Улучшение результатов суперкомпиляции при помощи препроцессирования исходной программы

Т.В. Рудикова

МГТУ имени Н.Э Баумана

Второе совместное рабочее совещание
ИПС имени А.К. Айламазяна РАН и МГТУ имени Н.Э Баумана
по функцианальному языку программирования Рефал

11 июня 2019 года

Цель работы

- Написать препроцессор позволяющий обогатить конфигурации, без изменения при этом самого суперкомпилятора
- Рассмотреть примеры влияние препроцессирования на процесс суперкомпиляции

Описание работы первого препроцессора

Первый препроцессор предоставляет следующие способы преобразования результатных выражений исходных программ:

- замена константных символов на вызовы функций
- замена круглых скобок на вызовы функций
- добавление вызовов пустых функций

Описание работы второго препроцессора

Второй препроцессор предоставляет следующие способы преобразования исходных программ:

- преобразование символов в пары «символ-координата», в которых координата является уникальной для каждого символа
- преобразование скобочных термов следующим образом: (значение) → ((значение) координата), где координата является уникальной для каждого скобочного терма

Программа замены литеры ‘A’ на литеру ‘B’

```
$ENTRY Fab { e.X
       = <DoFab e.X ('B')> }

DoFab {
  e.X 'A' (e.Res)
     = <DoFab e.X ('B' e.Res)>;
  e.X s.1 (e.Res)
     = <DoFab e.X (s.1 e.Res)>;
  /* empty */ (e.Res) = e.Res;
}
```

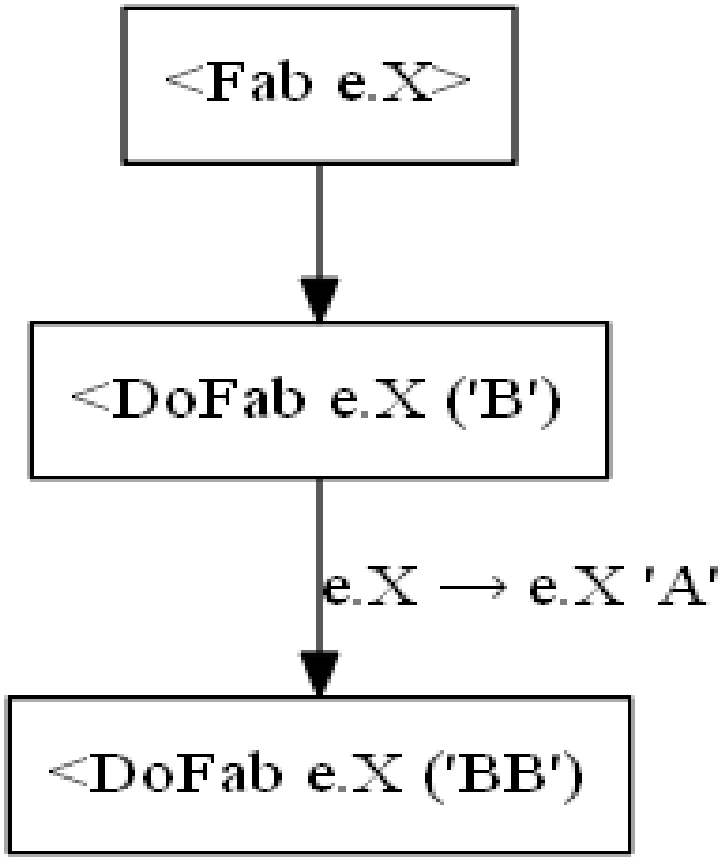
```
$ENTRY Fab { e.X
    = <DoFab e.X <b1 <c1>>> }

DoFab {
  e.X 'A' (e.Res)
    = <DoFab e.X <b2 <c2> e.Res>>;
  e.X s.1 (e.Res)
    = <DoFab e.X <b3 s.1 e.Res>>;
  /* empty */ (e.Res) = e.Res;
}

b1 { e.X = (e.X);}
c1 { = 'B';}
b2 { e.X = (e.X);}
c2 { = 'B';}
b3 { e.X = (e.X);}
```

Программа замены литеры ‘A’ на литеру ‘B’

```
$ENTRY Fab { e.X
         = <DoFab e.X ('B')> }

DoFab {
  e.X 'A' (e.Res)
     = <DoFab e.X ('B' e.Res)>;
  e.X s.1 (e.Res)
     = <DoFab e.X (s.1 e.Res)>;
  /* empty */ (e.Res) = e.Res;
}
```

```
$ENTRY Fab { e.vX
     = <DoFab e.vX (((‘B' K-1)) K-2)> }

DoFab {
  e.vX ('A' s.K-3) ((e.vRes) s.K-4)
  = <DoFab e.vX ((('B' K-5) e.vRes) K-6)>;
  e.vX (s.v1 s.K-7) ((e.vRes) s.K-8)
  = <DoFab e.vX (((s.v1 s.K-7) e.vRes) K-9)>;
  /* empty */ ((e.vRes) s.K-10) = e.vRes;
}
```

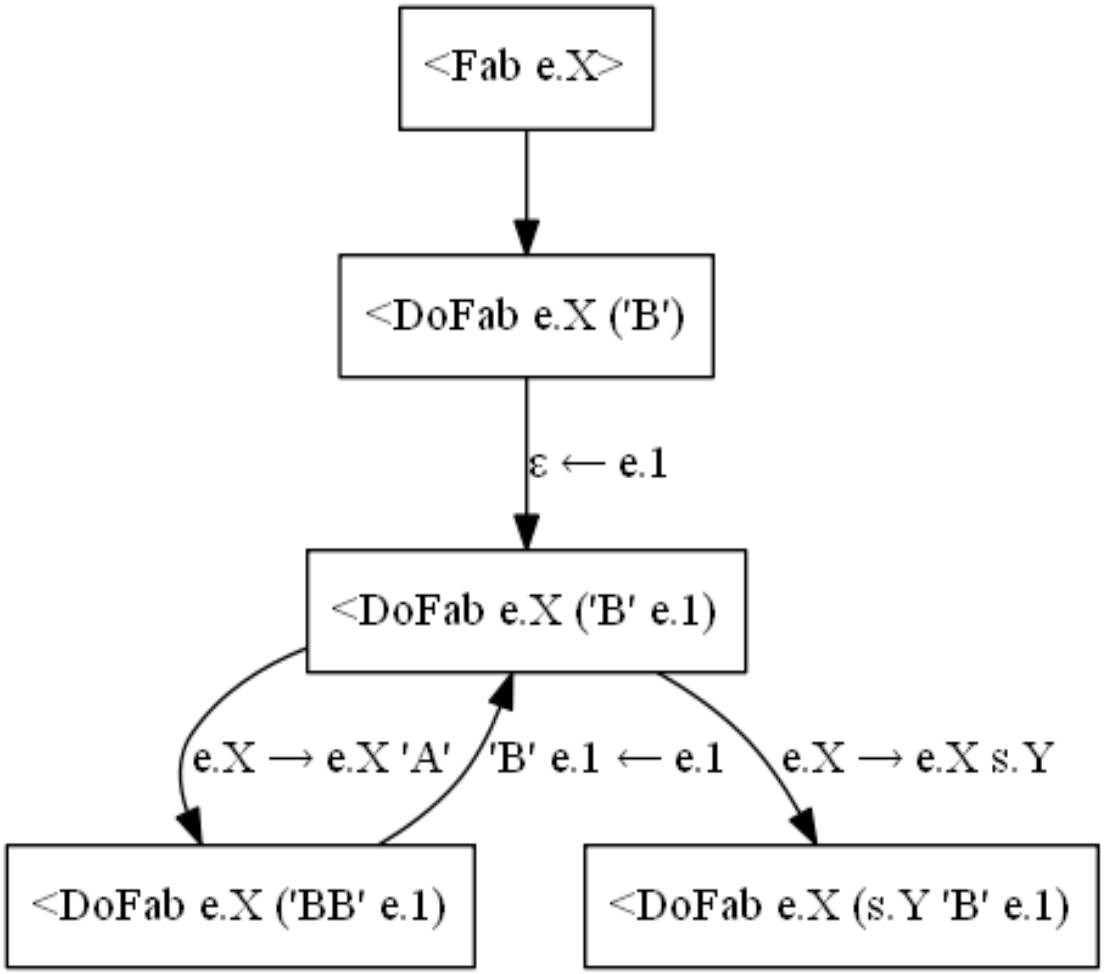
Суперкомпиляция исходной программы



Суперкомпиляция исходной программы

- *<DoFab e.X ('B')>* и *<DoFab e.X ('BB')>* похожи по отношению Хигмана-Крускала
- Возможно два обобщения: *<DoFab e.0 ('B' e.1)>* и *<DoFab e.0 (e.1 'B')>*

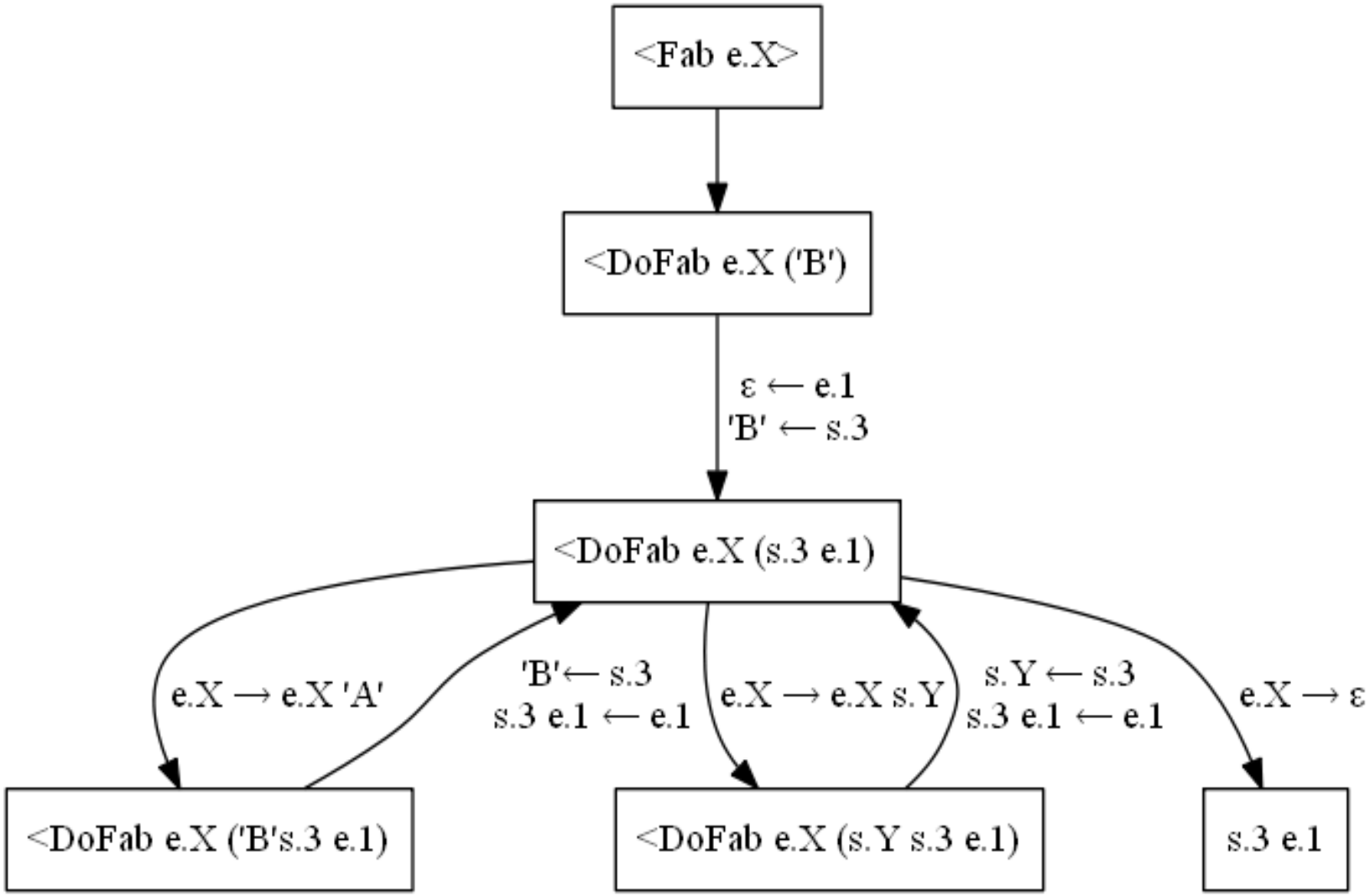
Суперкомпиляция исходной программы



Суперкомпиляция исходной программы

- конфигурации *<DoFab e.X ('B' e.1)>* и *<DoFab e.X (s.Y 'B' e.1)>* похожи
- Их обобщение: *<DoFab e.X (s.3 e.1)>.*

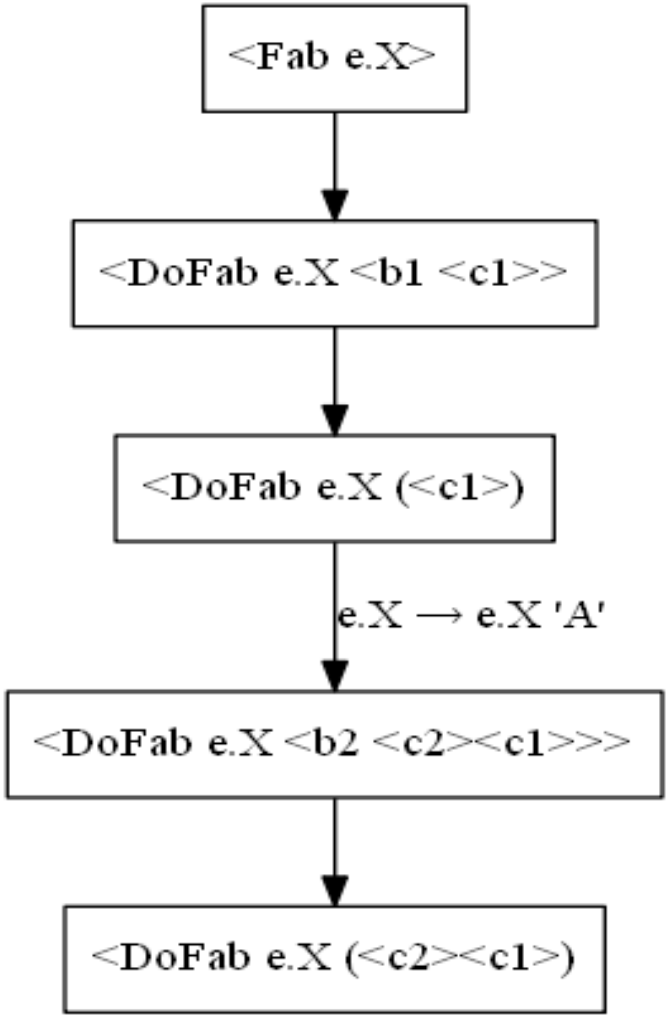
Суперкомпиляция исходной программы



Остаточная программа

```
$ENTRY Fab {
 e.41 'A', <F50 (e.41) 'BB'> : s.140 s.141 (e.142)
        =  s.140 s.141 e.142;
 e.41 s.101, <F50 (e.41) s.101 'B'> : s.127 s.128 (e.129)
        =  s.127 s.128 e.129;
        =  'B';
}

/*
*  InputFormat: <F50 (e.118 ) s.119 s.120 e.121 >
*  OutputFormat: ==> s.140 s.141 (e.142 )
*/
F50 {
 (e.118 'A') s.119 s.120 e.121,
        <F50 (e.118) 'B' s.119 s.120 e.121> : s.151 s.152 (e.153)
                                            =  s.151 s.152 (e.153);
 (e.118 s.122) s.119 s.120 e.121,
        <F50 (e.118) s.122 s.119 s.120 e.121> : s.154 s.155 (e.156)
                                            =  s.154 s.155 (e.156);
 () s.119 s.120 e.121 =  s.119 s.120 (e.121);
}
```

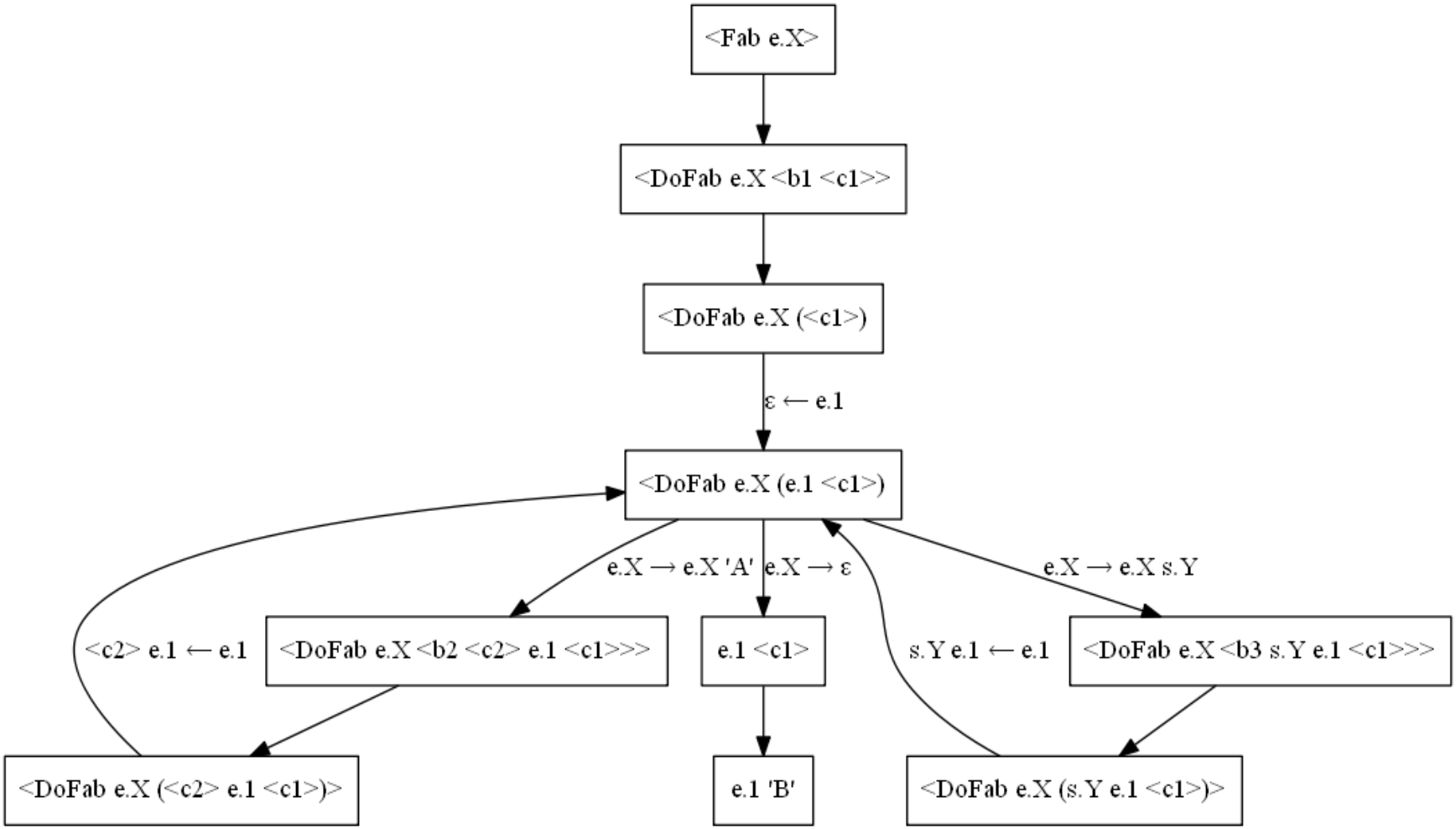
Суперкомпиляция после обработки первым препроцессором



Суперкомпиляция после обработки первым препроцессором

- Конфигурации *<DoFab e.X (<c1>)>* и *<DoFab e.X (<c2><c1>)>* похожи
- В данном случае обобщение единственно: *<DoFab e.X (e.1 <c1>)>*

Суперкомпиляция после обработки первым препроцессором



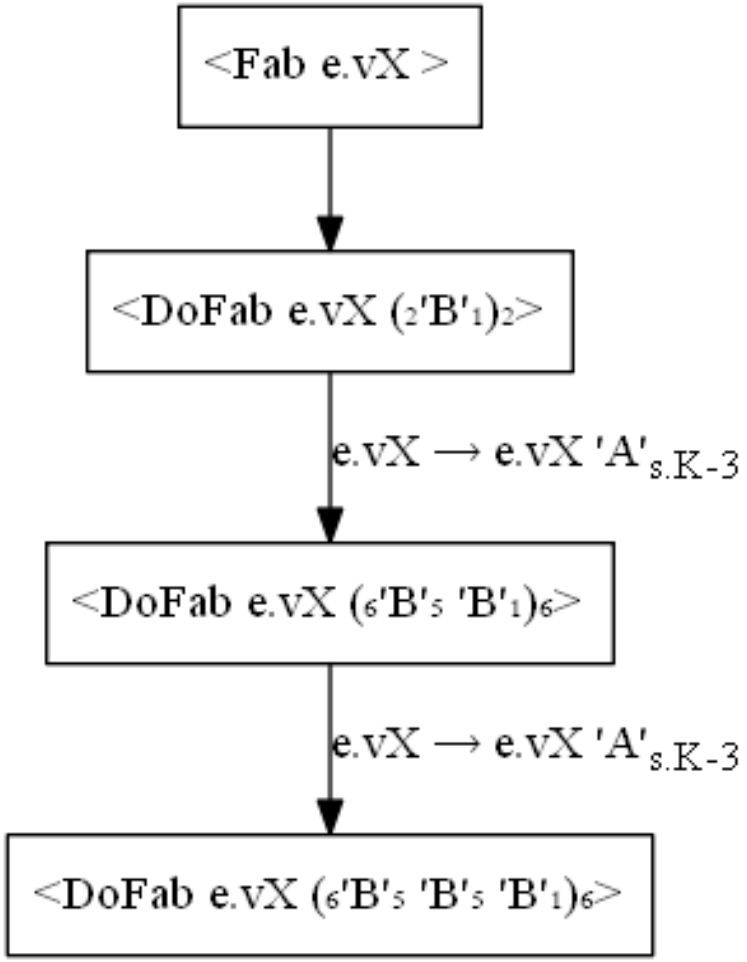
Остаточная программа

```
$ENTRY Fab {
 e.41 'A' =  <F84 (e.41) 'B'> 'B';
 e.41 s.101 =  <F84 (e.41) s.101> 'B';
 =  'B';
}

/*
*  InputFormat: <F84 (e.133 ) e.134 >
*  OutputFormat: ==> e.149
*/
F84 {
 (e.133 'A') e.134 =  <F84 (e.133) 'B' e.134>;
 (e.133 s.135) e.134 =  <F84 (e.133) s.135 e.134>;
 () e.134 =  e.134;
}
```

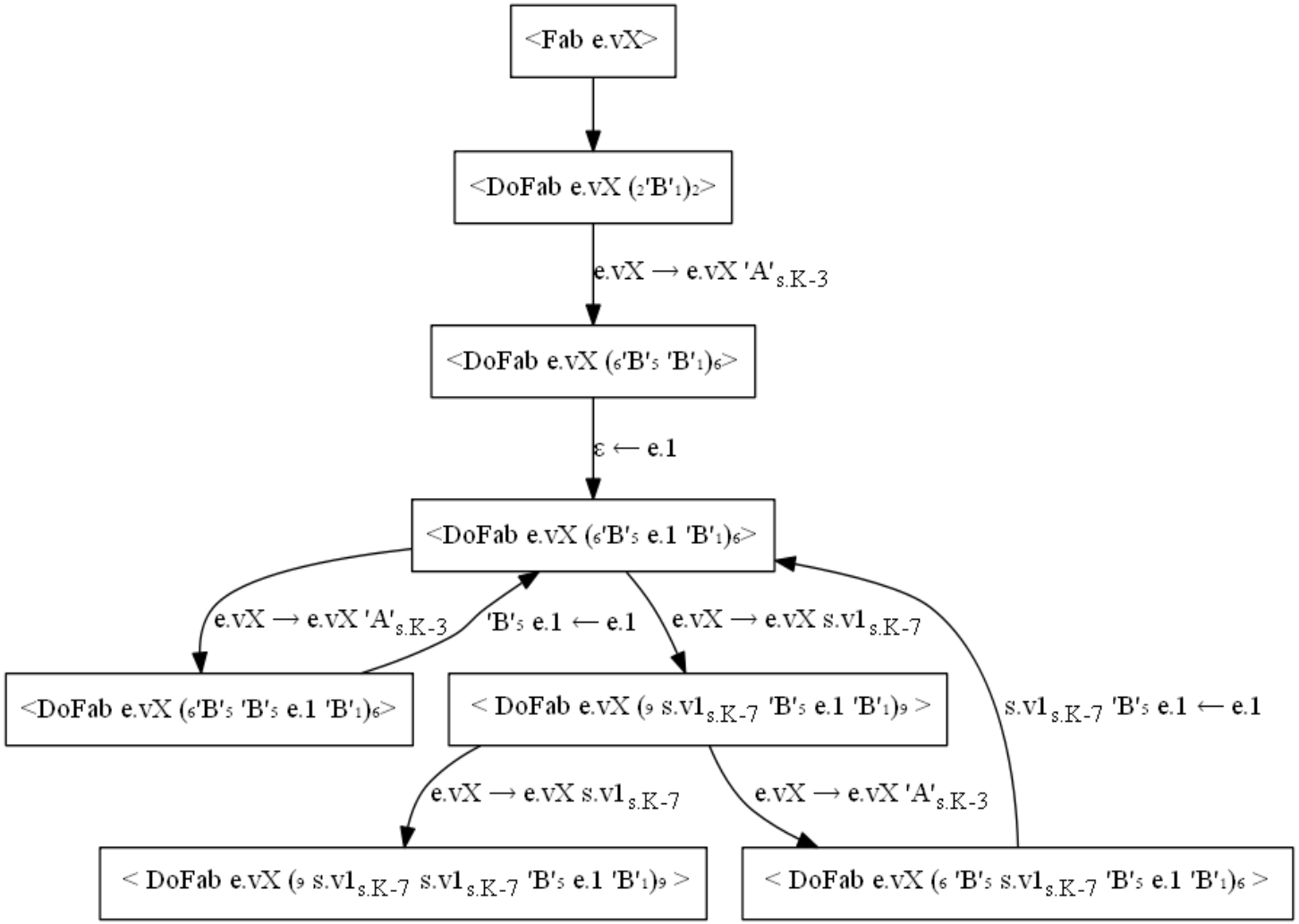
Суперкомпиляция после обработки вторым препроцессором

- Для сокращения записи введём следующие обозначения:
- (Sym K-1) будем записывать как Sym_1
- ((e.Expr) K-2) как $({}_2\mathrm{e.Expr})_2$
- (s.Var s.K-3) как $\mathrm{s.Var}_{\mathrm{s.K}-3}$

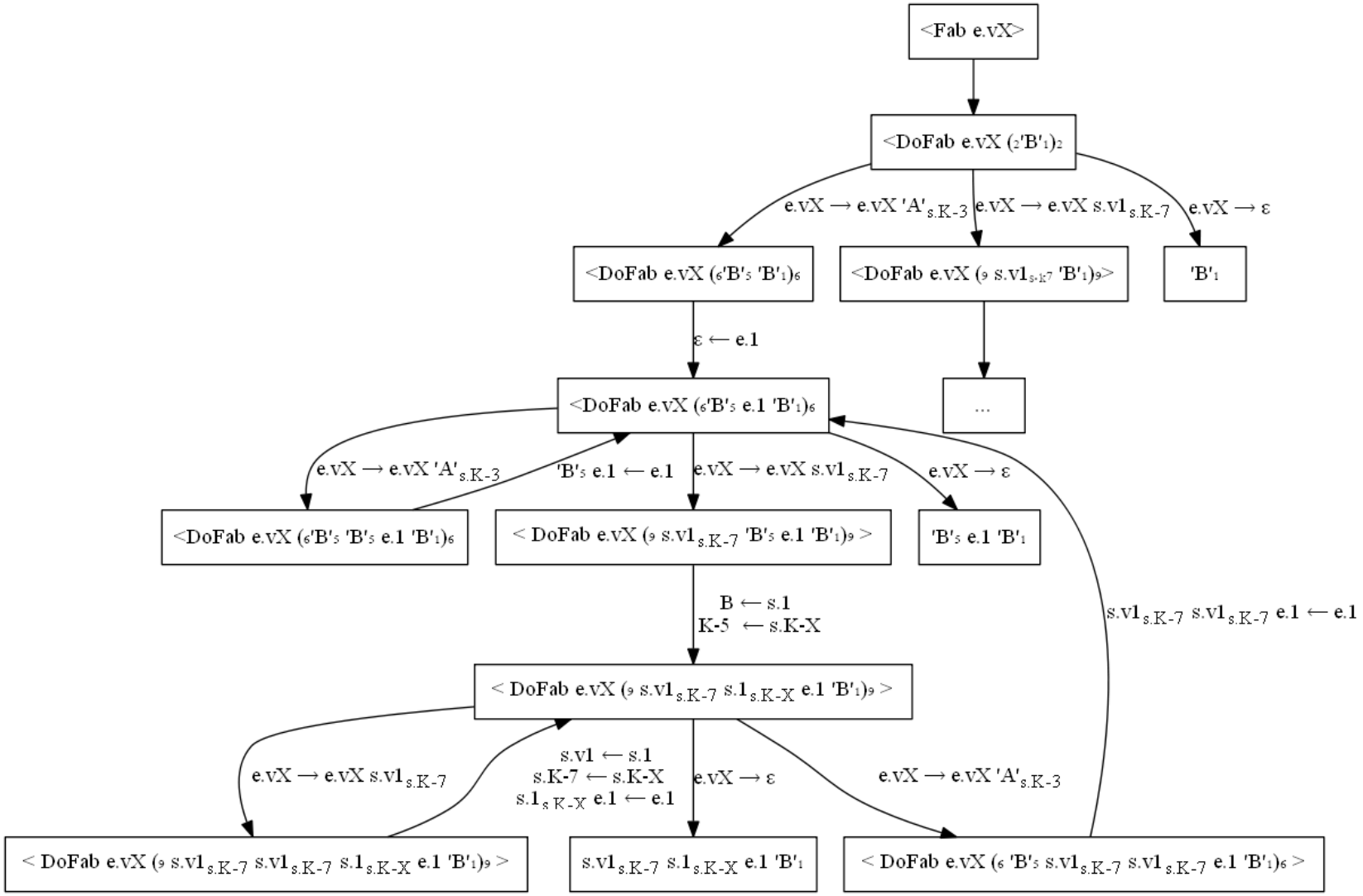
Суперкомпиляция после обработки вторым препроцессором



Суперкомпиляция после обработки вторым препроцессором



Суперкомпиляция после обработки вторым препроцессором



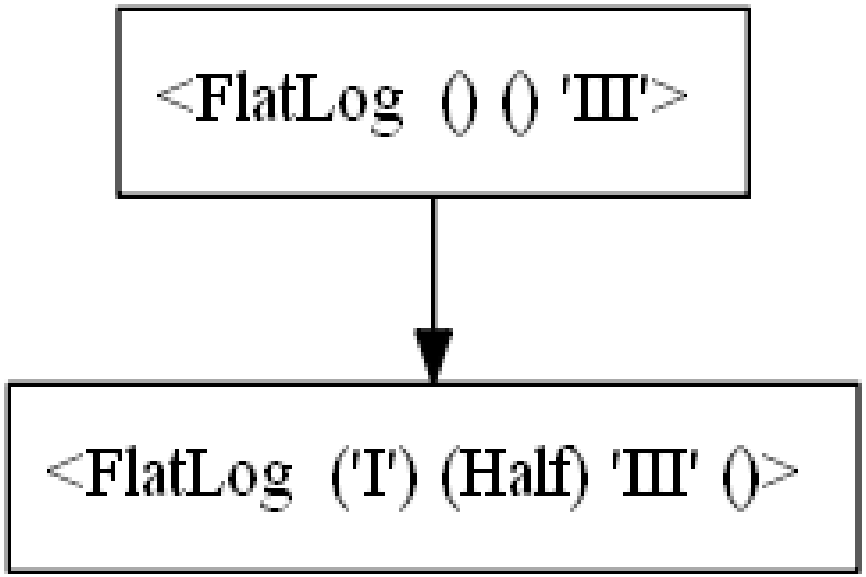
Остаточная программа

```
$ENTRY Fab {
 e.41 ('A' s.103) =  <F23 (e.41)>;
 e.41 ('A' s.147) (s.102 s.103) =  <F23 (e.41) (s.102 s.103)>;
 e.41 (s.146 s.147) (s.102 s.103) =  <F49 (e.41) s.146 s.147 s.103 s.102>;
 (s.102 s.103) =  (s.102 s.103) ('B' K-1);
 =  ('B' K-1);
}

/*
*  InputFormat: <F49 (e.128 ) s.131 s.132 s.133 s.134 e.135 >
*  OutputFormat: ==> e.0
*/
F49 {
 (e.128 ('A' s.138)) s.131 s.132 s.133 s.134 e.135
          =  <F23 (e.128) (s.131 s.132) (s.134 s.133) e.135>;
 (e.128 (s.137 s.138)) s.131 s.132 s.133 s.134 e.135
          =  <F49 (e.128) s.137 s.138 s.132 s.131 (s.134 s.133) e.135>;
 () s.131 s.132 s.133 s.134 e.135 =  (s.131 s.132) (s.134 s.133) e.135 ('B' K-1);
}

/*
*  InputFormat: <F23 (e.110 ) e.112 >
*  OutputFormat: ==> e.0
*/
F23 {
 (e.110 ('A' s.115)) e.112 =  <F23 (e.110) ('B' K-5) e.112>;
 (e.110 (s.114 s.115)) e.112 =  <F49 (e.110) s.114 s.115 K-5 'B' e.112>;
 () e.112 =  ('B' K-5) e.112 ('B' K-1);
}
```

Программа вычисления целочисленного логарифма

```
$ENTRY Go {
  = <FlatLog  () () 'III'>
}

FlatLog {
  (e.Acc) () /* empty */ = e.Acc;
  (e.Acc) () 'I' e.X
       =  <FlatLog (e.Acc 'I') (Half)  'I' e.X ()>;
  (e.Acc) (Half) e.X 'II' (e.Res)
       = <FlatLog (e.Acc) (Half) e.X (e.Res 'I')>;
  (e.Acc) (Half) e.X (e.Res) = <FlatLog (e.Acc) () e.Res>;
}
```

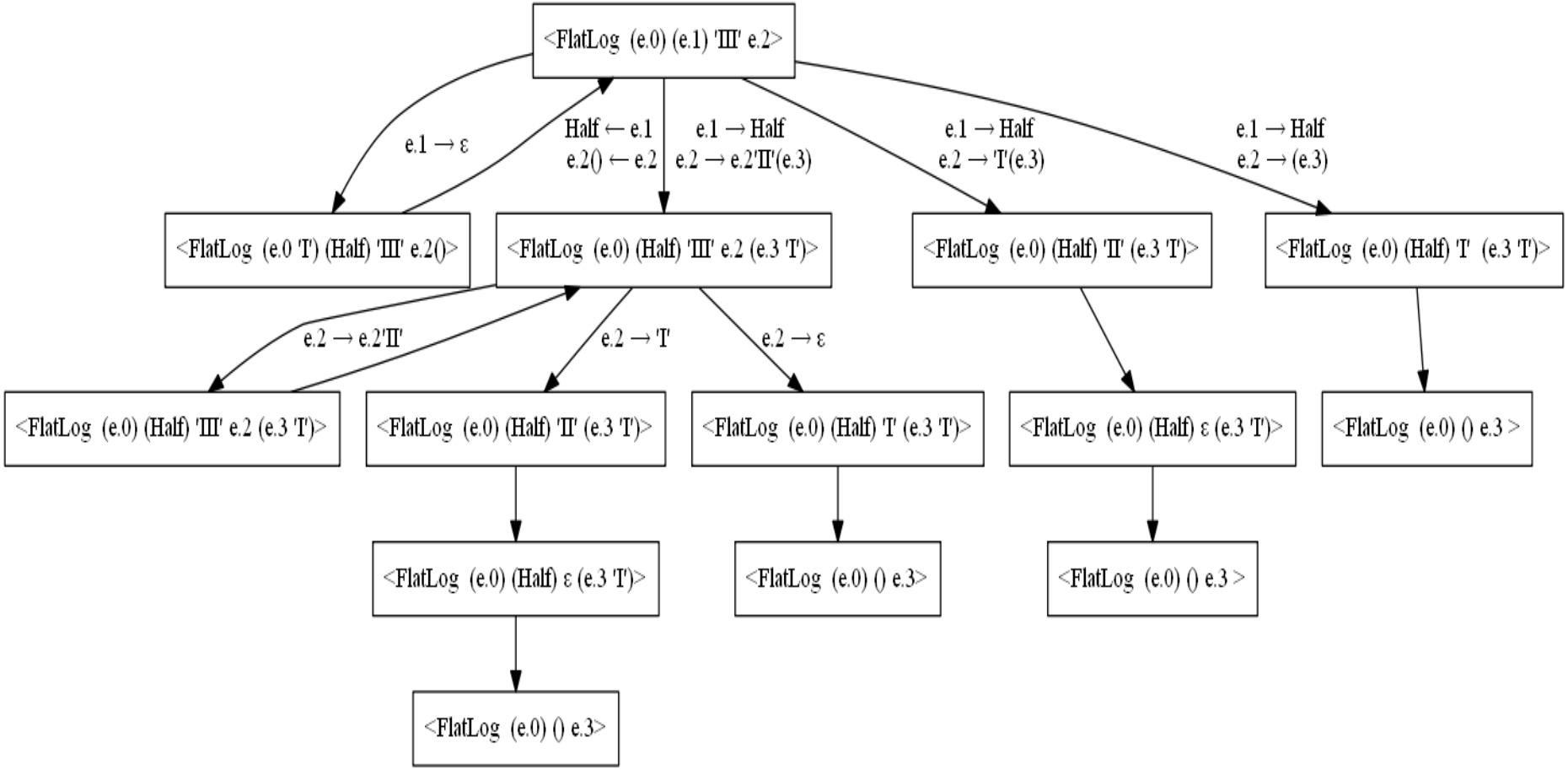
Суперкомпиляция исходной программы



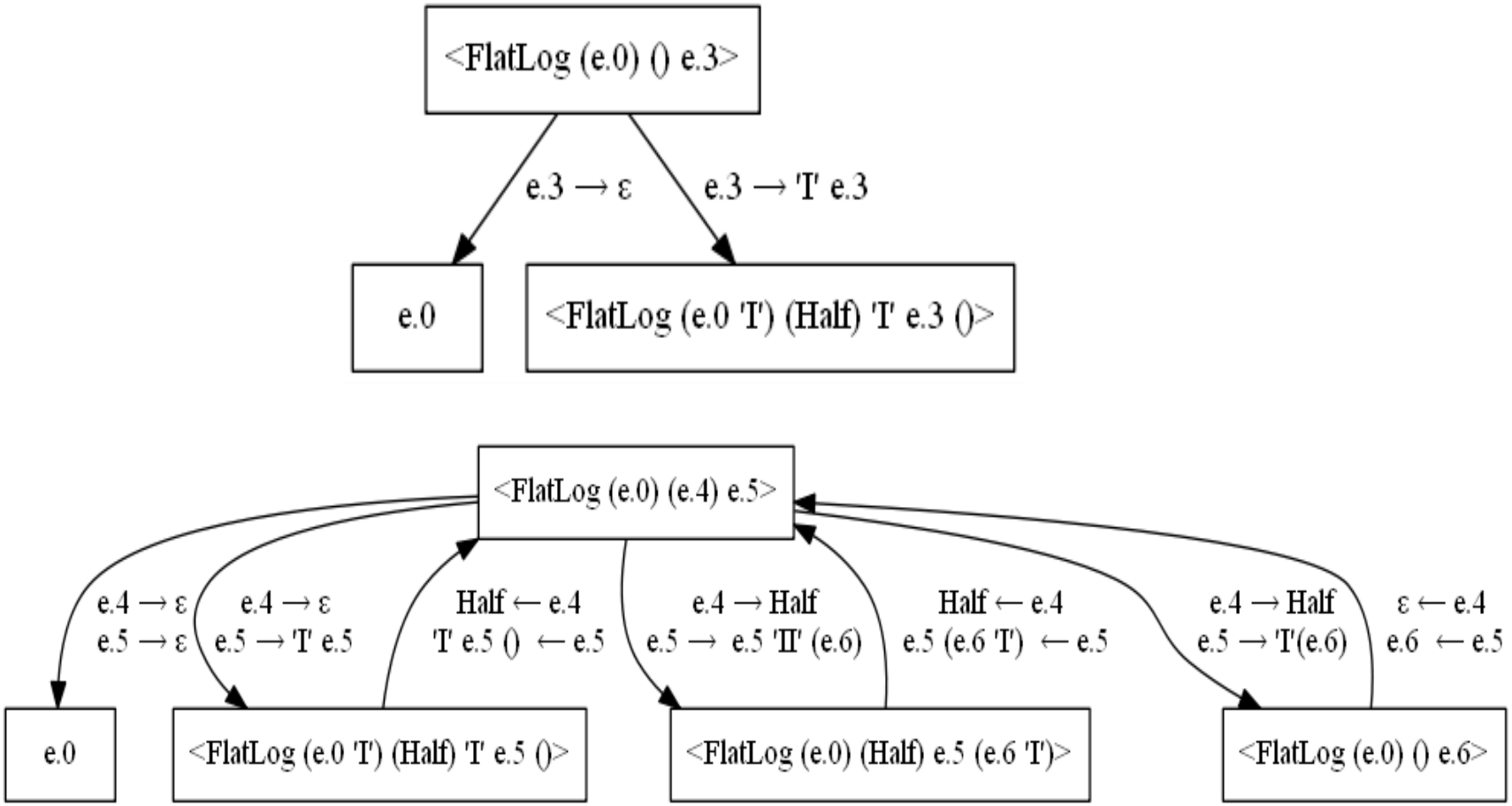
Суперкомпиляция исходной программы

- Конфигурации *<FlatLog () () 'III'>* и *<FlatLog ('I') (Half) 'III' ()>* похожи
- Их обобщение: *<FlatLog (e.0) (e.1) 'III' e.2>*

Суперкомпиляция исходной программы



Суперкомпиляция исходной программы



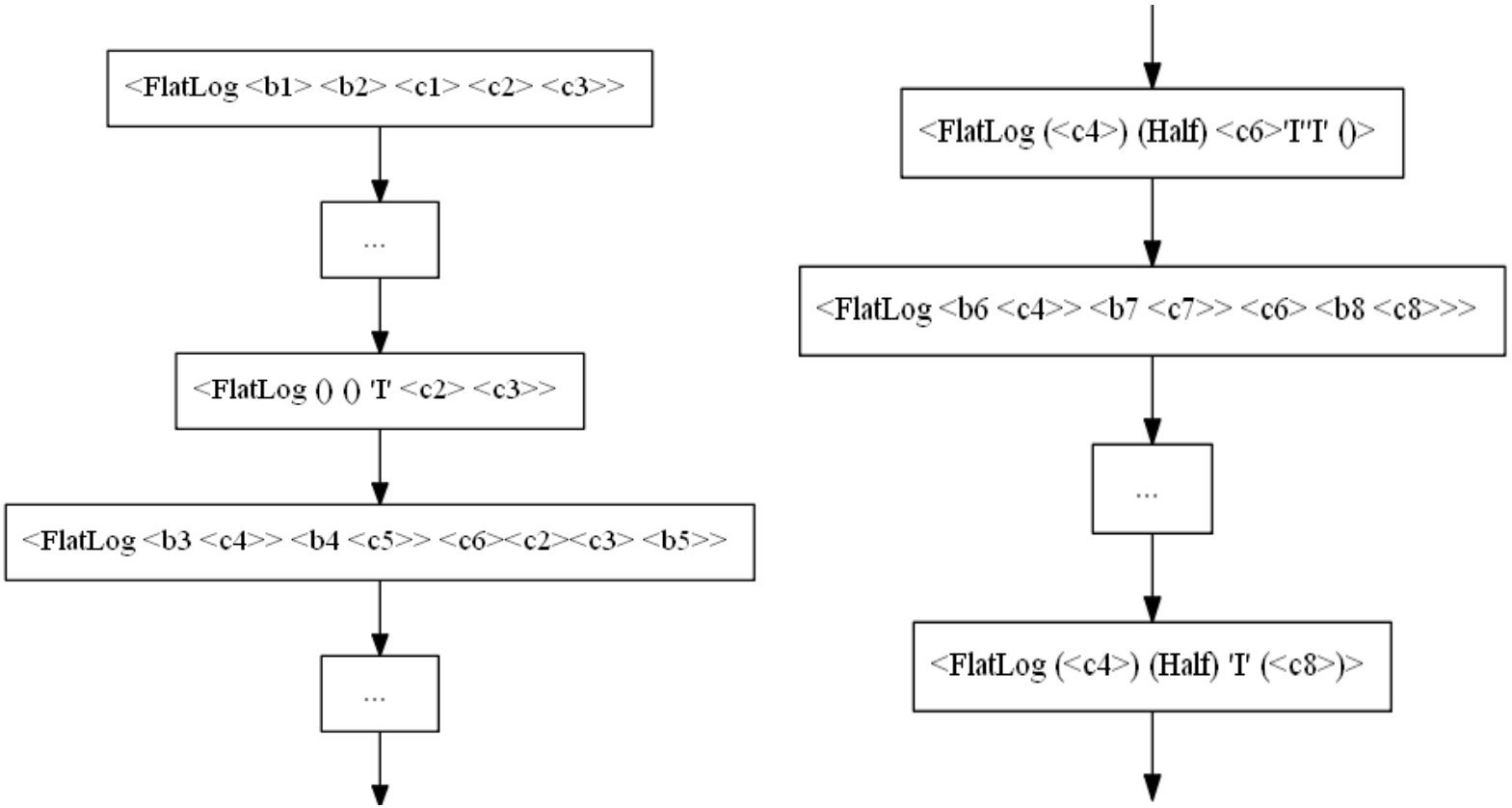
Остаточная программа

```
$ENTRY Go {
 =  <F122 () () 'III'>;
}

/*
*  InputFormat: <F122 (e.113 ) (e.114 ) e.115 >
*  OutputFormat: ==> e.127
*/
F122 {
 (e.113) () =  e.113;
 (e.113) () 'I' e.115 =  <F122 (e.113 'I') (Half) 'I' e.115 ()>;
 (e.113) (Half) e.115 'II' (e.118)
        =  <F122 (e.113) (Half) e.115 (e.118 'I')>;
 (e.113) (Half) e.115 (e.118) =  <F122 (e.113) () e.118>;
 }
```

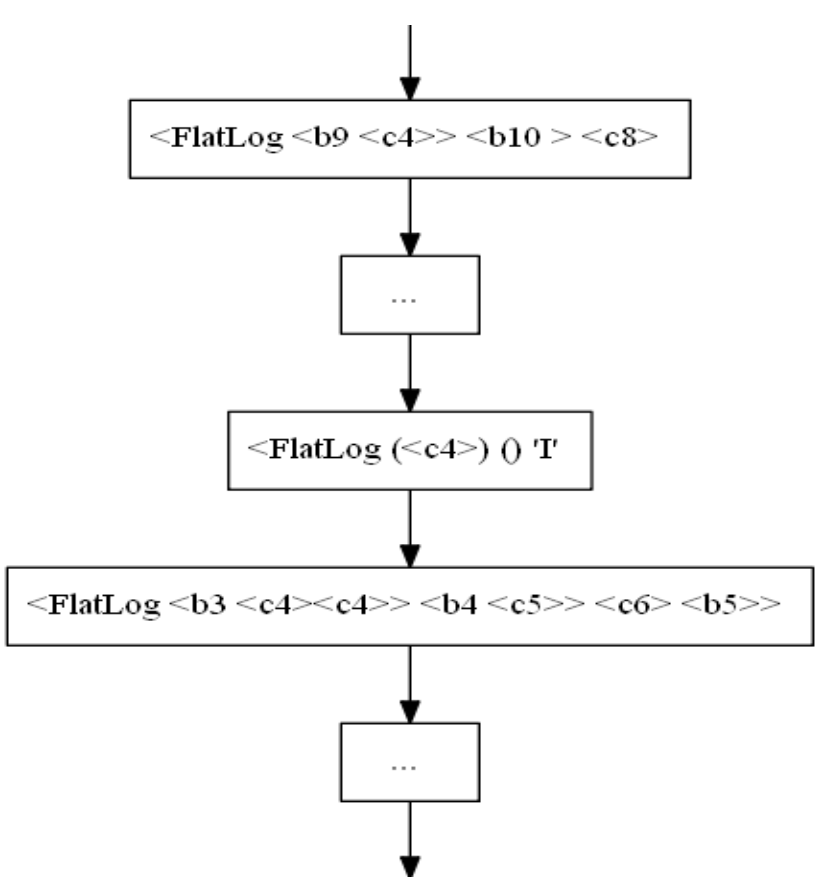
Программа, обработанная первым препроцессором

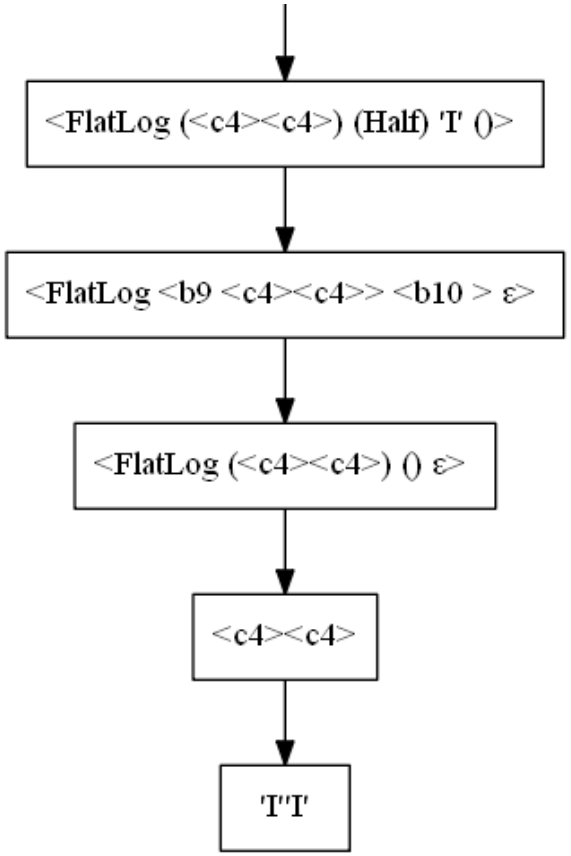
```
$ENTRY Go {
  = <FlatLog  <b1 > <b2 > <c1><c2><c3>>
}

FlatLog {
  (e.Acc) () /* empty */ = e.Acc;
  (e.Acc) () 'I' e.X
       =  <FlatLog <b3 e.Acc <c4>> <b4 <c5>>  <c6> e.X <b5 >>;
  (e.Acc) (Half) e.X 'I''I' (e.Res)
       = <FlatLog <b6 e.Acc> <b7 <c7>> e.X <b8 e.Res <c8>>>;
  (e.Acc) (Half) e.X (e.Res) = <FlatLog <b9 e.Acc> <b10 > e.Res>;
}
b1 { e.X = (e.X);}
…
b10 { e.X = (e.X);}

c1 { = 'I';}
c2 { = 'I';}
c3 { = 'I';}
c4 { = 'I';}
c5 { = Half;}
c6 { = 'I';}
c7 { = Half;}
c8 { = 'I';}
```

Суперкомпиляция после обработки первым препроцессором



Суперкомпиляция после обработки первым препроцессором

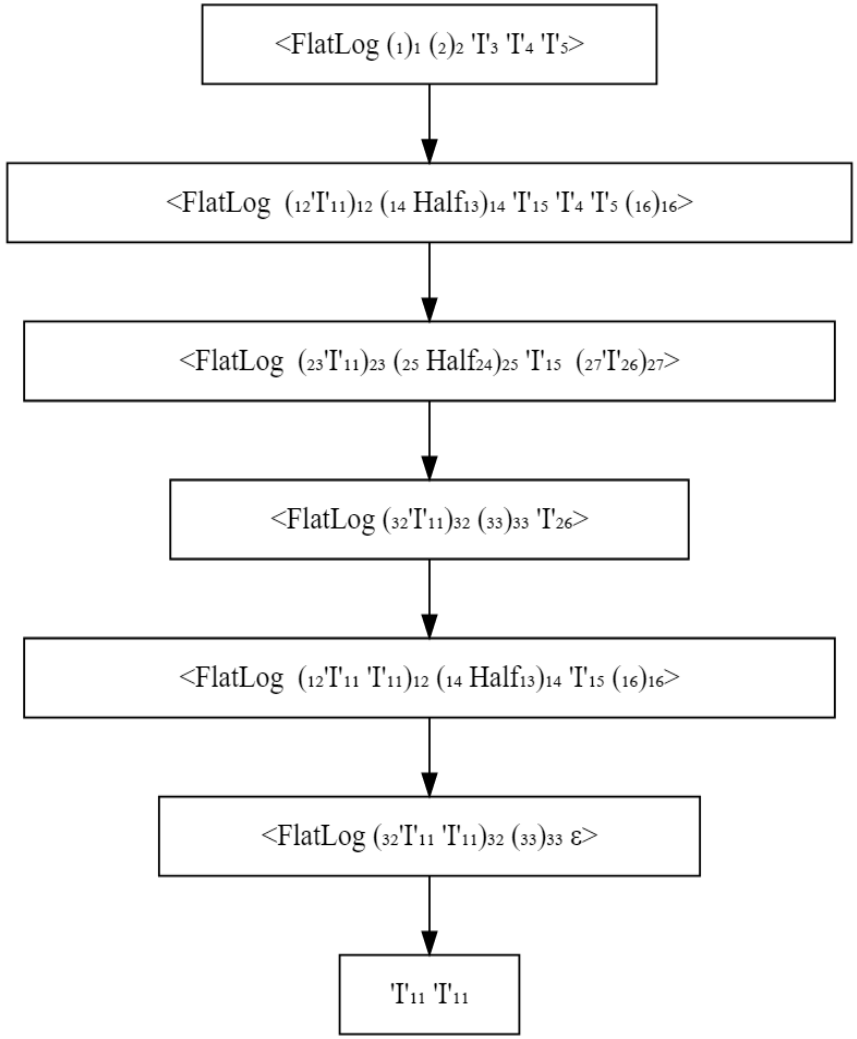




Программа, обработанная вторым препроцессором

```
$ENTRY Go {
  = <FlatLog  (() K-1) (() K-2) ('I' K-3)('I' K-4)('I' K-5)>
}

FlatLog {
  ((e.vAcc) s.K-6) (() s.K-7) /* empty */ = e.vAcc;
  ((e.vAcc) s.K-8) (() s.K-9) ('I' s.K-10) e.vX
        =  <FlatLog ((e.vAcc ('I' K-11)) K-12)
                    (((Half K-13)) K-14)  ('I' K-15) e.vX (() K-16)>;

  ((e.vAcc) s.K-17) (((Half s.K-18)) s.K-19) e.vX
  ('I' s.K-20)('I' s.K-21) ((e.vRes) s.K-22)
        = <FlatLog ((e.vAcc) K-23)
                   (((Half K-24)) K-25) e.vX ((e.vRes ('I' K-26)) K-27)>;

  ((e.vAcc) s.K-28) (((Half s.K-29)) s.K-30) e.vX ((e.vRes) s.K-31)
        = <FlatLog ((e.vAcc) K-32) (() K-33) e.vRes>;
}
```

Суперкомпиляция после обработки вторым препроцессором



Остаточные программы

```
$ENTRY Go {
 =  'II';
}
```

```
$ENTRY Go {
 =  ('I' K-11) ('I' K-11);
}
```

Спасибо за внимание