1	4	4	
t	Rslash_at		2	1	2	@			1	5	5	

⇒* State: 3 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rslash_asterisk		3	1	3				1	0	3	1

⇒/ State: 4 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	R2slash		4	1	3				1	0	4	1

⇒@ State: 5 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rslash_at		2	1	3				1	0	5	1

⇒*Rslash_at* State: 6 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	3	2	|.|			1	7	7	

⇒\|.\| State: 7 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	3	3				1	0	7	2

⇒*Rslash_asterisk* State: 8 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	1	2	|.|			1	9	9	

⇒\|.\| State: 9 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	1	3				1	0	9	2

⇒*R2slash* State: 10 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	2	2	|.|			1	11	11	

⇒\|.\| State: 11 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rc_k_of_cweb_or_cpp		1	2	3				1	0	11	2

17. Index.

⟨ Cc_k_of_cweb_or_cpp constructor directive 5 ⟩
⟨ Cc_k_of_cweb_or_cpp op directive 6 ⟩
⟨ Cc_k_of_cweb_or_cpp user-declaration directive 7 ⟩
⟨ R2slash subrule 1 op directive 14 ⟩
⟨ Rc_k_of_cweb_or_cpp subrule 1 op directive 9 ⟩
⟨ Rc_k_of_cweb_or_cpp subrule 2 op directive 10 ⟩
⟨ Rc_k_of_cweb_or_cpp subrule 3 op directive 11 ⟩
⟨ Rslash_asterisk subrule 1 op directive 13 ⟩
⟨ Rslash_at subrule 1 op directive 12 ⟩

c_k_of_cweb_or_cpp Grammar

Date: October 30, 2011 at 13:47

File: c_k_of_cweb_or_cpp.lex Ns: NS_c_k_of_cweb_or_cpp

Version: 1.0 Debug: false

Grammar Comments: Type: Thread

C++ or cweb type comments lexer.

1 element(s) in Lookahead Expression below

eolr