1. Copyright.

2. *cweave_lhs_sdc* **grammar.**
Write out cweave rule's lhs directives sentences.

3. Fsm Ccweave_lhs_sdc class.

4. Ccweave_lhs_sdc user-declaration directive.

⟨ Ccweave_lhs_sdc user-declaration directive 4 ⟩ ≡

```
public: std::ofstream * cweave_file_;
  KCHARP rule_name_;
  void initialize(std::ofstream * Cweave_file, yacco2::KCHARP Rule_name);
  void wrt_directive(const char *Directive, T_syntax_code * Sdc);
  void output_sdcode_title(const char *Directive);
```

5. Ccweave_lhs_sdc user-implementation directive.

⟨ Ccweave_lhs_sdc user-implementation directive 5 ⟩ ≡

```
void Ccweave_lhs_sdc::initialize(std::ofstream * Cweave_file, yacco2::KCHARP Rule_name)
{
  cweave_file_ = Cweave_file;
  rule_name_ = Rule_name;
}
```

6. *wrt_directive*.

```
⟨ More code 6 ⟩ ≡
  void Ccweave_lhs_sdc::wrt_directive(const char *Directive, T_syntax_code * Sdc)
  {
    char big_buf_[BIG_BUFFER_32K];
    char xa[Max_cweb_item_size];
    KCHARP cweave_sentence = "@<␣%s␣%s␣directive@>=\n";
    output_sdcode_title(Directive);
    if (Sdc ≡ 0) {
      XLATE_SYMBOLS_FOR_cweave(rule_name_, xa);
      sprintf(big_buf_, cweave_sentence, xa, Directive);
      (*cweave_file_) ≪ "/";
      (*cweave_file_) ≪ "/␣no␣sdcode" ≪ endl;
      return;
    }
    if (Sdc→cweb_marker() ≠ 0) {
      WRT_CWEB_MARKER(cweave_file_, Sdc→cweb_marker());
    }
    XLATE_SYMBOLS_FOR_cweave(rule_name_, xa);
    sprintf(big_buf_, cweave_sentence, xa, Directive);
    (*cweave_file_) ≪ big_buf_;
    string xlate;
    int len = Sdc→syntax_code()→length();
    string & sdc = *Sdc→syntax_code();      /* prescan @ due to cweave reqmts */
    for (int x = 0; x < len; ++x) {
      char nc = sdc[x];
      if (nc ≡ '@') {      /* check next char for cweave type directives */
        char nnc = sdc[x + 1];
        if ((nnc ≡ '*') ∨ (nnc ≡ '<') ∨ (nnc ≡ '>')) {
          xlate += nc;
          continue;
        }
        else {
          xlate += "@@";
          continue;
        }
      }
      else {
        xlate += nc;
        continue;
      }
    }
    (*cweave_file_) ≪ xlate.c_str() ≪ endl;
  }
```

See also section 7.

7. *output_sdcode_title*.

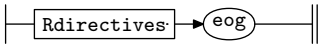
```
⟨ More code 6 ⟩ +≡
  void Ccweave_lhs_sdc::output_sdcode_title(const char *Directive)
  {
    char big_buf_[BIG_BUFFER_32K];
    char xa[Max_cweb_item_size];
    XLATE_SYMBOLS_FOR_cweave(rule_name_, xa);
    KCHARP cweave_sentence = "@*3␣%s␣%s␣directive.\n";
    sprintf(big_buf_, cweave_sentence, xa, Directive);
    (*cweave_file_) ≪ big_buf_;
  }
```

8. Ccweave_lhs_sdc user-prefix-declaration directive.

```
⟨ Ccweave_lhs_sdc user-prefix-declaration directive 8 ⟩ ≡
#include "o2_externs.h"
```

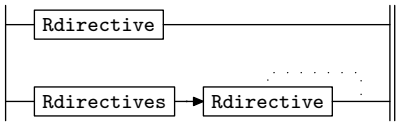
9. *Rweave_sdc* **rule.**

Rweave_sdc



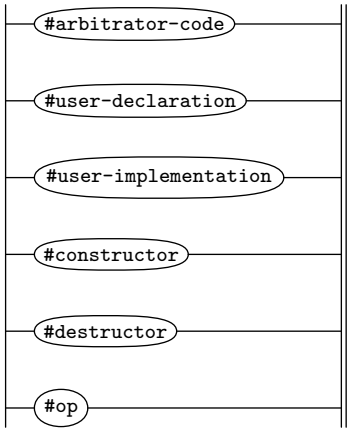
10. *Rdirectives* **rule.**

Rdirectives

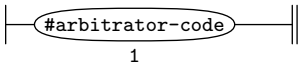


11. *Rdirective* **rule.**

Rdirective



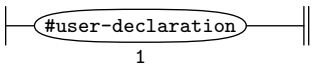
12. *Rdirective*'s subrule 1.



⟨ Rdirective subrule 1 op directive 12 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__->fsm_tbl__;
string include_war_code;
include_war_code.append("#include␣\"war_begin_code.h\"\n");
include_war_code.append(sf->p1__->syntax_code( )->syntax_code( )->c_str( ));
include_war_code.append("#include␣\"war_end_code.h\"\n");
sf->p1__->syntax_code( )->syntax_code( )->clear( );
sf->p1__->syntax_code( )->syntax_code( )->append(include_war_code.c_str( ));
KCHARP sdc = "arbitrator-code";
fsm->wrt_directive(sdc, sf->p1__->syntax_code( ));
```

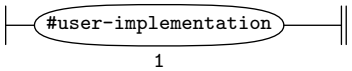
13. *Rdirective*'s subrule 2.



⟨ Rdirective subrule 2 op directive 13 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__->fsm_tbl__;
KCHARP sdc = "user-declaration";
fsm->wrt_directive(sdc, sf->p1__->syntax_code( ));
```

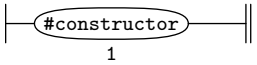
14. *Rdirective*'s subrule 3.



⟨ Rdirective subrule 3 op directive 14 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__->fsm_tbl__;
KCHARP sdc = "user-implementation";
fsm->wrt_directive(sdc, sf->p1__->syntax_code( ));
```

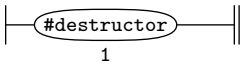
15. *Rdirective*'s subrule 4.



⟨ Rdirective subrule 4 op directive 15 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__->fsm_tbl__;
KCHARP sdc = "constructor";
fsm->wrt_directive(sdc, sf->p1__->syntax_code( ));
```

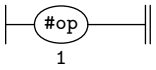
16. *Rdirective*'s subrule 5.



⟨ Rdirective subrule 5 op directive 16 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__->fsm_tbl__;
KCHARP sdc = "destructor";
fsm->wrt_directive(sdc, sf->p1__->syntax_code( ));
```

17. *Rdirective*'s subrule 6.



⟨ Rdirective subrule 6 op directive 17 ⟩ ≡

```
Ccweave_lhs_sdc * fsm = ( Ccweave_lhs_sdc * ) rule_info__.parser__→fsm_tbl__;
KCHARP sdc = "op";
fsm→wrt_directive(sdc, sf→p1__→syntax_code( ));
```

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

```
/*
 File: cweave_lhs_sdc.fsc
 Date and Time: Fri Jan  2 15:33:30 2015
*/
transitive   n
grammar-name "cweave_lhs_sdc"
name-space   "NS_cweave_lhs_sdc"
thread-name  "Ccweave_lhs_sdc"
monolithic   y
file-name    "cweave_lhs_sdc.fsc"
no-of-T      569
list-of-native-first-set-terminals 6
   T_arbitrator_code
   T_user_declaration
   T_constructor
   T_destructor
   T_op
   T_user_implementation
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
"Write out cweave rule's lhs directives sentences"
```

19. Lr1 State Network.

⇒ State: 1 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rdirective		3	1	1	#	arbitrator-code		1	2	2	
c	Rdirective		3	2	1	#	user-declaration		1	3	3	
c	Rdirective		3	4	1	#	constructor		1	4	4	
c	Rdirective		3	5	1	#	destructor		1	5	5	
c	Rdirective		3	6	1	#	op		1	6	6	
c	Rdirective		3	3	1	#	user-implementation		1	7	7	
c	Rdirectives		2	2	1	Rdirectives	*Rdirective*		1	8	10	
c	Rweave_sdc		1	1	1	Rdirectives	*eog*		1	8	9	
c	Rdirectives		2	1	1	Rdirective			1	11	11	

⇒*#arbitrator−code* State: 2 state type: ^{r}

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

⇒*#user−declaration* State: 3 state type: ^{r}

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

⇒*#constructor* State: 4 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	4	2				1	0	4	1

⇒*#destructor* State: 5 state type: ^{r}

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

⇒*#op* State: 6 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rdirective		3	6	2				1	0	6	1

⇒*#user−implementation* State: 7 state type: ^{r}

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

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

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rweave_sdc		1	1	2	eog			1	9	9	
c	Rdirective		3	1	1	#	arbitrator-code		8	2	2	
c	Rdirective		3	2	1	#	user-declaration		8	3	3	
c	Rdirective		3	4	1	#	constructor		8	4	4	
c	Rdirective		3	5	1	#	destructor		8	5	5	
c	Rdirective		3	6	1	#	op		8	6	6	
c	Rdirective		3	3	1	#	user-implementation		8	7	7	
t	Rdirectives		2	2	2	Rdirective			1	10	10	

⇒*eog* State: 9 state type: ^{r}

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

$\Rightarrow^{Rdirective}$

←	rule	→	R#	sr#	Po	←
t	Rdirectives		2	2	3	

State: 10 state type: ^{r}

	subrule element	→	Brn	Gto	Red	LA
			1	0	10	1

$\Rightarrow^{Rdirective}$

←	rule	→	R#	sr#	Po	←
t	Rdirectives		2	1	2	

State: 11 state type: ^{r}

	subrule element	→	Brn	Gto	Red	LA
			1	0	11	1

20. Index.

⟨ Ccweave_lhs_sdc user-declaration directive 4 ⟩
⟨ Ccweave_lhs_sdc user-implementation directive 5 ⟩
⟨ Ccweave_lhs_sdc user-prefix-declaration directive 8 ⟩
⟨ More code 6, 7 ⟩
⟨ Rdirective subrule 1 op directive 12 ⟩
⟨ Rdirective subrule 2 op directive 13 ⟩
⟨ Rdirective subrule 3 op directive 14 ⟩
⟨ Rdirective subrule 4 op directive 15 ⟩
⟨ Rdirective subrule 5 op directive 16 ⟩
⟨ Rdirective subrule 6 op directive 17 ⟩

cweave_lhs_sdc Grammar

Date: January 2, 2015 at 15:34

File: cweave_lhs_sdc.lex

Ns: NS_cweave_lhs_sdc

Version: 1.0 Debug: false

Grammar Comments: Type: Monolithic

Write out cweave rule's lhs directives sentences