

1. Copyright.

Copyright © Dave Bone 1998 - 2015

2. *mpost_output* Grammar.

Write out the *cweb*'s representation of the grammar's code and *mpost*'s railroad diagrams. These appropriate files have the grammar's name along with extension "w" for *cweave* to compile and extension "mp" for *mport* to draw from.

Enjoy the art work. All the grammar's vocabulary are cross referenced to accomodate the grammar writer. The railroad diagrams show left recursion and where the subrule has appropriate syntax directed code, it redraws the subrule with each element's position number below it. This aids in the writing of syntax directed code when individual parameters are referenced by use of the *sf* stack frame structure. Each element within the stack frame has a naming convention of *p#__* where the number sign is the element's position within the subrule sentence relative to 1.

Fodder to this grammar is provided by the tree container doing a prefix traverse. Why a prefix traverse? The parent node allowed me to declare the *mport* variables before their use.

Caveat: Grammar c++ comments must observe *cweave* dictums: underscored items like variable names must be enclosed with | and the number sign "#" must use its literal of backslash #. Without these observances, *pdftex* will honk.

3. Fsm Cmport_output class.

4. Cmport_output constructor directive.

```
< Cmport_output constructor directive 4 > ≡
    MPOST_CWEB_LOAD_XLATE_CHRS(this);
```

5. Cmpost_output op directive.

Due to a MS 2005 Visual Studio deficiency, part of the ctor's initialization is included here instead of all being done by MPOST_CWEB_EMIT_PREFIX_CODE. The reason is noted in the documentation of *o2externs*.

```
(Cmpost_output op directive 5) ≡
  no_subrules_per_rule_.push_back(0);
  time_t theTime = time(0);
  char *cTime = ctime(&theTime);
  gened_date_time_ += string(cTime);
  int n = gened_date_time_.find('\n');
  gened_date_time_[n] = '\u';
  mp_dimension_ += "\u00e1\u00e2\u00e3\u00e4\u00e5\u00e6\u00e7\u00e8\u00e9\u00e0\u00e1\u00e2\u00e3\u00e4\u00e5\u00e6\u00e7\u00e8\u00e9";
  w_fig_no_ = 0;
  rule_def_ = 0;
  subrule_def_ = 0;
  rule_no_ = 0;
  subrule_no_ = 0;
  elem_no_ = 0;
  no_of_rules_ = 0;
  no_of_subrules_ = 0;
  mp_filename_ += grammar_filename_prefix_.c_str();
  mp_filename_ += ".mp";
  omp_file_.open(mp_filename_.c_str(), ios_base::out | ios::trunc);
  if (!omp_file_) {
    CAbs_lr1_sym *sym = new Err_bad_filename(mp_filename_.c_str());
    sym->set_who_created(__FILE__, __LINE__);
    parser_->add_token_to_error_queue(*sym);
    parser_->set_stop_parse(true);
    return;
  }
  w_filename_ += grammar_filename_prefix_.c_str();
  w_filename_ += ".w";
  ow_file_.open(w_filename_.c_str(), ios_base::out | ios::trunc);
  if (!ow_file_) {
    CAbs_lr1_sym *sym = new Err_bad_filename(w_filename_.c_str());
    sym->set_who_created(__FILE__, __LINE__);
    parser_->add_token_to_error_queue(*sym);
    parser_->set_stop_parse(true);
    return;
  }
  w_index_filename_ += grammar_filename_prefix_.c_str();
  w_index_filename_ += "_idx.w";
  ow_index_file_.open(w_index_filename_.c_str(), ios_base::out | ios::trunc);
  if (!ow_index_file_) {
    CAbs_lr1_sym *sym = new Err_bad_filename(w_index_filename_.c_str());
    sym->set_who_created(__FILE__, __LINE__);
    parser_->add_token_to_error_queue(*sym);
    parser_->set_stop_parse(true);
    return;
  }
  KCHARP t_file = "T.w";
  ow_t_file_.open(t_file, ios_base::out | ios::trunc);
```

```

if ( $\neg$ ow_t_file_) {
  CAbs_lr1_sym * sym = new Err_bad_filename(t_file);
  sym $\rightarrow$ set_who_created(_FILE_, _LINE_);
  parser $\rightarrow$ add_token_to_error_queue(*sym);
  parser $\rightarrow$ set_stop_parse(true);
  return;
}
KCHARP err_file = "Err.w";
ow_err_file_.open(err_file, ios_base::out | ios::trunc);
if ( $\neg$ ow_err_file_) {
  CAbs_lr1_sym * sym = new Err_bad_filename(err_file);
  sym $\rightarrow$ set_who_created(_FILE_, _LINE_);
  parser $\rightarrow$ add_token_to_error_queue(*sym);
  parser $\rightarrow$ set_stop_parse(true);
  return;
}
KCHARP l rk_file = "Lrk.w";
ow_l rk_file_.open(l rk_file, ios_base::out | ios::trunc);
if ( $\neg$ ow_l rk_file_) {
  CAbs_lr1_sym * sym = new Err_bad_filename(l rk_file);
  sym $\rightarrow$ set_who_created(_FILE_, _LINE_);
  parser $\rightarrow$ add_token_to_error_queue(*sym);
  parser $\rightarrow$ set_stop_parse(true);
  return;
}
MPOST_CWEB_EMIT_PREFIX_CODE(this);

```

6. Cmpost_output user-declaration directive.

```

⟨ Cmpost_output user-declaration directive 6 ⟩ ≡
public: char big_buf_[BIG_BUFFER_32K]; std::map < int , std::string > xlated_names_; std::vector <
int > no_subrules_per_rule_;

std::string gened_date_time_;
std::string mp_filename_;
std::ofstream omp_file_;
std::string w_filename_;
std::string w_index_filename_;
std::ofstream ow_file_;
std::ofstream ow_index_file_;
std::ofstream ow_t_file_;
std::ofstream ow_err_file_;
std::ofstream ow_lrk_file_;
std::string grammar_filename_prefix_;
std::string fq_filename_noext_;

int w_fig_no_;
int rule_no_;
int subrule_no_;
int elem_no_;
int no_of_rules_;
int no_of_subrules_;

std::string rule_name_;
std::string elem_name_;
rule_def * rule_def_;
T_subrule_def * subrule_def_;
std::string mp_dimension_;

static void MPOST_CWEB_gen_dimension_name(Cmpost_output * Fsm, std::string & Mp_obj_name, int
Dimension);
static void MPOST_CWEB_calc_mp_obj_name(Cmpost_output * Fsm, std::string & Mp_obj_name, int
Elem_no);
static void MPOST_CWEB_wrt_mp_rhs_elem(Cmpost_output * Fsm, std::string & Elem_name,
std::string & Drw_how);
static void MPOST_CWEB_gen_sr_elem_xrefs(Cmpost_output * Fsm, AST * Subrule_tree);
static void MPOST_CWEB_woutput_sr_sdcode(Cmpost_output * Fsm, T_subrule_def * Subrule_def);
static void MPOST_CWEB_wrt_fsm(Cmpost_output * Fsm, T_fsm_phrase * Fsm_phrase);
static void MPOST_CWEB_wrt_rule_s_lhs_sdc(Cmpost_output * Fsm, rule_def * Rule_def);
static bool MPOST_CWEB_should_subrule_be_printed(Cmpost_output * Fs, T_subrule_def * Subrule_def);
static void MPOST_CWEB_xref_referred_T(Cmpost_output * Fsm, referred_T * R_T);
static void MPOST_CWEB_xref_referred_rule(Cmpost_output * Fsm, referred_rule * R_rule);
static void MPOST_CWEB_LOAD_XLATE_CHRS(Cmpost_output * Fsm);
static void MPOST_CWEB_EMIT_PREFIX_CODE(Cmpost_output * Fsm);
static void MPOST_CWEB_wrt_T(Cmpost_output * Fsm, T_terminals_phrase * T_phrase);
static void MPOST_CWEB_wrt_Err(Cmpost_output * Fsm, T_error_symbols_phrase * Err_phrase);
static void MPOST_CWEB_wrt_Lrk(Cmpost_output * Fsm, T_lr1_k_phrase * Lrk_phrase);

std::list < NS_yacco2_terminals :: rule_def *> rules_for_fs_prt_;

static void MPOST_CWEB_crt_rhs_sym_str(state_element * se, std::string * Xlated_str);

```

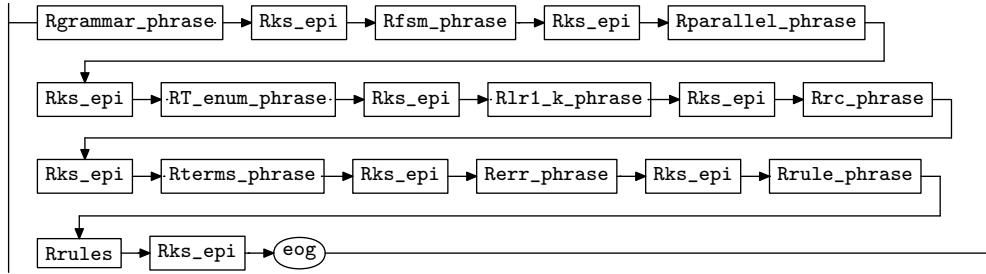
7. Cmpost_output user-prefix-declaration directive.

```
< Cmpost_output user-prefix-declaration directive 7 > ≡
#include "time.h"
#include "o2_externs.h"
extern void XLATE_SYMBOLS_FOR_cweave(const char *Sym_to_xlate, char *Xlated_sym);
extern void PRT_RULE_S_FIRST_SET(std::ofstream &Ofile, NS_yacco2_terminals::rule_def *Rule_def);
extern int MPOST_CWEB_xlated_symbol(AST *Sym, char *Xlated_sym);
extern STATES_typeLR1_STATES;
extern COMMON_LA_SETS_typeCOMMON_LA_SETS;
#include "cweave_fsm_sdc.h"
#include "cweave_lhs_sdc.h"
#include "cweave_sdc.h"
#include "cweave_T_sdc.h"
```

8. Rmpost_output rule.

Note how i use the sentence recognition to complete *mpost* figures. The individual figures are defined before their execution.

Rmpost_output



{ Rmpost_output subrule 1 op directive 8 } ≡

```

Cmpost_output * fsm = ( Cmpost_output * ) rule_info_.parser--fsm_tbl_;
KCHARP mp_init_vars = "init_variables;";
fsm->omp_file_ << mp_init_vars << endl;
KCHARP mp_figure = "beginfig(%i);\n" "drw_rule(%i);\n" "endfig;";
KCHARP sr_figure = "beginfig(%i);\n" "drw_rule_a_rhs(%i,%i);\n" "endfig;";
int rule_no = 1;
int fig_no(0);
for ( ; rule_no ≤ fsm->no_of_rules_; ++rule_no) {
    ++fig_no;
    sprintf(fsm->big_buf_, mp_figure, fig_no, rule_no);
    fsm->omp_file_ << fsm->big_buf_ << endl;
    int no_srs = fsm->no_subrules_per_rule_[rule_no];
    int sr_no(1);
    for ( ; sr_no ≤ no_srs; ++sr_no) {
        ++fig_no;
        sprintf(fsm->big_buf_, sr_figure, fig_no, rule_no, sr_no);
        fsm->omp_file_ << fsm->big_buf_ << endl;
    }
}
KCHARP mp_end = "end;";
fsm->omp_file_ << mp_end << endl;
fsm->omp_file_.close();
KCHARP w_fsc_file = "@**_First_Set_Language_for_\0linker.\fbreak\n" "\openin_1=\\"%s.f\sc\\n"
/* macro file open */
"\ifeof_1=true\n" /* test the result */
"\closein_1\n" "\else\n" "\closein_1\n" "\listing{\\"%s.fsc\"}\"\fi";
sprintf(fsm->big_buf_, w_fsc_file, fsm->fq_filename_noext_.c_str(), fsm->fq_filename_noext_.c_str());
fsm->ow_file_ << fsm->big_buf_ << endl;
KCHARP w_lr1_states = "@**_Lr1_State_Network.\fbreak\n" "\statecolumnspace_\\$";
sprintf(fsm->big_buf_, w_lr1_states, "\\");
fsm->ow_file_ << fsm->big_buf_ << endl;
KCHARP w_stateno = /* stateno, vectored into symbol, conflict */
"\stateno{\\$}\\{\\$}\\{\\$}\\n" "\halign{\\n" "\\span\\statetemplate\\n" "\\statetitle";
char vectored_into[Max_cweb_item_size];
  
```

```

char rule_name[Max_cweb_item_size];
char subrule_sym[Max_cweb_item_size];
char cur_sym[Max_cweb_item_size];
char nxtsym_1[Max_cweb_item_size];
char nxtsym_2[Max_cweb_item_size];
char state_type[Max_cweb_item_size];
char la_set_no[Max_cweb_item_size];

string w_possible_ar_code;
KCHARP la_set_no_template = "%i";
KCHARP state_type_template = "%s";
KCHARP three_symbols_string_template = "%s\u00d7%s\u00d7%s"; /* 3 symbols: current and 2 la */
KCHARP w_stateno_subrule = "{%s}\u00d7\n" /* closure or transitive */
"[%s]\u00d7\n" /* rule */
"[%i\\hss]\u00d7\n" /* rule no */
"[%i\\hss]\u00d7\n" /* subrule no */
"[%i\\hss]\u00d7\n" /* element pos */
"[%s]\u00d7\n" /* current symbol */
"[%i\\hss]\u00d7\n" /* born state no */
"[%i\\hss]\u00d7\n" /* goto state no */
"[%i\\hss]\u00d7\n" /* reduced state no */
"[%s\\cr"; /* la set if reduced */

int fnd_reduced_stateno(0);

STATES_ITER_type si = LR1_STATES.begin();
STATES_ITER_type sie = LR1_STATES.end();
for ( ; si != sie; ++si) { /* walk the states */
    vectored_into[0] = (char) 0;
    w_possible_ar_code.clear();
    state_type[0] = (char) 0;
    state * cur_state = *si;
    if (cur_state->state_no_ != 1) { /* state 1 vector into is empty */
        XLATE_SYMBOLS_FOR_cweave(cur_state->entry_symbol_literal(), vectored_into);
    }
    switch (cur_state->state_type_) {
        case 0:
        {
            /*shift only */
            sprintf(state_type, state_type_template, "\\Shiftonly");
            break;
        }
        case 1:
        {
            sprintf(state_type, state_type_template, "\\Reduceonly");
            break;
        }
        case 2:
        {
            sprintf(state_type, state_type_template, "\\ShiftReduce");
            break;
        }
        case 3:
        {
            sprintf(state_type, state_type_template, "\\MultipleReduces");
        }
    }
}

```

```

        break;
    }
case 4:
{
    sprintf(state_type, state_type_template, "\\ShiftandMultipleReduces");
    break;
}
w_possible_ar_code.append(vectored_into);
if (cur_state→vectored_into_by_elem_ ≡ LR1_PARALLEL_OPERATOR) {
    /* add arbitration code? to vectored_into string */
    if (cur_state→arbitrator_name_ ≠ 0) {
        w_possible_ar_code += "\\arbitration-code:\\|";
        w_possible_ar_code += cur_state→arbitrator_name→c_str();
        w_possible_ar_code += "|";
    }
    else {
        w_possible_ar_code += "\\arbitration-code:\\\\epsilon";
    }
}
sprintf(fsm→big_buf_, w_stateno, cur_state→state_no_, w_possible_ar_code.c_str() /*vectored_into */ ,
state_type);
fsm→ow_file_ << fsm→big_buf_ << endl;
S_VECTORS_ITER_type svi = cur_state→state_s_vector_.begin();
S_VECTORS_ITER_type svie = cur_state→state_s_vector_.end();
stringrhs_syms_str;
for ( ; svi ≠ svie; ++svi) {
    S_VECTOR_ELEMS_ITER_type seli = svi→second.begin();
    S_VECTOR_ELEMS_ITER_type selie = svi→second.end();
    for ( ; seli ≠ selie; ++seli) {
        rhs_syms_str.clear();
        state_element * se = *seli;
        rule_def * rd = se→subrule_def_its_rule_def();
        T_subrule_def * srd = se→subrule_def_;
        rule_name[0] = (char) 0;
        subrule_sym[0] = (char) 0;
        cur_sym[0] = (char) 0;
        la_set_no[0] = (char) 0;
        XLATE_SYMBOLS_FOR_cweave(rd→rule_name()→c_str(), rule_name);
        int elem_pos(0);
        if (se→previous_state_ ≡ 0) elem_pos = 1;
        elem_pos = MPOST_CWEB_xlated_symbol(se→sr_element_, cur_sym);
        if (se→goto_state_ ≡ 0) { /* reducing subrule */
            sprintf(la_set_no, la_set_no_template, se→common_la_set_idx_ + 1);
        }
        else { /* no la set */
        }
        if (se→reduced_state_ ≠ 0) {
            fnd_reduced_stateno = se→reduced_state→state_no_;
        }
        else { /* find from merged into */
            fnd_reduced_stateno = -1;
        }
    }
}
```

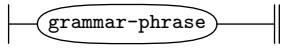
```

if (se->next_state_element->reduced_state_ ≠ 0) {
    fnd_reduced_stateno = se->next_state_element->reduced_state->state_no_;
}
}
fsm->MPOST_CWEB_crt_rhs_sym_str(se, &rhs_syms_str);
sprintf(fsm->big_buf_, w_stateno_subrule, (elem_pos ≡ 1) ? "c" : "t", rule_name, rd->rule_no(),
        srd->subrule_no_of_rule(), elem_pos, rhs_syms_str.c_str(), se->closure_state->state_no_,
        (se->goto_state_ ≠ 0) ? se->goto_state->state_no_ : 0, fnd_reduced_stateno, la_set_no);
fsm->ow_file_ << fsm->big_buf_ << endl;
}
}
fsm->ow_file_ << "}" << endl;
}
KCHARP w_end = "@**\u2022Index.";
fsm->ow_file_ << w_end << endl;
fsm->ow_file_.close();

```

9. Rgrammar_phrase rule.

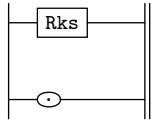
Rgrammar_phrase



10. Rks_epsilon rule.

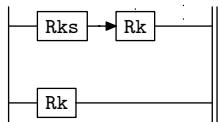
Due to technical problems with the *convertMPtoPDF* macro, the epsilon subrule is drawn with a dot.

Rks_epsilon



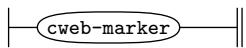
11. Rks rule.

Rks



12. Rk rule.

Rk

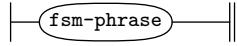


(Rk subrule 1 op directive 12) ≡

```

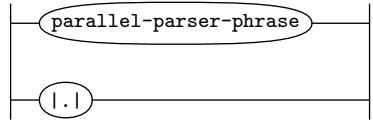
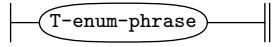
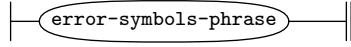
Cmpost_output * fsm = ( Cmpost_output * ) rule_info_.parser_->fsm_tbl_;
WRT_CWEB_MARKER(&fsm->ow_file_, sf->p1_->ast());

```

13. *Rfsm_phrase rule.***Rfsm_phrase**

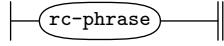
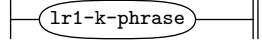
$\langle \text{Rfsm_phrase subrule 1 op directive 13} \rangle \equiv$

$Cmpost_output * fsm = (Cmpost_output *) rule_info...parser_>fsm_tbl_;$
 $fsm \rightarrow MPOST_CWEB_wrt_fsm(fsm, sf \rightarrow p1_);$

14. *Rparallel_phrase rule.***Rparallel_phrase****15. *RT_enum_phrase rule.*****RT_enum_phrase****16. *Rerr_phrase rule.*****Rerr_phrase**

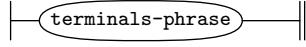
$\langle \text{Rerr_phrase subrule 1 op directive 16} \rangle \equiv$

$Cmpost_output * fsm = (Cmpost_output *) rule_info...parser_>fsm_tbl_;$
 $fsm \rightarrow MPOST_CWEB_wrt_Err(fsm, sf \rightarrow p1_);$

17. *Rrc_phrase rule.***Rrc_phrase****18. *Rlr1_k_phrase rule.*****Rlr1_k_phrase**

$\langle \text{Rlr1_k_phrase subrule 1 op directive 18} \rangle \equiv$

$Cmpost_output * fsm = (Cmpost_output *) rule_info...parser_>fsm_tbl_;$
 $fsm \rightarrow MPOST_CWEB_wrt_Lrk(fsm, sf \rightarrow p1_);$

19. *Rterms_phrase rule.***Rterms_phrase**

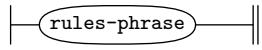
$\langle \text{Rterms_phrase subrule 1 op directive 19} \rangle \equiv$

$Cmpost_output * fsm = (Cmpost_output *) rule_info...parser_>fsm_tbl_;$
 $fsm \rightarrow MPOST_CWEB_wrt_T(fsm, sf \rightarrow p1_);$

20. *Rrule-phrase rule.*

Define the number of rules for *mpost*.

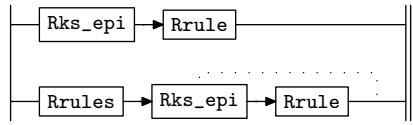
Rrule_phrase



$\langle \text{Rrule_phrase} \text{ subrule 1 op directive } 20 \rangle \equiv$
 $Cmpost_output * fsm = (Cmpost_output *) \text{rule_info_}.parser_fsm_tbl_;$
 $fsm\text{-no_of_rules_} = sf\text{-p1_rules_alphabet()}\text{-size}();$
 $KCHARP mp_no_of_rules = "no_of_rules_{\wedge}:=_{\wedge}i";$
 $sprintf(fsm\text{-big_buf_}, mp_no_of_rules, fsm\text{-no_of_rules_});$
 $fsm\text{-omp_file_} \ll fsm\text{-big_buf_} \ll endl;$

21. *Rrules rule.*

Rrules

**22. *Rrule rule.***

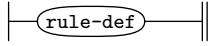
Rrule



23. Rrule_def rule.

Define rule for *mpost* and *cweb*. To help the grammar writer, walk the rule's subrules to generate a *cweb* cross-reference per subrule element. Originally this was done at the subrule level to be specific to the subrule entry instead of at the overall rule level. This meant that each subrule had to be redrawn with a *cweb* directive even when there was no syntax directed code present. Just ugly is my take so i follow Occam's dictum.

Rrule_def



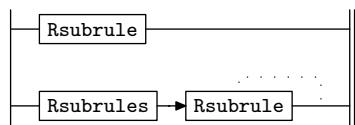
```

⟨ Rrule_def subrule 1 op directive 23 ⟩ ≡
  Cmpost_output * fsm = ( Cmpost_output * ) rule_info__.parser__->fsm_tbl__;
  fsm->rule_def_ = sf->p1_;
  fsm->rules_for_fs_prt_.push_back(fsm->rule_def_);
  KCHARPrname = fsm->rule_def->rule_name()~c_str();
  ++fsm->w_fig_no_;
  ++fsm->rule_no_;
  fsm->subrule_no_ = 0;
  KCHARP mp_rule_names = "rule_names[%i].literal":="%s";"rule_names[%i].vnam\
e":="%s";";
  sprintf(fsm->big_buf_, mp_rule_names, fsm->rule_no_, rname, fsm->rule_no_, rname);
  fsm->omp_file_ << fsm->big_buf_ << endl;
  T_subrules_phrase * sr_ph = fsm->rule_def->subrules();
  fsm->no_of_subrules_ = sr_ph->no_subrules();
  KCHARP mp_rule_s_no_rhs = "rule_s_no_rhs[%i]":=%i;";
  sprintf(fsm->big_buf_, mp_rule_s_no_rhs, fsm->rule_no_, fsm->no_of_subrules_);
  fsm->omp_file_ << fsm->big_buf_ << endl;
  fsm->no_subrules_per_rule_.push_back(fsm->no_of_subrules_); /* [rule #] */
  KCHARP rule_cweb = "@*2|@!%s|rule.\\"fbreak\n";
  sprintf(fsm->big_buf_, rule_cweb, rname);
  fsm->ow_file_ << fsm->big_buf_;
  if (fsm->rule_def->cweb_marker() ≠ 0) {
    WRT_CWEB_MARKER(&fsm->ow_file_, fsm->rule_def->cweb_marker());
  }
  KCHARP rule_cweb_diagram = "\\"fbreak\n"\"convertMPtoPDF{%.s.%i}{1}{1}\n";
  sprintf(fsm->big_buf_, rule_cweb_diagram, fsm->grammar_filename_prefix_.c_str(), fsm->w_fig_no_);
  fsm->ow_file_ << fsm->big_buf_;
  std::vector < T_subrule_def *> *srules = sr_ph->subrules();
  std::vector < T_subrule_def *> :: iterator i = srules->begin();
  std::vector < T_subrule_def *> :: iterator ie = srules->end();
  for ( ; i ≠ ie; ++i) {
    T_subrule_def * srd = *i;
    fsm->MPOST_CWEB_gen_sr_elem_xrefs(fsm, srd->subrule_s_tree());
  }
  fsm->MPOST_CWEB_wrt_rule_s_lhs_sdc(fsm, fsm->rule_def_);

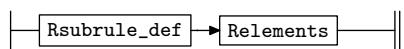
```

24. *Rsubrules rule.*

Rsubrules

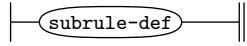
**25.** *Rsubrule rule.*

Rsubrule



26. Rsubrule_def rule.

Rsubrule_def



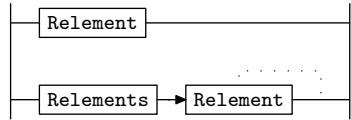
The *eosubrule* T has no meaning in the drawn subrule. It is just a specific end-of-subrule indicator. So it is removed from the rhs element count of the subrule for *mpost* processing so that it does not get drawn. The balance of the logic relates to *cweb*'s document: comments about the subrule and the appropriate *mpost* rendering.

One subtlety is the forced drawing of a single subrule that also has a rule's lhs syntax directed code. Normally I save the redrawing of the subrule as the forest and the tree is the same. But having the rule syntax code requires to have the subrule directive. Without it there are 2 code directives back-to-back that honks.

```
(Rsubrule_def subrule 1 op directive 26) ≡
Cmpost_output * fsm = ( Cmpost_output * ) rule_info...parser→fsm_tbl→;
++fsm→w_fig_no_;
++fsm→subrule_no_;
fsm→elem_no_ = 0;
fsm→subrule_def_ = sf→p1→;
int no_elements = fsm→subrule_def→no_of_elems() - 1;
if (no_elements ≡ 0) no_elements = 1; /* epsilon subrule */
KCHARP mp_rule_s_subrule_no_elems = "rule_s_subrule_no_elems[%i][%i]:=↓%i";
sprintf(fsm→big_buf_, mp_rule_s_subrule_no_elems, fsm→rule_no_, fsm→subrule_no_, no_elements);
fsm→omp_file_ << fsm→big_buf_ << endl;
if (fsm→MPOST_CWEB_should_subrule_be_printed(fsm, fsm→subrule_def_) ≡ false) return;
KCHARP rname = fsm→rule_def→rule_name()→c_str();
KCHARP subrule_cweb = "@*3↓|%s↓'s↓subrule↓%i.\\"fbreak\n";
if ((fsm→rule_def→rule_lhs() ≠ 0) ∨ (fsm→no_of_subrules_ > 1)) {
    sprintf(fsm→big_buf_, subrule_cweb, rname, fsm→subrule_no_);
    fsm→ow_file_ << fsm→big_buf_;
}
if (fsm→subrule_def→cweb_marker() ≠ 0) {
    WRT_CWEB_MARKER(&fsm→ow_file_, fsm→subrule_def→cweb_marker());
    cout << "subrule's↓cweb↓marker↓for↓rule:↓" << rname << endl;
}
else {
    cout << "no↓subrule's↓cweb↓marker↓for↓rule:↓" << rname << endl;
}
KCHARP subrule_cweb_diagram = "\\"fbreak"\\"convertMPtoPDF{%s.%i}{1}{1}\n";
if ((fsm→rule_def→rule_lhs() ≠ 0) ∨ (fsm→no_of_subrules_ > 1)) {
    sprintf(fsm→big_buf_, subrule_cweb_diagram, fsm→grammar_filename_prefix_.c_str(), fsm→w_fig_no_);
    fsm→ow_file_ << fsm→big_buf_;
}
fsm→MPOST_CWEB_woutput_sr_sdcode(fsm, fsm→subrule_def_);
```

27. Relements rule.

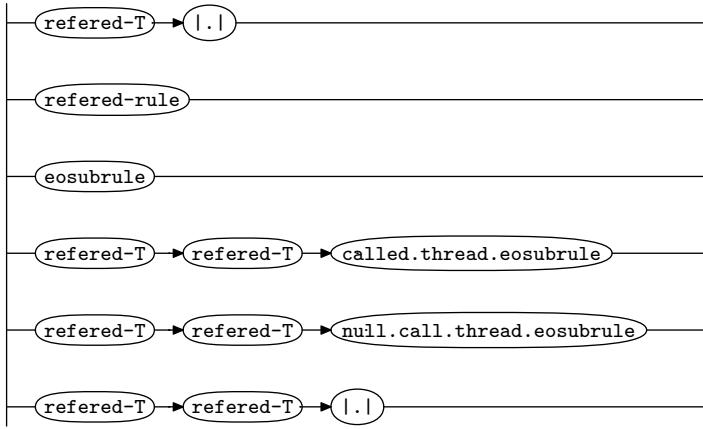
Relements



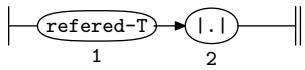
28. Relement rule.

Use of `| . |` to make grammar lr(1).

Relement



29. Relement's subrule 1.



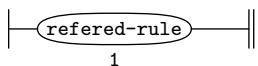
Why is there a test for `\"` within the T? Glad u asked as *mpost* has problems dealing with this legitimate escape sequence and the double quote sequence within the c string. So this was my 1st attempt to silence the honk. Unfortunately the `char(34)` *mpost* operator does not convert the 34 into the `"` character. For the moment i will leave this as is. U will see other attempts like this throughout the grammar.

$\langle \text{Relement subrule 1 op directive 29} \rangle \equiv$

```

Cmpost_output * fsm = ( Cmpost_output * ) rule_info__.parser__->fsm_tbl__;
++fsm->elem_no_;
std :: string mp_obj_type("Circle_solid");
std :: string mp_elem_name(sf->p1__->t_in_stbl()->t_def()->t_name()->c_str());
string :: size_type x = mp_elem_name.find("\\\\\"");
if (x != string :: npos) {
    mp_elem_name.clear();
    mp_elem_name += "char(34)";
}
fsm->MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
    
```

30. Relement's subrule 2.



$\langle \text{Relement subrule 2 op directive 30} \rangle \equiv$

```

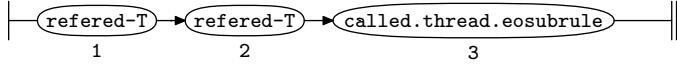
Cmpost_output * fsm = ( Cmpost_output * ) rule_info__.parser__->fsm_tbl__;
++fsm->elem_no_;
std :: string mp_obj_type("Box_solid");
std :: string mp_elem_name(sf->p1__->Rule_in_stbl()->r_def()->rule_name()->c_str());
fsm->MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
    
```

31. Relement's subrule 3.

|—**(eosubrule)**—||
 1

⟨ Relement subrule 3 op directive 31 ⟩ ≡
 $Cmpost_output * fsm = (Cmpost_output *) rule_info_..parser_..fsm_tbl_..;$
 $\quad\quad\quad fsm\rightarrow elem_no_;$
if ($fsm\rightarrow elem_no_ \equiv 1$) { /* epsilon */
 $\quad\quad\quad std::string mp_obj_type("Circle_solid");$
 $\quad\quad\quad std::string mp_elem_name("□");$
 $\quad\quad\quad fsm\rightarrow MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);$
}

32. Relement's subrule 4.



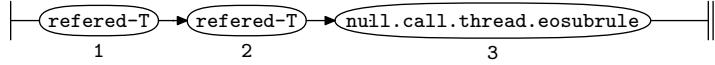
Why is there a re-aligning on the number of *rule_s_subrule_no_elems*? Two reasons: Due to the tree walk being top down, initially the number of items are determined by the *subrule - def T* before processing the actual subrule sentence. Due to the calling of threads, this number gets shrunk as i combine the 3 parts of the called thread into 1 item. Though it looks like a mistake in the emitted *mpost* code, it's a slight inefficiency to manage a nice picture.

This comment hold for the other version of thread call below.

```

⟨ Relement subrule 4 op directive 32 ⟩ ≡
Cmpost_output * fsm = ( Cmpost_output * ) rule_info...parser→fsm_tbl_;
KCHARP re_align_nos = "rule_s_subrule_no_elems[%i][%i]:=_%i;";
sprintf(fsm→big_buf_, re_align_nos, fsm→rule_no_, fsm→subrule_no_, 3);
fsm→omp_file_ << fsm→big_buf_ << "%_re-align_cnt" << endl;
++fsm→elem_no_;
std::string mp_obj_type("Circle_dotted");
std::string mp_elem_name;
if (sf→p1→its_t_def()→enum_id() ≡ T_Enum :: T_LR1_parallel_operator_) {
    mp_elem_name += "|||";
}
else {
    mp_elem_name += "|t|";
}
fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
++fsm→elem_no_;
mp_obj_type.clear();
mp_obj_type += "Circle_dotted";
mp_elem_name.clear();
mp_elem_name += sf→p2→t_in_stbl()→t_def()→t_name()→c_str();
string::size_type x = mp_elem_name.find("\\\\\"");
if (x ≠ string::npos) {
    mp_elem_name.erase();
    mp_elem_name += "char(34)";
}
fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
++fsm→elem_no_;
mp_elem_name.clear();
mp_obj_type.clear();
mp_obj_type += "Box_dotted";
mp_elem_name += sf→p3→ns()→identifier()→c_str();
mp_elem_name += "::";
mp_elem_name += sf→p3→called_thread_name()→identifier()→c_str();
fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
  
```

33. Relement's subrule 5.

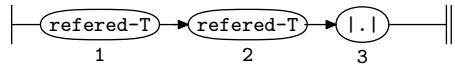


```

⟨ Relement subrule 5 op directive 33 ⟩ ≡
  Cmpost_output * fsm = ( Cmpost_output * ) rule_info...parser→fsm_tbl_;
  KCHARPre_align_nos = "rule_s_subrule_no_elems[%i] [%i]:=_%i;";
  sprintf(fsm→big_buf_, re_align_nos, fsm→rule_no_, fsm→subrule_no_, 3);
  fsm→omp_file_ << fsm→big_buf_ << "%_re-align_cnt" << endl;
  ++fsm→elem_no_;
  std::string mp_obj_type("Circle_dotted");
  std::string mp_elem_name;
  if (sf→p1→its_t_def()→enum_id() ≡ T_Enum::T_LR1_parallel_operator_) {
    mp_elem_name += "|||";
  }
  else {
    mp_elem_name += "|t|";
  }
  fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
  mp_elem_name.clear();
  ++fsm→elem_no_;
  mp_elem_name += sf→p2→t_in_stbl()→t_def()→t_name()→c_str();
  string::size_type x = mp_elem_name.find("\\\\\"");
  if (x ≠ string::npos) {
    mp_elem_name.clear();
    mp_elem_name += "char(34)";
  }
  fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);
  ++fsm→elem_no_;
  mp_elem_name.clear();
  mp_obj_type.clear();
  mp_obj_type += "Box_dotted";
  mp_elem_name += "NULL";
  fsm→MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);

```

34. Relement's subrule 6.



$\langle \text{Relement subrule 6 op directive 34} \rangle \equiv$
 $Cmpost_output * fsm = (Cmpost_output *) rule_info_parser_fsm_tbl_;$
 $\quad \quad \quad ++fsm\rightarrow elem_no_;$
 $\quad std::string mp_obj_type("Circle_solid");$
 $\quad std::string mp_elem_name(sf\rightarrow p1\rightarrow t_in_stbl()\rightarrow t_def()\rightarrow t_name()\rightarrow c_str());$
 $\quad string::size_typex = mp_elem_name.find("\\\\");$
 $\quad \text{if } (x \neq string::npos) \{$
 $\quad \quad mp_elem_name.erase();$
 $\quad \quad mp_elem_name += "char(34)";$
 $\quad \}$
 $\quad fsm\rightarrow MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);$
 $\quad \quad \quad ++fsm\rightarrow elem_no_;$
 $\quad mp_obj_type.clear();$
 $\quad mp_obj_type += "Circle_solid";$
 $\quad mp_elem_name.clear();$
 $\quad mp_elem_name += sf\rightarrow p2\rightarrow t_in_stbl()\rightarrow t_def()\rightarrow t_name()\rightarrow c_str();$
 $\quad x = mp_elem_name.find("\\\\");$
 $\quad \text{if } (x \neq string::npos) \{$
 $\quad \quad mp_elem_name.erase();$
 $\quad \quad mp_elem_name += "char(34)";$
 $\quad \}$
 $\quad fsm\rightarrow MPOST_CWEB_wrt_mp_rhs_elem(fsm, mp_elem_name, mp_obj_type);$

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

```
/*
File: mpost_output.fsc
Date and Time: Fri Jan  2 15:33:44 2015
*/
transitive    n
grammar-name  "mpost_output"
name-space    "NS_mpost_output"
thread-name   "Cmpost_output"
monolithic    y
file-name     "mpost_output.fsc"
no-of-T       569
list-of-native-first-set-terminals 1
  T_grammar_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
"Output grammar rules railroad diagrams for mpost that cweb program uses."
```

36. Lr1 State Network.

					State: 1 state type: s								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rgrammar_phrase			2 1 1	grammar-phrase					1 2 2				
c Rmpost_output			1 1 1	Rgrammar_phrase <u>Rks.epiϵ Rfsm_phrase</u>					1 3 22				
\Rightarrow grammar-phrase					State: 2 state type: r								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
t Rgrammar_phrase			2 1 2						1 0 2 1				
\Rightarrow Rgrammar_phrase					State: 3 state type: s/r								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rks.epi			3 2 1	ϵ					3 0 3 2				
c Rk			5 1 1	cweb-marker					3 42 42				
t Rmpost_output			1 1 2	Rks.epi <u>Rfsm_phrase</u>					1 4 22				
c Rks			4 1 1	Rks <u>Rk</u>					3 43 44				
c Rks.epi			3 1 1	Rks					3 43 43				
c Rks			4 2 1	Rk					3 45 45				
\Rightarrow Rks.epi					State: 4 state type: s								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rfsm_phrase			6 1 1	fsm-phrase					4 46 46				
t Rmpost_output			1 1 3	Rfsm_phrase <u>Rks.epiϵ Rparallel_phrase</u>					1 5 22				
\Rightarrow Rfsm_phrase					State: 5 state type: s/r								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rks.epi			3 2 1	ϵ					5 0 5 3				
c Rk			5 1 1	cweb-marker					5 42 42				
t Rmpost_output			1 1 4	Rks.epi <u>Rparallel_phrase</u>					1 6 22				
c Rks			4 1 1	Rks <u>Rk</u>					5 43 44				
c Rks.epi			3 1 1	Rks					5 43 43				
c Rks			4 2 1	Rk					5 45 45				
\Rightarrow Rks.epi					State: 6 state type: s								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rparallel_phrase			7 2 1						6 47 47				
c Rparallel_phrase			7 1 1	parallel-parser-phrase					6 48 48				
t Rmpost_output			1 1 5	Rparallel_phrase <u>Rks.epiϵ RT_enum_phrase</u>					1 7 22				
\Rightarrow Rparallel_phrase					State: 7 state type: s/r								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c Rks.epi			3 2 1	ϵ					7 0 7 4				
c Rk			5 1 1	cweb-marker					7 42 42				
t Rmpost_output			1 1 6	Rks.epi <u>RT_enum_phrase</u>					1 8 22				
c Rks			4 1 1	Rks <u>Rk</u>					7 43 44				
c Rks.epi			3 1 1	Rks					7 43 43				
c Rks			4 2 1	Rk					7 45 45				
\Rightarrow Rks.epi					State: 8 state type: s								
\Rightarrow		← rule	→ R# sr# Po	← subrule element					→ Brn Gto Red LA				
c RT_enum_phrase			8 1 1	T-enum-phrase					8 49 49				

t Rmpost_output	1 1 7	RT_enum_phrase <u>Rks.epi</u> ^ε <u>Rlr1.k.phrase</u>	1 9 22
$\Rightarrow^{RT_enum_phrase}$			
← rule	→ R# sr# Po	← subrule element	State: 9 state type: ^{s/r}
c Rks.epi	3 2 1	ε	→ Brn Gto Red LA 9 0 9 5
c Rk	5 1 1	cweb-marker	9 42 42
t Rmpost_output	1 1 8	Rks.epi <u>Rlr1.k.phrase</u>	1 10 22
c Rks	4 1 1	Rks <u>Rk</u>	9 43 44
c Rks.epi	3 1 1	Rks	9 43 43
c Rks	4 2 1	Rk	9 45 45
$\Rightarrow^{Rks.epi}$			
← rule	→ R# sr# Po	← subrule element	State: 10 state type: ^s
c Rlr1.k.phrase	11 1 1	lr1-k-phrase	→ Brn Gto Red LA 10 50 50
t Rmpost_output	1 1 9	Rlr1.k.phrase <u>Rks.epi</u> ^ε <u>Rrc.phrase</u>	1 11 22
$\Rightarrow^{Rrc.phrase}$			
← rule	→ R# sr# Po	← subrule element	State: 11 state type: ^{s/r}
c Rks.epi	3 2 1	ε	→ Brn Gto Red LA 11 0 11 6
c Rk	5 1 1	cweb-marker	11 42 42
t Rmpost_output	1 1 10	Rks.epi <u>Rrc.phrase</u>	1 12 22
c Rks	4 1 1	Rks <u>Rk</u>	11 43 44
c Rks.epi	3 1 1	Rks	11 43 43
c Rks	4 2 1	Rk	11 45 45
$\Rightarrow^{Rks.epi}$			
← rule	→ R# sr# Po	← subrule element	State: 12 state type: ^s
c Rrc.phrase	10 1 1	rc-phrase	→ Brn Gto Red LA 12 51 51
t Rmpost_output	1 1 11	Rrc.phrase <u>Rks.epi</u> ^ε <u>Rterms.phrase</u>	1 13 22
$\Rightarrow^{Rterms.phrase}$			
← rule	→ R# sr# Po	← subrule element	State: 13 state type: ^{s/r}
c Rks.epi	3 2 1	ε	→ Brn Gto Red LA 13 0 13 7
c Rk	5 1 1	cweb-marker	13 42 42
t Rmpost_output	1 1 12	Rks.epi <u>Rterms.phrase</u>	1 14 22
c Rks	4 1 1	Rks <u>Rk</u>	13 43 44
c Rks.epi	3 1 1	Rks	13 43 43
c Rks	4 2 1	Rk	13 45 45
$\Rightarrow^{Rks.epi}$			
← rule	→ R# sr# Po	← subrule element	State: 14 state type: ^s
c Rterms.phrase	12 1 1	terminals-phrase	→ Brn Gto Red LA 14 52 52
t Rmpost_output	1 1 13	Rterms.phrase <u>Rks.epi</u> ^ε <u>Rerr.phrase</u>	1 15 22
$\Rightarrow^{Rerr.phrase}$			
← rule	→ R# sr# Po	← subrule element	State: 15 state type: ^{s/r}
c Rks.epi	3 2 1	ε	→ Brn Gto Red LA 15 0 15 8
c Rk	5 1 1	cweb-marker	15 42 42
t Rmpost_output	1 1 14	Rks.epi <u>Rerr.phrase</u>	1 16 22
c Rks	4 1 1	Rks <u>Rk</u>	15 43 44
c Rks.epi	3 1 1	Rks	15 43 43
c Rks	4 2 1	Rk	15 45 45

\Rightarrow^{Rks_epi}			State: 16 state type: s	
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		
c Rerr_phrase	9 1 1	error-symbols-phrase		→ Brn Gto Red LA
t Rmpost_output	1 1 15	Rerr_phrase <u>$Rks_epi^\epsilon Rrule_phrase$</u>	16 53 53	
			1 17 22	
$\Rightarrow^{Rerr_phrase}$		State: 17 state type: s/r		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
c Rks_епи	3 2 1	ϵ	17 0 17 9	
c Rk	5 1 1	cweb-marker	17 42 42	
t Rmpost_output	1 1 16	Rks_епи <u>$Rrule_phrase$</u>	1 18 22	
c Rks	4 1 1	Rks <u>Rk</u>	17 43 44	
c Rks_епи	3 1 1	Rks	17 43 43	
c Rks	4 2 1	Rk	17 45 45	
\Rightarrow^{Rks_epi}		State: 18 state type: s		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
c Rrule_phrase	13 1 1	rules-phrase	18 54 54	
t Rmpost_output	1 1 17	Rrule_phrase <u>$Rrules$</u>	1 19 22	
$\Rightarrow^{Rrule_phrase}$		State: 19 state type: s/r		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
c Rks_епи	3 2 1	ϵ	19 0 19 10	
c Rk	5 1 1	cweb-marker	19 42 42	
c Rrules	14 1 1	Rks_епи <u>$Rrule$</u>	19 55 56	
c Rks	4 1 1	Rks <u>Rk</u>	19 43 44	
c Rks_епи	3 1 1	Rks	19 43 43	
c Rks	4 2 1	Rk	19 45 45	
c Rrules	14 2 1	Rrules <u>$Rks_epi^\epsilon Rrule$</u>	19 20 24	
t Rmpost_output	1 1 18	Rrules <u>Rks_epi^ϵ</u> <u>eog</u>	1 20 22	
\Rightarrow^{Rrules}		State: 20 state type: s/r		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
c Rks_епи	3 2 1	ϵ	20 0 20 11	
c Rk	5 1 1	cweb-marker	20 42 42	
t Rrules	14 2 2	Rks_епи <u>$Rrule$</u>	19 21 24	
t Rmpost_output	1 1 19	Rks_епи <u>eog</u>	1 21 22	
c Rks	4 1 1	Rks <u>Rk</u>	20 43 44	
c Rks_епи	3 1 1	Rks	20 43 43	
c Rks	4 2 1	Rk	20 45 45	
\Rightarrow^{Rks_epi}		State: 21 state type: s		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
t Rmpost_output	1 1 20	eog	1 22 22	
c Rrule_def	16 1 1	rule-def	21 23 23	
t Rrules	14 2 3	Rrule	19 24 24	
c Rrule	15 1 1	Rrule_def <u>$Rsubrules$</u>	21 25 27	
\Rightarrow^{eog}		State: 22 state type: r		
\leftarrow rule	\rightarrow R# sr# Po \leftarrow	subrule element		→ Brn Gto Red LA
t Rmpost_output	1 1 21		1 0 22 12	

$\Rightarrow^{rule-def}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 23 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rrule_def	16 1 2		21 0 23 13
\Rightarrow^{Rrule}	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 24 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rrules	14 2 4		19 0 24 14
\Rightarrow^{Rrule_def}	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 25 state type: s subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
c Rsubrule_def	19 1 1	subrule-def		25 26 26
t Rrule	15 1 2	Rsubrules		21 27 27
c Rsubrules	17 2 1	Rsubrules <u>Rsubrule</u>		25 27 28
c Rsubrules	17 1 1	Rsubrule		25 41 41
c Rsubrule	18 1 1	Rsubrule_def <u>Relements</u>		25 29 38
$\Rightarrow^{subrule-def}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 26 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
t Rsubrule_def	19 1 2			25 0 26 15
$\Rightarrow^{Rsubrules}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 27 state type: s/r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
t Rrule	15 1 3			21 0 27 14
c Rsubrule_def	19 1 1	subrule-def		27 26 26
t Rsubrules	17 2 2	Rsubrule		25 28 28
c Rsubrule	18 1 1	Rsubrule_def <u>Relements</u>		27 29 38
$\Rightarrow^{Rsubrule}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 28 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
t Rsubrules	17 2 3			25 0 28 16
$\Rightarrow^{Rsubrule_def}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 29 state type: s subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
c Relement	21 4 1	refered-T		29 30 34
c Relement	21 5 1	refered-T		29 30 35
c Relement	21 6 1	refered-T		29 30 33
c Relement	21 1 1	refered-T		29 30 31
c Relement	21 2 1	refered-rule		29 36 36
c Relement	21 3 1	eosubrule		29 37 37
t Rsubrule	18 1 2	Relements		27 38 38
c Relements	20 2 1	Relements <u>Relement</u>		29 38 39
c Relements	20 1 1	Relement		29 40 40
$\Rightarrow^{refered-T}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 30 state type: s subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
t Relement	21 1 2	.		29 31 31
t Relement	21 4 2	refered-T		29 32 34
t Relement	21 5 2	refered-T		29 32 35
t Relement	21 6 2	refered-T		29 32 33
$\Rightarrow . $			State: 31 state type: r	

\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA	
t Rlement			21	1	3				29	0	31	17	
$\Rightarrow^{referred-T}$							State: 32 state type: ^s						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	6	3	.1.			29	33	33		
t Rlement			21	4	3	called thread eosubrule			29	34	34		
t Rlement			21	5	3	null call thread eosubrule			29	35	35		
$\Rightarrow^{.1.}$							State: 33 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	6	4				29	0	33	17	
$\Rightarrow^{calledthreadeosubrule}$							State: 34 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	4	4				29	0	34	17	
$\Rightarrow^{nullcallthreadeosubrule}$							State: 35 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	5	4				29	0	35	17	
$\Rightarrow^{referred-rule}$							State: 36 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	2	2				29	0	36	17	
$\Rightarrow^{eosubrule}$							State: 37 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rlement			21	3	2				29	0	37	17	
$\Rightarrow^{Relements}$							State: 38 state type: ^{s/r}						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rsubrule			18	1	3				27	0	38	16	
c Rlement			21	4	1	referred-T			38	30	34		
c Rlement			21	5	1	referred-T			38	30	35		
c Rlement			21	6	1	referred-T			38	30	33		
c Rlement			21	1	1	referred-T			38	30	31		
c Rlement			21	2	1	referred-rule			38	36	36		
c Rlement			21	3	1	eosubrule			38	37	37		
t Rlements			20	2	2	Rlement			29	39	39		
$\Rightarrow^{Relement}$							State: 39 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Relements			20	2	3				29	0	39	17	
$\Rightarrow^{Relement}$							State: 40 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Relements			20	1	2				29	0	40	17	
$\Rightarrow^{Rsubrule}$							State: 41 state type: ^r						
	\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t Rsubrules			17	1	2				25	0	41	16	

$\Rightarrow^{cweb-marker}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 42 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rk	5 1 2		3 0 42 18
\Rightarrow^{Rks}	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 43 state type: s/r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rks.epi	3 1 2		3 0 43 2
	c Rk	5 1 1	cweb-marker	43 42 42
	t Rks	4 1 2	Rk	3 44 44
\Rightarrow^{Rk}	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 44 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rks	4 1 3		3 0 44 18
\Rightarrow^{Rk}	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 45 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rks	4 2 2		3 0 45 18
$\Rightarrow^{fsm-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 46 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rfsm_phrase	6 1 2		4 0 46 19
$\Rightarrow^{ . }$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 47 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rparallel_phrase	7 2 2		6 0 47 20
$\Rightarrow^{parallel-parser-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 48 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rparallel_phrase	7 1 2		6 0 48 20
$\Rightarrow^{T-enum-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 49 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t RT_enum_phrase	8 1 2		8 0 49 21
$\Rightarrow^{lr1-k-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 50 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rlr1_k_phrase	11 1 2		10 0 50 22
$\Rightarrow^{rc-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 51 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rrc_phrase	10 1 2		12 0 51 23
$\Rightarrow^{terminals-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 52 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rterms_phrase	12 1 2		14 0 52 24
$\Rightarrow^{error-symbols-phrase}$	$\leftarrow \text{rule}$	$\rightarrow R\# \ sr\# \ Po \leftarrow$	State: 53 state type: r subrule element	$\rightarrow \text{Brn} \ \text{Gto} \ \text{Red} \ \text{LA}$
	t Rerr_phrase	9 1 2		16 0 53 25
$\Rightarrow^{rules-phrase}$			State: 54 state type: r	

\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t	Rrule_phrase		13	1	2				18	0	54	26
\Rightarrow^{Rks_epi}							State: 55 state type: ^s					
\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
c	Rrule_def		16	1	1		rule-def		55	23	23	
t	Rrules		14	1	2		Rrule		19	56	56	
c	Rrule		15	1	1		Rrule_def <u>Rsubrules</u>		55	25	27	
\Rightarrow^{Rrule}							State: 56 state type: ^r					
\leftarrow	rule	\rightarrow	R#	sr#	Po	\leftarrow	subrule element	\rightarrow	Brn	Gto	Red	LA
t	Rrules		14	1	3				19	0	56	14

37. Index.

ϵ : 10.
 $| . |$: 14, 28.
 $--$: 2.
 $--FILE--$: 5.
 $--LINE--$: 5.
`add_token_to_error_queue`: 5.
`append`: 8.
`arbitrator_name_`: 8.
`AST`: 6, 7.
`ast`: 12.
`begin`: 8, 23.
`big_buf_`: 6, 8, 20, 23, 26, 32, 33.
`BIG_BUFFER_32K`: 6.
`c_str`: 5, 8, 23, 26, 29, 30, 32, 33, 34.
`CAbs_lr1_sym`: 5.
called thread eosubrule: 28.
`called_thread_name`: 32.
`clear`: 8, 29, 32, 33, 34.
`close`: 8.
`closure_state_`: 8.
`Cmpost_output`: 6, 8, 12, 13, 16, 18, 19, 20, 23, 26, 29, 30, 31, 32, 33, 34.
`common_la_set_idx`: 8.
`COMMON_LA_SETS`: 7.
`COMMON_LA_SETS_type`: 7.
`convertMPtoPDF`: 10.
`cout`: 26.
`cTime`: 5.
`ctime`: 5.
`cur_state`: 8.
`cur_sym`: 8.
`cweave`: 2.
`cweb`: 2, 23, 26.
`cweb-marker`: 12.
`cweb_marker`: 23, 26.
`def`: 32.
`Dimension`: 6.
`Drw_how`: 6.
`Elem_name`: 6.
`elem_name_`: 6.
`Elem_no`: 6.
`elem_no_`: 5, 6, 26, 29, 30, 31, 32, 33, 34.
`elem_pos`: 8.
`end`: 8, 23.
`endl`: 8, 20, 23, 26, 32, 33.
`entry_symbol_literal`: 8.
`enum_id`: 32, 33.
`eog`: 8.
eosubrule: 28.
`eosubrule`: 26.
`erase`: 32, 34.
`Err_bad_filename`: 5.
`err_file`: 5.
`Err_phrase`: 6.
`error-symbols-phrase`: 16.
`false`: 26.
`fig_no`: 8.
`find`: 5, 29, 32, 33, 34.
`fnd_reduced_stateno`: 8.
`fq_filename_noext`: 6, 8.
`Fs`: 6.
`Fsm`: 6.
`fsm`: 8, 12, 13, 16, 18, 19, 20, 23, 26, 29, 30, 31, 32, 33, 34.
`fsm-phrase`: 13.
`Fsm_phrase`: 6.
`fsm_tbl_`: 8, 12, 13, 16, 18, 19, 20, 23, 26, 29, 30, 31, 32, 33, 34.
`gened_date_time_`: 5, 6.
`goto_state_`: 8.
`grammar-phrase`: 9.
`grammar-filename-prefix_`: 5, 6, 23, 26.
`identifier`: 32.
`ie`: 23.
`ios`: 5.
`ios_base`: 5.
`iterator`: 23.
`its_rule_def`: 8.
`its_t_def`: 32, 33.
`KCHARP`: 5, 8, 20, 23, 26, 32, 33.
`la_set_no`: 8.
`la_set_no_template`: 8.
`list`: 6.
`lrk_file`: 5.
`Lrk_phrase`: 6.
`lr1-k-phrase`: 18.
`LR1_PARALLEL_OPERATOR`: 8.
`LR1_STATES`: 7, 8.
`map`: 6.
`Max_cweb_item_size`: 8.
`mp_dimension_`: 5, 6.
`mp_elem_name`: 29, 30, 31, 32, 33, 34.
`mp_end`: 8.
`mp_figure`: 8.
`mp_filename_`: 5, 6.
`mp_init_vars`: 8.
`mp_no_of_rules`: 20.
`Mp_obj_name`: 6.
`mp_obj_type`: 29, 30, 31, 32, 33, 34.
`mp_rule_names`: 23.
`mp_rule_s_no_rhs`: 23.
`mp_rule_s_subrule_no_elems`: 26.

mpost: 2, 8, 20, 23, 26, 29, 32.
MPOST_CWEB_calc_mp_obj_name: 6.
MPOST_CWEB_crt_rhs_sym_str: 6, 8.
MPOST_CWEB_EMIT_PREFIX_CODE: 5, 6.
MPOST_CWEB_gen_dimension_name: 6.
MPOST_CWEB_gen_sr_elem_xrefs: 6, 23.
MPOST_CWEB_LOAD_XLATE_CHRS: 4, 6.
MPOST_CWEB_should_subrule_be_printed: 6, 26.
MPOST_CWEB_woutput_sr_sdcode: 6, 26.
MPOST_CWEB_wrt_Err: 6, 16.
MPOST_CWEB_wrt_fsm: 6, 13.
MPOST_CWEB_wrt_Lrk: 6, 18.
MPOST_CWEB_wrt_mp_rhs_elem: 6, 29, 30, 31, 32, 33, 34.
MPOST_CWEB_wrt_rule_s_lhs_sdc: 6, 23.
MPOST_CWEB_wrt_T: 6, 19.
MPOST_CWEB_xlated_symbol: 7, 8.
MPOST_CWEB_xref_referred_rule: 6.
MPOST_CWEB_xref_referred_T: 6.
mpost_output: 2.
n: 5.
next_state_element_: 8.
no_elements: 26.
no_of_elems: 26.
no_of_rules_: 5, 6, 8, 20.
no_of_subrules_: 5, 6, 23, 26.
no_srs: 8.
no_subrules: 23.
no_subrules_per_rule_: 5, 6, 8, 23.
npos: 29, 32, 33, 34.
ns: 32.
NS_yacc2_terminals: 6, 7.
null call thread eosubrule: 28.
nxtsym_1: 8.
nxtsym_2: 8.
Ofile: 7.
ofstream: 6, 7.
omp_file_: 5, 6, 8, 20, 23, 26, 32, 33.
open: 5.
out: 5.
ow_err_file_: 5, 6.
ow_file_: 5, 6, 8, 12, 23, 26.
ow_index_file_: 5, 6.
ow_lrk_file_: 5, 6.
ow_t_file_: 5, 6.
o2externs: 5.
parallel-parser-phrase: 14.
parser_: 5, 8, 12, 13, 16, 18, 19, 20, 23, 26, 29, 30, 31, 32, 33, 34.
pdftex: 2.
previous_state_: 8.
PRT_RULE_S_FIRST_SET: 7.

push_back: 5, 23.
p1_: 12, 13, 16, 18, 19, 20, 23, 26, 29, 30, 32, 33, 34.
p2_: 32, 33, 34.
p3_: 32.
r_def: 30.
R_rule: 6.
R_T: 6.
rc_phrase: 17.
rd: 8.
re_align_nos: 32, 33.
reduced_state_: 8.
refered_rule: 28.
refered_T: 28.
refered_rule: 6.
refered_T: 6.
Relement: 27.
Relement: 28, 29, 30, 31, 32, 33, 34.
Relements: 27.
Relements: 25, 27.
Rerr_phrase: 8.
Rerr_phrase: 16.
Rfsm_phrase: 8.
Rfsm_phrase: 13.
Rgrammar_phrase: 8.
Rgrammar_phrase: 9.
rhs_syms_str: 8.
Rk: 11.
Rk: 12.
Rks: 11.
Rks: 10, 11.
Rks_epi: 8, 21.
Rks_epi: 10.
Rlr1_k_phrase: 8.
Rlr1_k_phrase: 18.
Rmpost_output: 8.
rname: 23, 26.
Rparallel_phrase: 8.
Rparallel_phrase: 14.
Rrc_phrase: 8.
Rrc_phrase: 17.
Rrule: 22.
Rrule: 21.
Rrule_def: 22.
Rrule_phrase: 8.
Rrule_def: 23.
Rrule_phrase: 20.
Rrules: 8, 21.
Rrules: 21.
Rsubrule: 24.
Rsubrule: 25.
Rsubrule_def: 25.

Rsubrule_def: 26.
Rsubrules: 24.
Rsubrules: 22, 24.
RT_enum_phrase: 8.
RT_enum_phrase: 15.
Rterms_phrase: 8.
Rterms_phrase: 19.
rule-def: 23.
rule_cweb: 23.
rule_cweb_diagram: 23.
Rule_def: 6, 7.
rule_def: 6, 7, 8.
rule_def: 5, 6, 23, 26.
Rule_in_stbl: 30.
rule_info_: 8, 12, 13, 16, 18, 19, 20, 23, 26, 29,
 30, 31, 32, 33, 34.
rule_lhs: 26.
rule_name: 8, 23, 26, 30.
rule_name_: 6.
rule_no: 8.
rule_no_: 5, 6, 23, 26, 32, 33.
rule_s_subrule_no_elems: 32.
rules-phrase: 20.
rules_alphabet: 20.
rules_for_fs_prt: 6, 23.
S_VECTOR_ELEMS_ITER_type: 8.
S_VECTORS_ITER_type: 8.
se: 6, 8.
second: 8.
seli: 8.
selie: 8.
set_stop_parse: 5.
set_who_created: 5.
sf: 2, 12, 13, 16, 18, 19, 20, 23, 26, 29, 30,
 32, 33, 34.
si: 8.
sie: 8.
size: 20.
size_type: 29, 32, 33, 34.
sprintf: 8, 20, 23, 26, 32, 33.
sr_element_: 8.
sr_figure: 8.
sr_no: 8.
sr_ph: 23.
srd: 8, 23.
srules: 23.
state: 8.
state_element: 6, 8.
state_no_: 8.
state_s_vector_: 8.
state_type: 8.
state_type_: 8.
state_type_template: 8.
STATES_ITER_type: 8.
STATES_type: 7.
std: 6, 7, 23, 29, 30, 31, 32, 33, 34.
string: 5, 6, 8, 29, 30, 31, 32, 33, 34.
subrule: 32.
subrule-def: 26.
subrule_cweb: 26.
subrule_cweb_diagram: 26.
Subrule_def: 6.
subrule_def: 5, 6, 8, 26.
subrule_no_: 5, 6, 23, 26, 32, 33.
subrule_no_of_rule: 8.
subrule_s_tree: 23.
subrule_sym: 8.
Subrule_tree: 6.
subrules: 23.
svi: 8.
svie: 8.
Sym: 7.
sym: 5.
Sym_to_xlate: 7.
T-enum-phrase: 15.
t_def: 29, 32, 33, 34.
T_Enum: 32, 33.
T_error_symbols_phrase: 6.
t_file: 5.
T_fsm_phrase: 6.
t_in_stbl: 29, 32, 33, 34.
T_lr1_k_phrase: 6.
T_LR1_parallel_operator_: 32, 33.
t_name: 29, 32, 33, 34.
T_phrase: 6.
T_subrule_def: 6, 8, 23.
T_subrules_phrase: 23.
T_terminals_phrase: 6.
terminals-phrase: 19.
theTime: 5.
three_symbols_string_template: 8.
time: 5.
true: 5.
trunc: 5.
vector: 6, 23.
vectored_into: 8.
vectored_into_by_elem_: 8.
w_end: 8.
w_fig_no_: 5, 6, 23, 26.
w_filename_: 5, 6.
w_fsc_file: 8.
w_index_filename_: 5, 6.
w_lr1_states: 8.
w_possible_ar_code: 8.

w_stateno: 8.
w_stateno_subrule: 8.
WRT_CWEB_MARKER: 12, 23, 26.
XLATE_SYMBOLS_FOR_cweave: 7, 8.
xlated_names_: 6.
Xlated_str: 6.
Xlated_sym: 7.

```
< Cmpost_output constructor directive 4 >
< Cmpost_output op directive 5 >
< Cmpost_output user-declaration directive 6 >
< Cmpost_output user-prefix-declaration directive 7 >
< Relement subrule 1 op directive 29 >
< Relement subrule 2 op directive 30 >
< Relement subrule 3 op directive 31 >
< Relement subrule 4 op directive 32 >
< Relement subrule 5 op directive 33 >
< Relement subrule 6 op directive 34 >
< Rerr_phrase subrule 1 op directive 16 >
< Rfsm_phrase subrule 1 op directive 13 >
< Rk subrule 1 op directive 12 >
< Rlr1_k_phrase subrule 1 op directive 18 >
< Rmpost_output subrule 1 op directive 8 >
< Rrule_def subrule 1 op directive 23 >
< Rrule_phrase subrule 1 op directive 20 >
< Rsubrule_def subrule 1 op directive 26 >
< Rterms_phrase subrule 1 op directive 19 >
```

m_{post}_output Grammar

Date: January 2, 2015 at 15:37

File: m_{post}_output.lex

Ns: NS_m_{post}_output

Version: 1.0

Debug: false

Grammar Comments:

Type: Monolithic

Output grammar rules railroad diagrams for m_{post} that cweb program uses.

	Section	Page
Copyright	1	1
m_{post}_output Grammar	2	2
Fsm Cm _{post} _output class	3	2
Cm _{post} _output constructor directive	4	2
Cm _{post} _output op directive	5	3
Cm _{post} _output user-declaration directive	6	5
Cm _{post} _output user-prefix-declaration directive	7	6
Rm _{post} _output rule	8	7
Rgrammar_phrase rule	9	10
Rks_ksi rule	10	10
Rks rule	11	10
Rk rule	12	10
Rfsm_phrase rule	13	11
Rparallel_phrase rule	14	11
RT_enum_phrase rule	15	11
Rerr_phrase rule	16	11
Rrc_phrase rule	17	11
Rlr1_k_phrase rule	18	11
Rterms_phrase rule	19	11
Rrule_phrase rule	20	12
Rrules rule	21	12
Rrule rule	22	12
Rrule_def rule	23	13
Rsubrules rule	24	14
Rsubrule rule	25	14
Rsubrule_def rule	26	15
Relements rule	27	15
Relement rule	28	16
Relement's subrule 1	29	16
Relement's subrule 2	30	16
Relement's subrule 3	31	17
Relement's subrule 4	32	18
Relement's subrule 5	33	19
Relement's subrule 6	34	20
First Set Language for O ₂ ^{linker}	35	21

mpost_output Grammar	TABLE OF CONTENTS	1
Lr1 State Network	36	22
Index	37	29