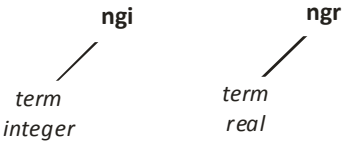
48	*simple_expression*	→	*term*
49	*simple_expression*	→	*sign term*
50	*simple_expression*	→	*simple_expression addop term*
51	*sign*	→	**+**
52	*sign*	→	**-**
53	*addop*	→	**+**
54	*addop*	→	**-**
55	*addop*	→	**or**

48	*simple_expression*	→	*term* $$=$1

49 *simple_expression* → *sign term*

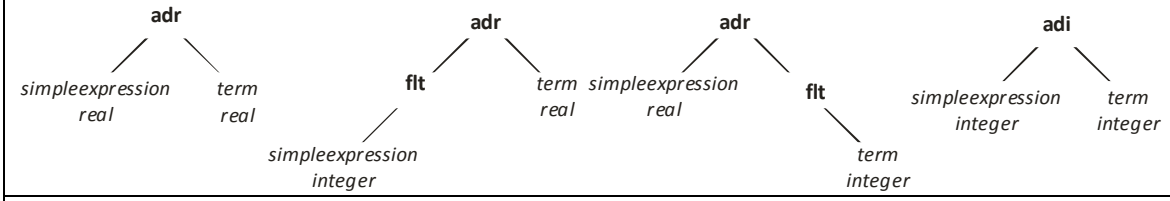


Exp simple_expression*(*string* sign*,*Exp* term*)

1. **if (***sign***==”+”) return** *term*
2. *term* must have either type integer or type real. No other type is permitted.
3. If term has type integer then
   - 3.1. Create PCode
     - 3.1.1.**ngi**
   - 3.2. Return expression e having
     - 3.2.1.PCode P created in 3.1
     - 3.2.2.Type integer
     - 3.2.3.Left sub expression: *term*
     - 3.2.4.Right sub expression: null
4. If term has type real then
   - 4.1. Create PCode
     - 4.1.1.**ngr**
   - 4.2. Return expression e having
     - 4.2.1.PCode P created in 4.1
     - 4.2.2.Type real
     - 4.2.3.Left sub expression: *term*
     - 4.2.4.Right sub expression: null

50 *simple_expression* → *simple_expression addop term*
Exp simple_expression*(*Exp* simpleexpression*,*string* addop*,*Exp* term*)
if **addop*=**"+" return** *add*(*simpleexpression*,*term*)
if **addop*=**"-" return** *subtract*(*simpleexpression*,*term*)
if **addop*=**"or" return** *disjunction*(*simpleexpression*,*term*)
else *error*(**"invalid addop"**)

50 *simple_expression* → *simple_expression addop term*
53 *addop* → **+**



Exp add*(*Exp* l*,*Exp* r*)

1. Semantics for term are dependent on *addop*. The information given here is for rule 53 *addop* → **+**.
2. Four expressions are returned depending on the types of input parameters *l* and *r*.

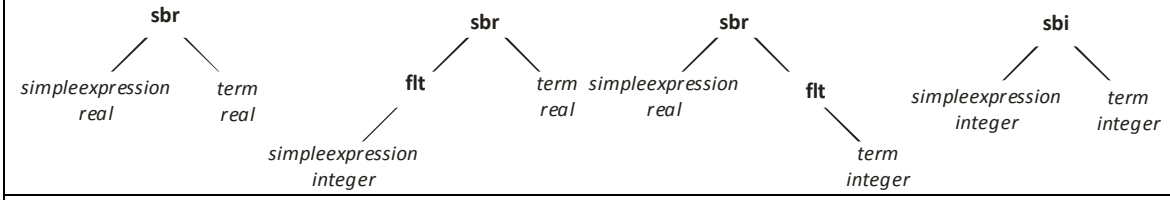
2.1. Parameter l has type real. Parameter r has type real.
2.1.1. Create PCode P
adr
2.1.2. Return Expression E
PCode P created in 2.1.1.
Type real
Left sub expression: parameter l
Right sub expression : parameter r

2.2. Parameter l has type integer. Parameter r has type real.
2.2.1. Create PCode P
flt
2.2.2. Create Expression L
PCode P created in 2.2.1.
Type real
Left subexpression: l
2.2.3. Create PCode P
adr
2.2.4. Return Expression E
PCode P created in 2.2.3.
Type real
Left sub expression: expression L created in 2.2.2.
Right sub expression : parameter r

2.3. Parameter l has type real. Parameter rhas type integer.
- 2.3.1. Create PCode P
  - **flt**
- 2.3.2. Create Expression R
  - PCode P created in 2.3.1.
  - Type real
  - Right subexpression: r
- 2.3.3. Create PCode P
  - **adr**
- 2.3.4. Return Expression E
  - PCode P created in 2.2.3.
  - Type real
  - Left sub expression: parameter r
  - Right sub expression : expression R created in 2.3.3.

2.4. Parameter l has type integer. Parameter r has type integer.
- 2.4.1. Create PCode P
  - **adi**
- 2.4.2. Return Expression E
  - PCode P created in 2.4.1.
  - Type integer
  - Left sub expression: parameter l
  - Right sub expression : parameter r

50 *simple_expression* → *simple_expression addop term*
54 *addop* → **-**



***Exp* subtract*(*Exp* l*,*Exp* r*)**
1. Semantics for term are dependent on *addop*. The information given here is for rule 54 *addop* → **-**.
2. Four expressions are returned depending on the types of input parameters *l* and *r*.

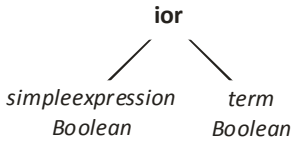
2.1. Parameter l has type real. Parameter r has type real.
- 2.1.1. Create PCode P
  - **sbr**
- 2.1.2. Return Expression E
  - PCode P created in 2.1.1.
  - Type real
  - Left sub expression: parameter l
  - Right sub expression : parameter r

2.2. Parameter l has type integer. Parameter r has type real.
- 2.2.1. Create PCode P
  - **flt**
- 2.2.2. Create Expression L
  - PCode P created in 2.2.1.
  - Type real
  - Left subexpression: l
- 2.2.3. Create PCode P
  - **sbr**
- 2.2.4. Return Expression E
  - PCode P created in 2.2.3.
  - Type real
  - Left sub expression: expression L created in 2.2.2.
  - Right sub expression : parameter r

2.3. Parameter l has type real. Parameter rhas type integer.
- 2.3.1. Create PCode P
  - **flt**
- 2.3.2. Create Expression R
  - PCode P created in 2.3.1.
  - Type real
  - Right subexpression: r
- 2.3.3. Create PCode P
  - **sbr**
- 2.3.4. Return Expression E
  - PCode P created in 2.2.3.
  - Type real
  - Left sub expression: parameter r
  - Right sub expression : expression R created in 2.3.3.

2.4. Parameter l has type integer. Parameter r has type integer.
- 2.4.1. Create PCode P
  - **sbi**
- 2.4.2. Return Expression E
  - PCode P created in 2.4.1.
  - Type integer
  - Left sub expression: parameter l
  - Right sub expression : parameter r

50 *simple_expression* → *simple_expression addop term*
55 *addop* → **or**



Exp disjunction*(*Exp* l*,*Exp* r*)

1. Semantics for term are dependent on *addop*. The information given here is for rule 55 addop → **or**.
2. Both input expressions, l and r, must have type Boolean.
3. Create PCode P
   3.1. **ior**
4. Create expression E having
   4.1. PCode P created in step 3
   4.2. Type: Boolean
   4.3. Left sub expression: l.
   4.4. Right sub expression: r.