1. *thread_fs_calc* **grammar.**
Create a rule's first set of called threads by building a closure-only state. The terminals within this state are the "called threads" used for the "list-of-transitive-threads" construct of the generated "fsc" emitted file for O_2^{linker}. 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 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_CALLED_THREADS_FS_OF_RULE` is coded an external to overcome this deficiency.

2. Fsm Cthread_fs_calc class.

3. Cthread_fs_calc op directive.

⟨ Cthread_fs_calc op directive 3 ⟩ ≡
```
rule_def_ = 0;
subrule_def_ = 0;
elem_t_ = 0; ip_can_ = ( tok_can < AST *> * ) parser__→token_supplier__;
```

4. Cthread_fs_calc user-declaration directive.

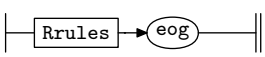
⟨ Cthread_fs_calc user-declaration directive 4 ⟩ ≡
```
public: FS_ELEM_LIST_type fs_list_;
RULES_IN_FS_LIST_type rules_in_fs_list_;
rule_def * rule_def_;
T_subrule_def * subrule_def_;
AST * elem_t_;
tok_can < AST *> *ip_can_;
```

5. Cthread_fs_calc user-prefix-declaration directive.

⟨ Cthread_fs_calc user-prefix-declaration directive 5 ⟩ ≡
```
#include "o2_externs.h"
```

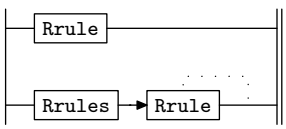
6. *Rthread_fs_calc* **rule.**

```
Rthread_fs_calc
```



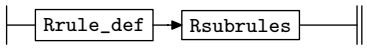
7. *Rrules* **rule.**

```
Rrules
```



8. *Rrule* **rule.**

`Rrule`

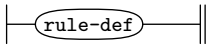


⟨ Rrule subrule 1 op directive 8 ⟩ ≡

```
Cthread_fs_calc * fsm = ( Cthread_fs_calc * ) rule_info__.parser__->fsm_tbl__;
GEN_CALLED_THREADS_FS_OF_RULE(fsm->fs_list_, fsm->rules_in_fs_list_, fsm->rule_def_);
```

9. *Rrule_def* **rule.**

`Rrule_def`



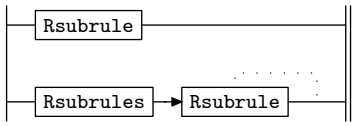
Initialize for its subrule findings.

⟨ Rrule_def subrule 1 op directive 9 ⟩ ≡

```
Cthread_fs_calc * fsm = ( Cthread_fs_calc * ) rule_info__.parser__->fsm_tbl__;
fsm->rule_def_ = sf->p1__;
fsm->rules_in_fs_list_.clear( );
fsm->fs_list_.clear( );
```

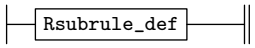
10. *Rsubrules* **rule.**

`Rsubrules`



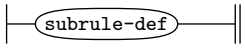
11. *Rsubrule* **rule.**

`Rsubrule`



12. *Rsubrule_def* **rule.**

`Rsubrule_def`



Create the entry within the *fs_list_*. Only the 1st element of eac subrule is evaluated.

⟨ Rsubrule_def subrule 1 op directive 12 ⟩ ≡

```
Cthread_fs_calc * fsm = ( Cthread_fs_calc * ) rule_info__.parser__->fsm_tbl__;
fsm->subrule_def_ = sf->p1__;
AST * sr_t = fsm->subrule_def_->subrule_s_tree( );
AST * et = AST::get_spec_child(*sr_t, 1);
fsm->fs_list_.push_back(FS_ELEM_type(fsm->rule_def_, fsm->subrule_def_, et));
```

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

```
/*
 File: thread_fs_calc.fsc
 Date and Time: Sun Oct 30 13:39:24 2011
*/
transitive   n
grammar-name "thread_fs_calc"
name-space   "NS_thread_fs_calc"
thread-name  "Cthread_fs_calc"
monolithic   y
file-name    "thread_fs_calc.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 of thread calls per rule."
```

14. 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	Rrules		2	2	1	Rrules *Rrule*			1	3	5	
c	Rthread_fs_calc		1	1	1	Rrules *eog*			1	3	4	
c	Rrules		2	1	1	Rrule			1	12	12	
c	Rrule		3	1	1	Rrule_def *Rsubrules*			1	6	8	

⇒$^{rule-def}$ 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

⇒^{Rrules} State: 3 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rthread_fs_calc		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 *Rsubrules*			3	6	8	

⇒^{eog} State: 4 state type: ^{r}

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

⇒^{Rrule} State: 5 state type: ^{r}

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

⇒$^{Rrule\_def}$ State: 6 state type: ^{s}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
c	Rsubrule_def		7	1	1	subrule-def			6	7	7	
t	Rrule		3	1	2	Rsubrules			3	8	8	
c	Rsubrules		5	2	1	Rsubrules *Rsubrule*			6	8	9	
c	Rsubrules		5	1	1	Rsubrule			6	11	11	
c	Rsubrule		6	1	1	Rsubrule_def			6	10	10	

⇒$^{subrule-def}$ State: 7 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rsubrule_def		7	1	2				6	0	7	4

⇒^{Rsubrules} State: 8 state type: $^{s/r}$

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rrule		3	1	3				3	0	8	3
c	Rsubrule_def		7	1	1	subrule-def			8	7	7	
t	Rsubrules		5	2	2	Rsubrule			6	9	9	
c	Rsubrule		6	1	1	Rsubrule_def			8	10	10	

⇒^{Rsubrule} State: 9 state type: ^{r}

←	rule	→	R#	sr#	Po	←	subrule element	→	Brn	Gto	Red	LA
t	Rsubrules		5	2	3				6	0	9	4

$\Rightarrow^{Rsubrule\_def}$

	←	rule	→	R#	sr#	Po	←
t	Rsubrule			6	1	2	

State: 10 state type: ^{r}

	subrule element	→	Brn	Gto	Red	LA
			8	0	10	4

$\Rightarrow^{Rsubrule}$

	←	rule	→	R#	sr#	Po	←
t	Rsubrules			5	1	2	

State: 11 state type: ^{r}

	subrule element	→	Brn	Gto	Red	LA
			6	0	11	4

$\Rightarrow^{Rrule}$

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

State: 12 state type: ^{r}

	subrule element	→	Brn	Gto	Red	LA
			1	0	12	3

15. Index.

⟨ Cthread_fs_calc op directive 3 ⟩
⟨ Cthread_fs_calc user-declaration directive 4 ⟩
⟨ Cthread_fs_calc user-prefix-declaration directive 5 ⟩
⟨ Rrule subrule 1 op directive 8 ⟩
⟨ Rrule_def subrule 1 op directive 9 ⟩
⟨ Rsubrule_def subrule 1 op directive 12 ⟩

thread_fs_calc Grammar

Date: October 30, 2011 at 13:50

File: thread_fs_calc.lex

Ns: NS_thread_fs_calc

Version: 1.0 Debug: false

Grammar Comments: Type: Monolithic

Determine first set of thread calls per rule.