1. Copyright.

Copyright © Dave Bone 1998 - 2015

2. *first_set_rules* **grammar.**
Create a rule's first set by building a closure-only state. The terminals within this state are its first set. Each rule's first set within the grammar is built this way including the "start rule" of the grammar and possibly rules used only in a "parallel-la-boundary" expression.

The difference between this first set calculation and the one described in "The Theory of Parsing, Translation, and Compiling Volume 1: Parsing" by Aho and Ullman on Page 300 is mine only evaluates the terminal(s) within the state while theirs "derives" the terminal string with all their substitutions to arrive at a set of terminals. **Theirs is correct** as i did not completely walk the rule's subsubrules when evaluating a partially epsiloned string at follow set calculation — it assumed that the rule's first set was already calculated: α string is made up of an epsiloned rule Ra_ϵ followed by β string of rules or terminals. The first set was not filled in properly composed of FS(α) and FS(β) but only the first string FS(α). When the follow set was calculated, it did not explode the follow set string into its symbol composites. Boy i'm dumb. Lets look at the short of it: the original first set calculation was for efficiency at the expense (when viewed now) to explode per follow string its composite first sets. What a dumb idea — as all mistakes are! So now its a closure only state with interior symbols added due to epsilon rules. The first set view is all the terminals brought in from the rule's subrules. As other rules can be brought in due to closure, their subrules are added to the state. Now epsilon rules that start the subrule's symbol string are the camillions. They add their own teminals to the first set but also disappear and allow their next right symbol to be included in the state. This is recursive as these partially consumed subrules are then evaluated as above.

Below is my then thought process and its assumptions:
So what's the difference on k = 1 symbol lookahead? Epsilon rules. I do not pursue the lookahead terminal string that would be derived. For the record, this gets done when the **follow set** of a symbol string is calculated. The First set of a rule is used with an epsilon check to determine whether the next symbol in the string should be followed. Why advance if it is an epsilon rule? Epsilon is like a window where u see past the rule into its neighbour's setting; one can view it another way, an epsilon rule plays 2 parts: it provides its first set and provides its to-its-right string. When the follow set calculation hits the end-of-the-string — i call this right-bounded condition, the remaining follow set is found from the Follow set of its spawning rule(s). This is transitive as the spawning rule's follow set calculation could also hit the end-of-string condition. Now where are these follow set strings found? — in the state that spawned them. Each state contains the rule's follow set graph.

Definition of First set:
Terminals that start all substrings generated by the rule's productions. The grammar tree is walked in prefix formation accepting only "rule-def" followed by its "subrule-def" terminals. Each rule within the grammar follows this pattern: ie, the start-rule is the first to be evaluated. Though it is never referenced in a subrule i still create its first set.

The Algorithm.
The grammar reads each individual rule-def and all its subrule-def(s). Using its bottom-up recognition, *Rsubrule_def* adds the 1st element of the subrule into the *fs_list_*. *Rrule* processes the *fs_list_* as a closure-only state generating the rule's first set. In generating the first set, the elements in *fs_list_* are consumed as they are evaluated by removal from the list. Referenced terminals are added to the rule's first set. For 1st time referenced rules, their subrules are added at the end of *fs_list_* for eventual consumption. The neat thing about this algorithm is the 1st element in the *fs_list_* is only visited! It's a singular point of evaluation that is thrown out to be replaced by its next in line element: ahh the bank queue and the teller.

Due to *cweave* irregularities in formatting C++ code of this grammar, please see *o2externs* documentation where the routines `GEN_FS_OF_RULE` is coded an external to overcome this deficiency.

3. Fsm Cfirst_set_rules class.

4. Cfirst_set_rules op directive.

⟨ Cfirst_set_rules op directive 4 ⟩ ≡
*rule_def_* = 0;

5. Cfirst_set_rules user-declaration directive.

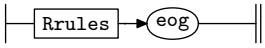
⟨ Cfirst_set_rules user-declaration directive 5 ⟩ ≡
public: *rule_def* ∗ *rule_def_*;

6. Cfirst_set_rules user-prefix-declaration directive.

⟨ Cfirst_set_rules user-prefix-declaration directive 6 ⟩ ≡
#**include** "o2_externs.h"

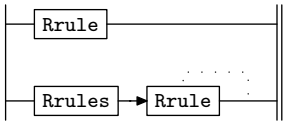
7. *Rfirst_set_rules* **rule.**

Rfirst_set_rules



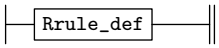
8. *Rrules* **rule.**

Rrules



9. *Rrule* **rule.**

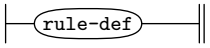
Rrule



⟨ Rrule subrule 1 op directive 9 ⟩ ≡
Cfirst_set_rules ∗ *fsm* = (*Cfirst_set_rules* ∗) *rule_info__.parser__→fsm_tbl__*;
GEN_FS_OF_RULE(*fsm→rule_def_*);

10. *Rrule_def* **rule.**

Rrule_def



Initialize for its subrule findings.

⟨ Rrule_def subrule 1 op directive 10 ⟩ ≡
Cfirst_set_rules ∗ *fsm* = (*Cfirst_set_rules* ∗) *rule_info__.parser__→fsm_tbl__*;
*fsm→rule_def_* = *sf→p1__*;

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

```
/*
 File: first_set_rules.fsc
 Date and Time: Fri Jan  2 15:33:36 2015
*/
transitive   n
grammar-name "first_set_rules"
name-space   "NS_first_set_rules"
thread-name  "Cfirst_set_rules"
monolithic   y
file-name    "first_set_rules.fsc"
no-of-T      569
list-of-native-first-set-terminals 1
   rule_def
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
"Determine first set per rule."
```

12. Lr1 State Network.

⇒ State: 1 state type: ^{s}

	← rule	→	R#	sr#	Po	← subrule element	→	Brn	Gto	Red	LA
c	Rrule_def		4	1	1	rule-def		1	2	2	
c	Rfirst_set_rules		1	1	1	Rrules $\overline{eog}$		1	3	4	
c	Rrules		2	2	1	Rrules $\overline{Rrule}$		1	3	5	
c	Rrules		2	1	1	Rrule		1	7	7	
c	Rrule		3	1	1	Rrule_def		1	6	6	

$\Rightarrow^{rule-def}$ State: 2 state type: ^{r}

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

$\Rightarrow^{Rrules}$ State: 3 state type: ^{s}

	← rule	→	R#	sr#	Po	← subrule element	→	Brn	Gto	Red	LA
t	Rfirst_set_rules		1	1	2	eog		1	4	4	
c	Rrule_def		4	1	1	rule-def		3	2	2	
t	Rrules		2	2	2	Rrule		1	5	5	
c	Rrule		3	1	1	Rrule_def		3	6	6	

$\Rightarrow^{eog}$ State: 4 state type: ^{r}

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

$\Rightarrow^{Rrule}$ State: 5 state type: ^{r}

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

$\Rightarrow^{Rrule\_def}$ State: 6 state type: ^{r}

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

$\Rightarrow^{Rrule}$ State: 7 state type: ^{r}

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

13. Index.

⟨ Cfirst_set_rules op directive 4 ⟩
⟨ Cfirst_set_rules user-declaration directive 5 ⟩
⟨ Cfirst_set_rules user-prefix-declaration directive 6 ⟩
⟨ Rrule subrule 1 op directive 9 ⟩
⟨ Rrule_def subrule 1 op directive 10 ⟩

first_set_rules Grammar

Date: January 2, 2015 at 15:35

File: first_set_rules.lex

Ns: NS_first_set_rules

Version: 1.0 Debug: false

Grammar Comments: Type: Monolithic

Determine first set per rule.