1. Copyright.

2. *unq_str* **Thread.**
This recognizer could be used in a "file name" option of a command line sequence: a quoted filename is used if it contains spaces or other abnormal characters to be included as part of it. The expression recognized is crude: just a wild character loop. Once parsed, it does no checking. It is left for the calling context to do the validity check. Regarding file processing, one must still go to the file system for the existance check.
Note how the lookahead (LA) character (current token) is checked using fsm's *la_bnds_chk* function to turn off the |+| consumption.

Why is there no escape sequence like UNIX's scripting: the blackslash character-to-literal-accept? Well i'm not fighting different contexts and how many variants do u need? Hear the stomping of my foot? Enough ... just quote it!

3. Fsm Cunq_str class.

4. Cunq_str op directive.

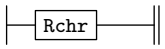
```
⟨ Cunq_str op directive 4 ⟩ ≡
  parser__→set_use_all_shift_on( );
  c_.erase( );
  if (la_bnds_fnd(parser__→start_token__) ≡ true) parser__→set_use_all_shift_off( );      /* stop the parse */
```

5. Cunq_str user-declaration directive.

```
⟨ Cunq_str user-declaration directive 5 ⟩ ≡
public: bool la_bnds_fnd(CAbs_lr1_sym * Sym)
  {
    using namespace NS_yacco2_terminals;
    int id = Sym→enumerated_id__;
    switch (id) {
    case T_Enum :: T_raw_open_brace_: break;
    case T_Enum :: T_raw_close_brace_: break;
    case T_Enum :: T_raw_lf_: break;
    case T_Enum :: T_raw_cr_: break;
    case T_Enum :: T_LR1_eog_: break;
    case T_Enum :: T_T_eol_: break;
    case T_Enum :: T_raw_ht_: break;
    case T_Enum :: T_raw_sp_: break;
    case T_Enum :: T_raw_vt_: break;
    case T_Enum :: T_raw_ff_: break;
    case T_Enum :: T_raw_dbl_quote_: break;
    default:
      {
        return false;
      }
    }
    return true;
  }
  ;
  std :: string c_;
```

6. *Runq_str* **rule.**

`Runq_str`

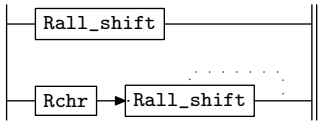


⟨ Runq_str subrule 1 op directive 6 ⟩ ≡

```
Cunq_str * fsm = ( Cunq_str * ) rule_info__.parser__→fsm_tbl__;
CAbs_lr1_sym * sym = new T_unquoted_string(fsm→c_);
sym→set_rc(*rule_info__.parser__→start_token__, __FILE__, __LINE__);
RSVP(sym);
```

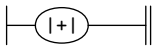
7. *Rchr* **rule.**

`Rchr`



8. *Rall_shift* **rule.**

`Rall_shift`



⟨ Rall_shift subrule 1 op directive 8 ⟩ ≡

```
Cunq_str * fsm = ( Cunq_str * ) rule_info__.parser__→fsm_tbl__;
CAbs_lr1_sym * sym = sf→p1__;
fsm→c_ += sym→id__;
if (fsm→la_bnds_fnd(rule_info__.parser__→current_token( )) ≡ true) {      /* chk LA */
  rule_info__.parser__→set_use_all_shift_off( );
}
```

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

```
/*
 File: unq_str.fsc
 Date and Time: Fri Jan  2 15:34:00 2015
*/
transitive   n
grammar-name "unq_str"
name-space   "NS_unq_str"
thread-name  "TH_unq_str"
monolithic   n
file-name    "unq_str.fsc"
no-of-T      569
list-of-native-first-set-terminals 1
   LR1_all_shift_operator
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
"Unquoted string of characters: raw and basic."
```

10. Lr1 State Network.

⇒ State: 1 state type: ^{s}

	←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rall_shift			3	1	1	`|+|`			1	2	2	
c	Runq_str			1	1	1	Rchr			1	3	3	
c	Rchr			2	2	1	Rchr *Rall_shift*			1	3	4	
c	Rchr			2	1	1	Rall_shift			1	5	5	

⇒`|+|` State: 2 state type: ^{r}

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

⇒^{Rchr} State: 3 state type: $^{s/r}$

	←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Runq_str			1	1	2				1	0	3	1
c	Rall_shift			3	1	1	`|+|`			3	2	2	
t	Rchr			2	2	2	Rall_shift			1	4	4	

⇒$^{Rall\_shift}$ State: 4 state type: ^{r}

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

⇒$^{Rall\_shift}$ State: 5 state type: ^{r}

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

11. Index.

⟨ Cunq_str op directive 4 ⟩
⟨ Cunq_str user-declaration directive 5 ⟩
⟨ Rall_shift subrule 1 op directive 8 ⟩
⟨ Runq_str subrule 1 op directive 6 ⟩

unq_str Grammar

Date: January 2, 2015 at 15:40

File: unq_str.lex Ns: NS_unq_str

Version: 1.0 Debug: false

Grammar Comments: Type: Thread

Unquoted string of characters: raw and basic.

1 element(s) in Lookahead Expression below

eolr