Улучшение результатов суперкомпиляции SCP4 посредством использования промежуточного интерпретатора

Атымханова Мадина

МГТУ им. Н.Э.Баумана

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

11 июня 2019 года

Суперкомпиляция (supervising compilation)

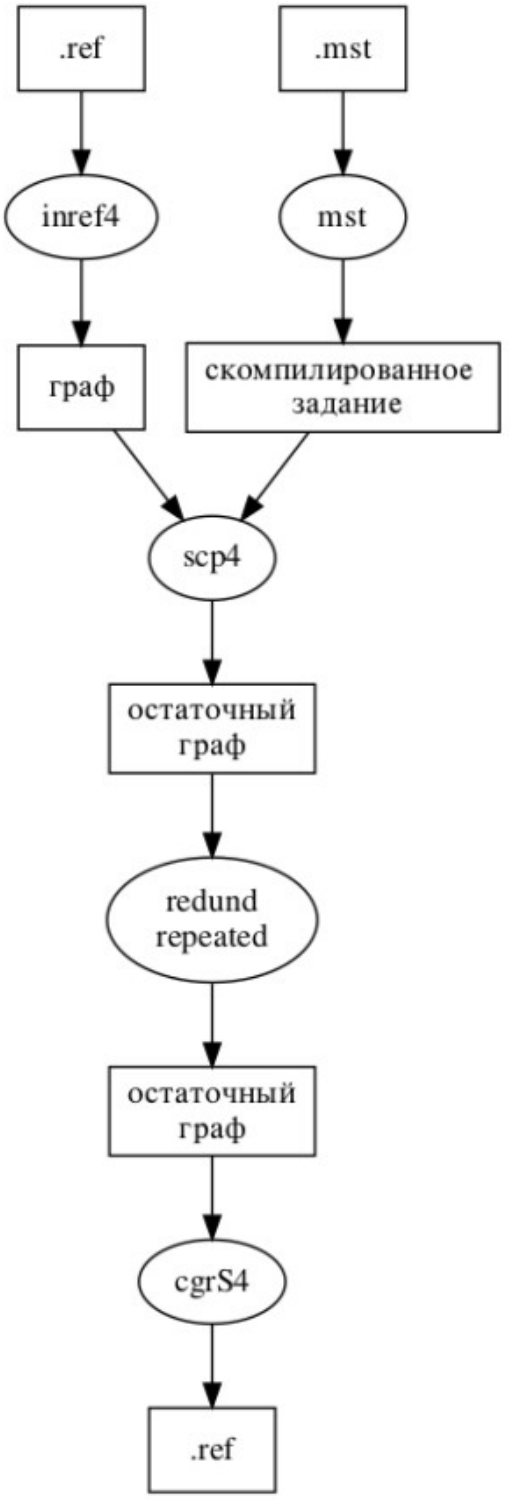
- это метод преобразования программ
- **Цель:** оптимизация программы и анализ.
- Суперкомпиляция преобразует исходные тексты программ, а затем с ней работает обычный компилятор (или интерпретатор).
- Техника преобразования программ, основанная на построении полной и самодостаточной модели программы.

Идея суперкомпиляции:

- Программе P_1 ставится в соответствие машина M_1, моделирующая в общем виде выполнение программы P_1.
- Контролируя и наблюдая (SUPERvises) работу машины M_1, суперкомпилятор создает (компилирует, COMPILEs) другую машину M_2, которая полностью описывает M_1.
- Машина M_2 далее может быть представлена в виде программы P_2.

SCP4

- Используемый в ВКР суперкомпилятор - суперкомпилятор SCP4 для языка РЕФАЛ-5.



Первая проекция Футамуры 1

- P_R (d1, d2) — программа, на языке R, которая принимает два аргумента.
- Специализатор:

  $SPEC_R$ (P_R , d1) = $P_{R,d1}$

  порождает программу

  $P_{R,d1}$(d2') ≡ P_R (d1, d2')
- $SPEC_R$ (P_R , d1)(d2) = P_R (d1, d2)

Первая проекция Футамуры 2

Интерпретатор INT_R (P_S , d) — программа на языке R, которая принимает текст программы на языке S, и входное данное для этой программы, - тот же результат, что и непосредственное выполнение P_S (d) на машине языка S.

$[[INT_R (P_S , d)]] = [[P_S (d)]]$

- Заменим: P_s — на INT_R , d1 — на P_S

  $SPEC_R (INT_R , P_S)(d) = INT_R (P_S , d) = P_S (d)$

- $SPEC(INT_R , P_S)$ — это некоторая программа на языке R (для машины R):

  $INT_{R,PS}(d') = INT_R (P_S , d') = P_S (d')$

- Т.е. результат специализации интерпретатора — это программа P_S скомпилированная для машины R:

  $P_R (d') \equiv INT_{R,PS} (d') = INT_R (P_S , d') = P_S (d')$

Вторая и третья проекции Футамуры

- Сам специализатор является программой на языке R от двух аргументов. А значит, его тоже можно специализировать:

  $P_S\ (d) = \ INT_R\ (P_S\ ,\ d) = SPEC_R\ (INT_R\ ,\ P_S\ )\ (d) =$

  $= SPEC\_R\ (SPEC\_R\ ,\ INT\_R\ )\ (P\_S\ )\ (d) =$

  $= SPEC\_R\ (SPEC\_R\ ,SPEC\_R\ )\ (INT\_R\ )\ (P\_S\ )\ (d)$
- Вторая проекция:

  Программа, которая принимает программу на языке S и формирует эквивалентную программу на языке R - это компилятор.

  $P_R = SPEC_R\ (INT_R\ ,\ P_S\ ) = SPEC_R\ (SPEC_R\ ,\ INT_R\ )\ (P_S\ ) = COMP$
- Третья проекция — компилятор компиляторов, программа, которая превращает интерпретаторы в компиляторы:

  $COMP_{R,S} = SPEC_R\ (SPEC_R\ ,\ INT_R\ ) = SPEC_R\ (SPEC_R\ ,\ SPEC_R\ )\ (INT_R\ ) = COMCOM_R\ (INT_R\ )$

Цель работы:

суперкомпилировать суперкомпилятором SCP4 написанный в ходе работы над ВКР интерпретатор промежуточного представления программы в формате и на языке, входных для суперкомпилятора SCP4.

Ожидаемый результат

- Корректное исполнение первой проекции Футамуры
- Изменение операционной семантики входной программы

Интерпретатор графов, пригодный для суперкомпиляции

- Компоненты:

- Таблица переменных
- Граф
- Информация для отката (стек)

Возможные конфигурации в интерпретируемом графе:

- Если граф имеет вид "сужение"(constriction), то из таблицы переменных выбирается сужаемая переменная и проверяется удовлетворяет ли она сужению.
- Если удовлетворяет - применяется сужение, это изменяет таблицу.
- Иначе выполняется откат.
- Если граф имеет вид "ветвления"(forking), то берется первая ветвь, остальные кладутся в список отката.
- Если граф имеет вид "рестрикции"(restriction), то проверяется рестрикция, иначе выполняется откат.
- Если граф имеет вид "конфигурация"(configuration), то подставляются переменные в выражение, выбирается вызов функции. Если вызов функции присутствует, то он выполняется и рекурсивно вызывается интерпретатор. Иначе возвращается значение выражения.
- "откат" - берется очередная ветвь Fork и выполняется.

Суперкомпилируется интерпретатор int.ref с заданием int.mst

```
$DEFINE Prog
<Inref4
  Trim {
       ' ' e.X = <Trim e.X>;
       e.X ' ' = <Trim e.X>;
       e.X = e.X;
    }
>;

<Int        e.X    (         ) > --
     <Trim  !   >   %.Prog     ;
```

Суперкомпиляция функции Trim

```
***
/*
*  InputFormat: <Go e.41 >
*  OutputFormat: ==> e.0
*/


$ENTRY Go {
 e.41 =  <F6 e.41>;
}

/*
*  InputFormat: <F6 e.41 >
*  OutputFormat: ==> e.101
*/
F6 {
 ' ' e.41 =  <F6 e.41>;
 e.41 ' ' =  <F6 e.41>;
 e.41 =  e.41;
}
```

Суперкомпиляция интерпретатора по программе с функцией Trim

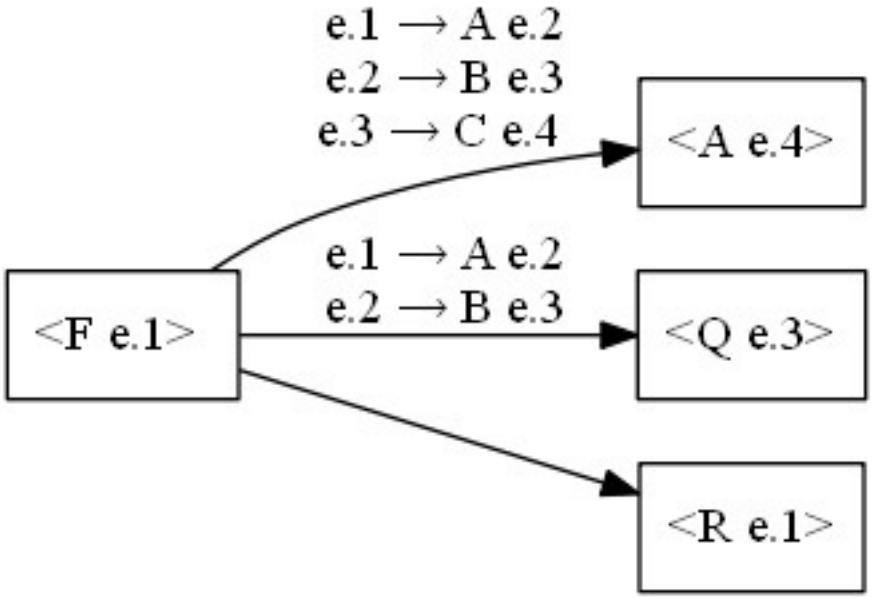
```
*$MST_FROM_ENTRY;

*$MATCHING ForRepeatedSpecialization ;

*$STRATEGY Applicative ;
/*
$ENTRY Go {
 = <Prout <int1 e.X >> ;
}
*/

***
/*
*  InputFormat: <int1 e.1 >
*  OutputFormat: ==> e.0
*/


int1 {
 e.1 =  <F74 e.1>;
}

/*
*  InputFormat: <F141 (e.1 ) s.144 >
*  OutputFormat: ==> s.191 (e.192 )
*/
F141 {
 (e.1 ' ') s.144, <F141 (e.1) s.144> : s.233 (e.234) =  s.233 (e.234);
 (e.1) s.144 =  s.144 (e.1);
}

/*
*  InputFormat: <F74 e.1 >
*  OutputFormat: ==> e.231
*/
F74 {
 ' ' e.1 =  <F74 e.1>;
 s.144 e.1, <F141 (e.1) s.144> : s.191 (e.192) =  s.191 e.192;
 e.1 ' ' =  <F74 e.1>;
 e.1 =  e.1;
}
```

Целевой пример

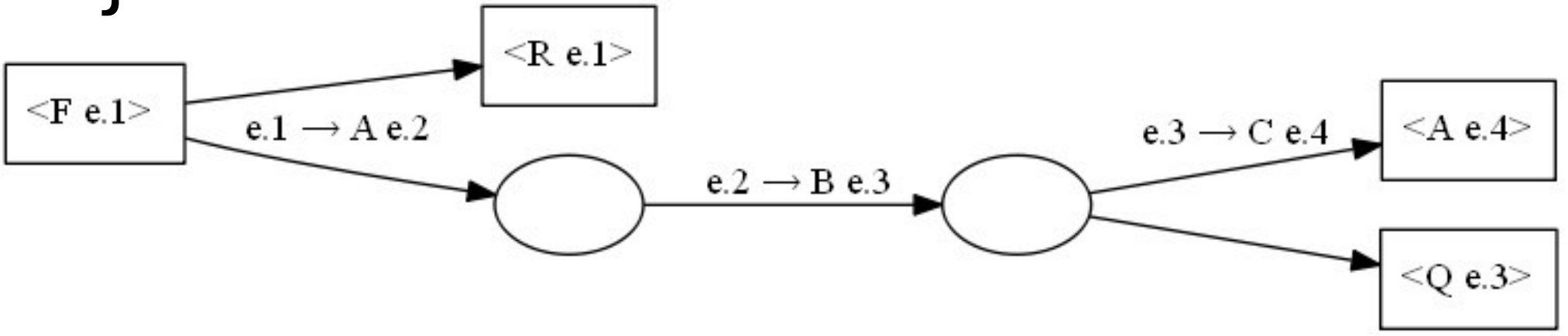
```
$DEFINE Prog
<Inref4
  Verif {
    e.X = <Pred <F e.X>>;
  }
  F {
    A B e.X = Hello;
    e.X = e.X;
  }
  Pred {
    A B e.X = False;
    e.X = True;
  }
>;


<Int All Lazy        e.X    (        ) > --
              <Verif  !   >   %.Prog     ;
```

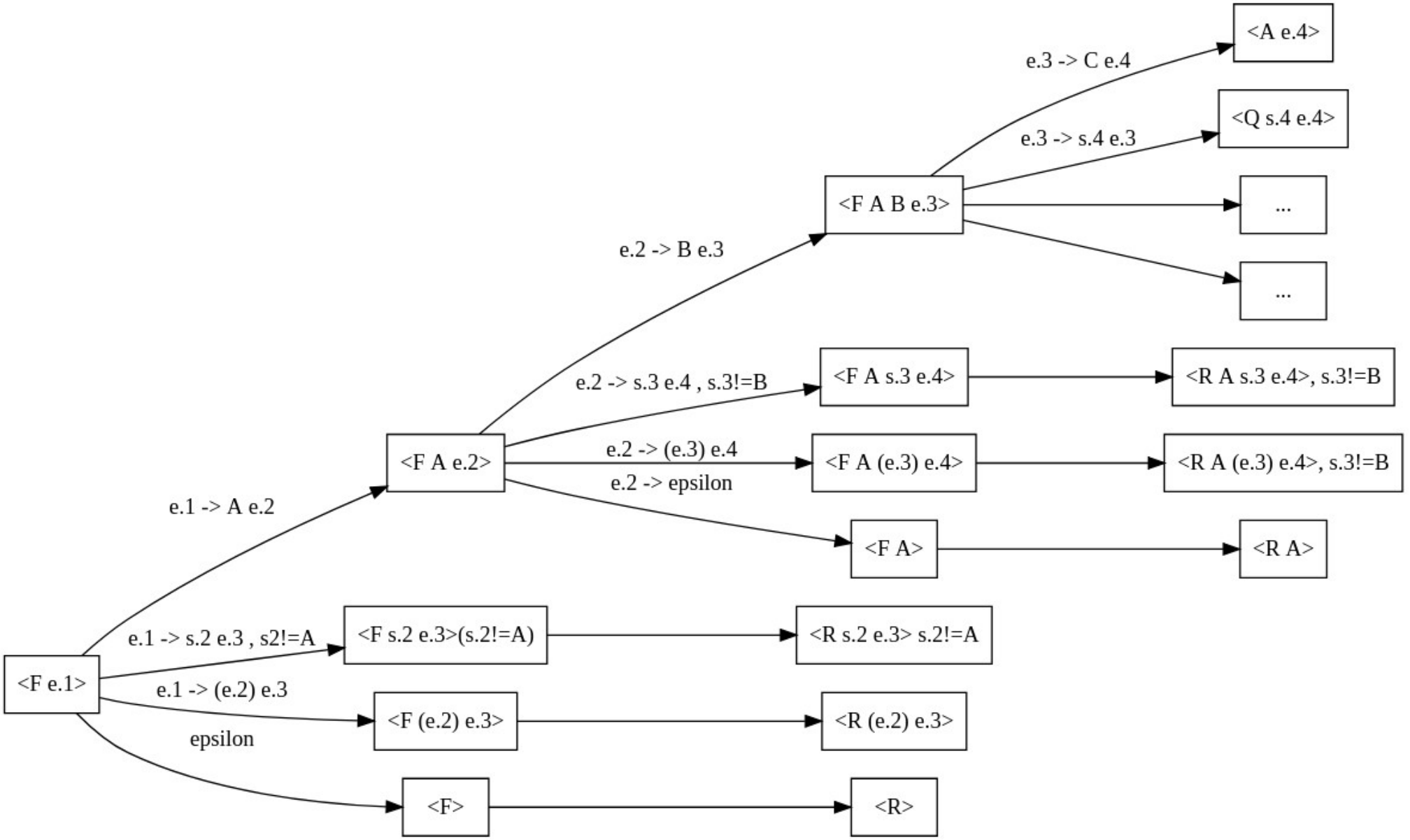
Целевой пример

```
$DEFINE Prog
<Inref4
  F {
     e.x = <A <A <A e.x > > >;
    }

  A {
      s.a s.a e.X = <A s.a e.X>;
      s.a s.b e.X = s.a <A s.b e.X>;
      e.X = e.X;
    }
>;


<Int    e.X    (        ) > --
     <F  !   >   %.Prog     ;
```




```
F {
   A B C e.X = <A e.x>
   A B e.X = <Q e.x>
   e.X = <R e.x>
}
```




<F e.1>
e.1 -> A e.2
e.1 -> s.2 e.3 , s2!=A
e.1 -> (e.2) e.3
epsilon
<F A e.2>
e.2 -> B e.3
e.2 -> s.3 e.4 , s.3!=B
e.2 -> (e.3) e.4
e.2 -> epsilon
<F s.2 e.3>(s.2!=A)
<F (e.2) e.3>
<F>
<F A B e.3>
<F A s.3 e.4>
<F A (e.3) e.4>
<F A>
e.3 -> C e.4
e.3 -> s.4 e.3
...
...
<A e.4>
<Q s.4 e.4>
<R s.2 e.3> s.2!=A
<R (e.2) e.3>
<R>
<R A s.3 e.4>, s.3!=B
<R A (e.3) e.4>, s.3!=B
<R A>