



Верифицированный спецификатор языка S-Graph

Владимир Гладштейн

По мотивам статьи Robert Glück, Andrei V. Klimov:
Occam's Razor in Metacomputation: the Notion of a Perfect
Process Tree



Задача

Инструменты

Н.У.О.

Решение



Задача

Начальные данные: S-Graph



```
Def ::= [Def]
Def  ::= (DEFINE Fname [Var] Tree)
Tree ::= (LET Var Exp Tree)
      | (CALL Fname [Exp])
      | (IF Cntr Tree Tree) | Exp
Cntr ::= (CONS? Arg Var Var)
      | (EQA? Arg Arg)
Exp  ::= Arg | (CONS Exp Exp)
Arg  ::= Val | Var
Val  ::= (ATOM Atom)
Var  ::= (VAR Name)
```



- $\text{int} :: \text{Tree} \rightarrow \text{Env} \rightarrow \text{Exp}$



- $\text{int} :: \text{Tree} \rightarrow \text{Env} \rightarrow \text{Exp}$
- $\text{int_Prog} :: \text{Def} \rightarrow \text{Prog} \rightarrow [\text{Exp}] \rightarrow \text{Exp}$



- $\text{int} :: \text{Tree} \rightarrow \text{Env} \rightarrow \text{Exp}$
- $\text{int_Prog} :: \text{Def} \rightarrow \text{Prog} \rightarrow [\text{Exp}] \rightarrow \text{Exp}$

Пример:

```
int_Prog  main  (main : p)  [CONS 'A 'B]
```



- $\text{dev} :: \text{Tree} \rightarrow \text{CEnv} \rightarrow \text{Tree}$
- $\text{spec} :: \text{Def} \rightarrow [\text{Exp}] \rightarrow \text{Def}$



- $\text{dev} :: \text{Tree} \rightarrow \text{CEnv} \rightarrow \text{Tree}$
- $\text{spec} :: \text{Def} \rightarrow [\text{Exp}] \rightarrow \text{Def}$
-

$\text{int_Prog} (\text{spec main } e_1) (\text{main} : p) e_2 =$
 $\text{int_Prog main (main} : p) (e_1 \mathrel{++} e_2)$

- программа после `dev` не (намного) сложнее



Инструменты

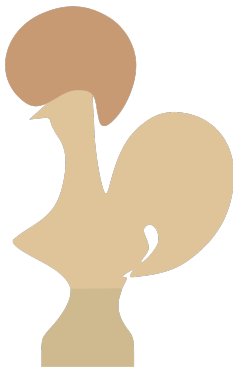


Figure 1: Coq Proof Assistant



H.Y.O.



Было

```
Cntr ::= (CONS? Arg Var Var)
        | (EQA? Arg Arg)
```

Стало

```
Cntr ::= (CONS? Arg)
        | (EQA? Arg Arg)
Tree += (HT Var Var Var Tree)
```



- $\text{int} :: \text{Int} \rightarrow \text{Tree} \rightarrow \text{Env} \rightarrow \text{Exp}$
- $\text{int_Prog} ::$
 $\text{Int} \rightarrow \text{Def} \rightarrow \text{Prog} \rightarrow [\text{Exp}] \rightarrow \text{Exp}$

Меняем `dev` : свежие переменные



Было

- $\text{dev} :: \text{Tree} \rightarrow \text{CEnv} \rightarrow \text{Tree}$

Надо

- $\text{dev} :: \text{Var} \rightarrow \text{Tree} \rightarrow \text{CEnv} \rightarrow \text{Tree}$



Было

```
dev (CALL fn as) e = dev t (mkEnv vs as e)
  where (DEFINE _ vs t) = getDef fn
```

Стало

```
dev f k t@(CALL fn as) e = f k t e
```




Решение



```

caller ::
  (Var -> Tree -> Cenv -> Tree) ->
  Var -> Tree -> Cenv -> Tree
caller f k (CALL fn as) (env, rs) =
  f (max k (maxvar t))
  t
  (mkEnv vs [x // env | x <- as], rs)
where (DEFINE _ vs t) = getDef fn

```



Definition (Correct transformation)

Будем говорить, что функция

$$f :: \text{Int} \rightarrow \text{Tree} \rightarrow \text{CEnv} \rightarrow \text{Tree}$$

корrekта, если, при определенных условиях,
выполняется

$$\text{int } n \ t \ (\text{comp } e_1 \ e_2) =$$

$$\text{int } n \ (f \ k \ t \ (e_1, \text{rs})) \ e_2$$

где $t = \text{CALL } \text{fn } \text{as}$



```
id_call k (CALL g args) c:=  
  let (e, _) = c in  
    CALL g [x // e | x <- args]
```



- `correct id_call`
- `correct f \rightarrow correct (caller (dev f))`
- `correct (caller (dev id_call))`
- $\forall n, \text{correct (devn } n)$
- `specn :: Int \rightarrow Def \rightarrow [Exp] \rightarrow Def`



Theorem (Корректность спецификатора)

*при определенных условиях, для любого m
выполняется*

$$\text{int_Prog } n \text{ main } (\text{main} : p) (e_1 \dashv\vdash e_2) = \\ \text{int_Prog } (n + 1) (\text{specn } m \text{ main } e_1) (\text{main} : p) e_2$$

Пример: Брюки превращаются



```
Definition eq_str : Def :=
  DEFINE eqStr [:: x1; x2] (
    Case x1 Of
    | h1 & t1 -->
      Case x2 Of
      | h2 & t2 -->
        If h1 =? h2 Then
          eqStr $ [:: (t1 : Exp); (t2 : Exp)]
        Else NO
      | ATOM --> NO
    | ATOM -->
      If (CONS? x2) Then
        NO
      Else If x1 =? x2 Then
        YES
      Else NO).
```

Превращаются...



К сожалению код не исполняемый, но
помучавшись можно доказать, что
`spesn 2 eq_tree [:: CONS '3 '4] =`

В элегантные шорты



```
DEFINE eqStr [:: x2] (  
  Case x2 Of  
  | h1' & h2' -->  
    If '3 =? h1' Then  
      If CONS? h2' Then NO  
      Else If '4 =? h2' Then YES  
      Else NO  
    Else NO  
  | ATOM --> NO).
```