Compilação de um Cálculo Lambda Estendido para Supercombinadores

Marco Rodrigo Costa

Orientadora: Mariza A. S. Bigonha

Co-orientador: Roberto S. Bigonha

Belo Horizonte, 06 de junho de 2000

Roteiro da Apresentação

- O Trabalho Proposto
- Contribuições
- Contexto do Trabalho
- Revisão da Literatura
- Compilador *LAMB* → *SUPER*
  - Esquema ***LL***
    - Esquema ***P***
    - Esquema ***L***

Roteiro da Apresentação...

- Resultados
- Conclusão
- Trabalhos Futuros

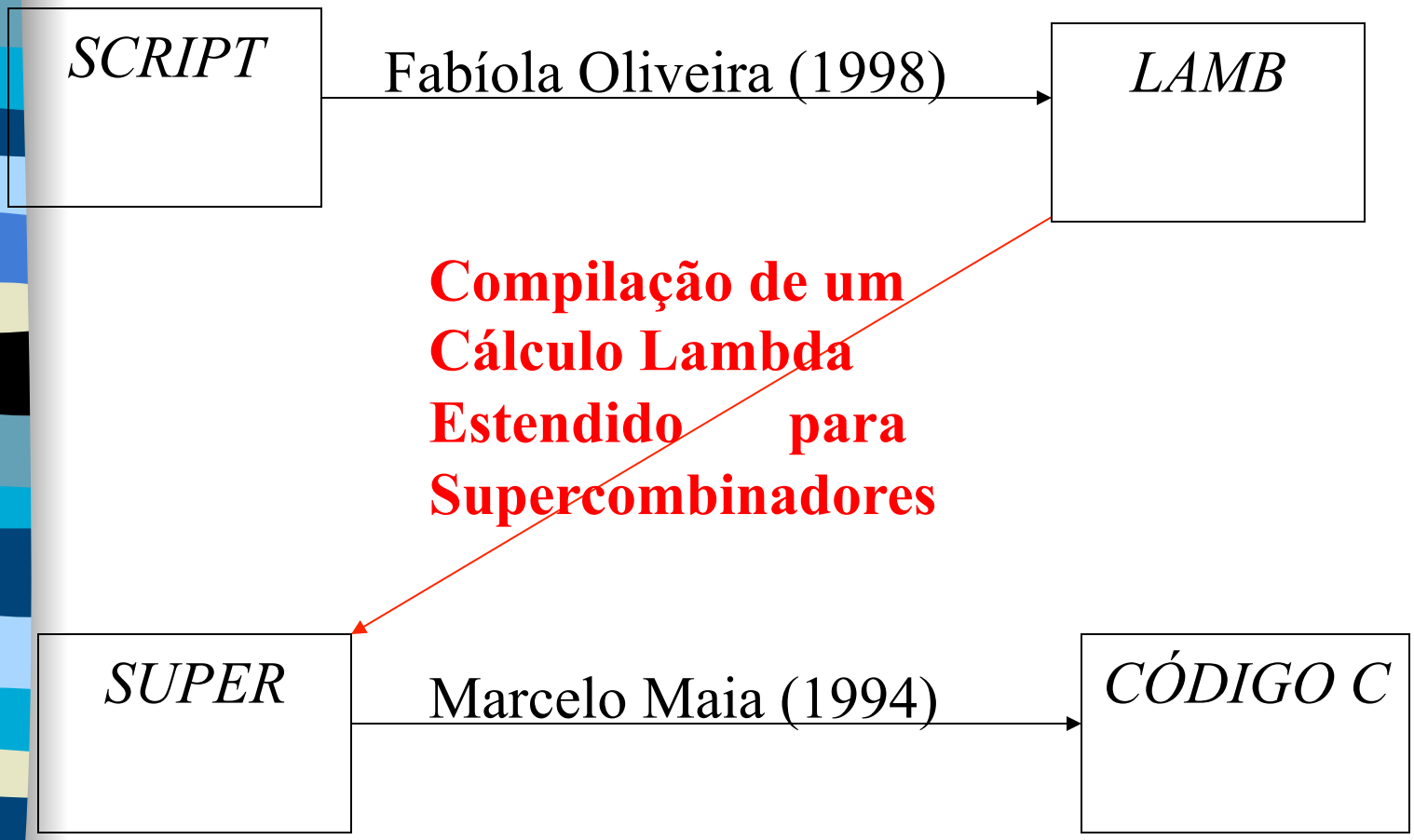
O Trabalho Proposto

- Problemas levantados em relação a *LAMB*
  - Abstrações lambda apresentam variáveis livres
  - As reduções de funções ocorrem sempre com um único argumento via abstrações lambda
- Objetivo: compilar programas de *LAMB* para *SUPER*

Contribuições

- Produção de uma etapa do compilador da linguagem *SCRIPT - lambda lifting*
- Estudo e desenvolvimento de técnicas de compilação para linguagens funcionais

Contexto do Trabalho



Revisão da Literatura

- Cálculo Lambda - *LAMB*
- Abordagens para tradução do Cálculo Lambda
  - Máquina SECD e *closures*
  - Proposta de David Wakeling: Máquina-X
  - Combinadores
  - Supercombinadores
  - Proposta de Peyton Jones e David Lester

Compilador *LAMB* → *SUPER*

- Esquemas de Compilação
  - Esquema *LL*
    - Esquema *P*
    - Esquema *L*

Esquema *LL*

- Função que mapeia expressões *LAMB* para *SUPER*:

$$LL : \text{ExpL} \rightarrow \text{ExpS}$$

  - Presença de padrões pressupõe esquema intermediário ⇒ composição de funções:
    - esquema *P*
    - esquema *L*

Esquema *P*

- Transforma os padrões de *LAMB*, estabelecendo nova linguagem: *LAMBASIC*

$$P : \text{ExpL} \rightarrow \text{ExpB}$$

Ocorrência de padrões na linguagem *LAMB*:

1- parâmetro de abstrações lambda

2- lado esquerdo de definições LET/DEF

3- parâmetro do operador IS

4- interno a outros padrões

Esquema *P*...

1- Padrões como parâmetro de abstrações lambda

...

Exemplo: Padrão Lista Vazia

P[LAM < >.exp] =

LET v = novoNome( )

IN [(LAM] v [.$EMPTY] v [→] *P*[exp] [,?)]

...

Aplicação:

LAMB: LAM < >.7 PLUS 3

LAMBASIC: (LAM nn1.$EMPTY nn1 → 7 PLUS 3, ?)

Esquema *P*...

2- Padrões no lado esquerdo de definições LET/DEF

P[defn_list IN exp] = *P*[defn_list] *P*[exp]

P[defn_list] = *P*[let_list]

P[let_list] = *P*[LET defn_a]

P[LET defn_a] = *P*[defn_a]

P[defn_a] = *P*[defn]

Esquema *P*...

2- padrão no lado esquerdo de definições LET/DEF...

P[defn] =

LET (left,right) = P_{PAIRS}[defn]

IN left NE ? → [LET] left [=] right [IN],

right [→] left [,]

P_{PAIRS}[defn] = P_{PAIRS}[pat = exp]

Esquema *P*...

2- Padrões no lado esquerdo de definições LET/DEF...

...

Exemplos: Constante e Identificador

P_{PAIRS}[k = exp] = < (?,[(k NE] *P*[exp] [)]) >

P_{PAIRS}[ide = exp] = < ([ide],*P*[exp]) >

...

Aplicações:

LAMB: LET 3 = 4 IN LET x = 2 IN x PLUS 1

LAMBASIC: (3 NE 4) → ?, LET x = 2 IN x PLUS 1

Esquema *P*...

3- Padrões como parâmetro do operador IS

...

Exemplos: Lista Vazia e Identificador

P[exp IS < >] = [(SIZE] *P*[exp] [EQ 0) → TT, FF]

P[exp IS ide] = [TT]

...

Aplicações:

LAMB: (<3,2> IS < >) AND (7 IS x)

LAMBASIC: ((SIZE <3,2> EQ 0) → TT, FF) AND (TT)

Esquema *L*

- Realiza o *lambda-lifting* efetivo

  L : ExpB → B → N→ ExpS X ExpS X Free

  – Exemplos de Esquemas *L*:

    1- Expressão com Operador Binário

    2- Expressão Condicional

    3- Abstração Lambda Atipada

Esquema *L*...

1- Expressão com Operador Binário

L [exp_1 op_di exp_2] b n =

LET (s_1,e_1,f_1) = L [exp_1] b n

LET (s_2,e_2,f_2) = L [exp_2] b n

IN (s_1 || s_2, built_in(op_di) || e_1 || e_2,$f_1 \cup f_2$)

Exemplo:

LAMBASIC: x PLUS y

SUPER: ([ ], PLUS x y, {x, y})

Esquema *L*...

2- Expressão Condicional

$L\ [exp_1 \rightarrow exp_2\ ,\ exp_3\ ]\ b\ n =$

LET $(s_1,e_1,f_1) = L\ [exp_1]\ b\ n$

LET $(s_2,e_2,f_2) = L\ [exp_2]\ b\ n$

LET $(s_3,e_3,f_3) = L\ [exp_3]\ b\ n$

IN $(s_1 \parallel s_2 \parallel s_3,\ [IF] \parallel e_1 \parallel e_2 \parallel e_3, f_1 \cup f_2 \cup f_3)$

Exemplo:

LAMBASIC: x EQ 0 → y, z

SUPER: ([ ], IF EQ x 0 y z, {x, y, z})

Esquema *L*...

3- Abstração Lambda Atipada

L [LAM ide.exp] b n =
LET b_1 = insert(b,(ide,n+1,?,?))
LET (s_1,e_1,f_1) = *L* [exp] (b_1) (n+1)
LET f = exclude(f_1,(ide,n+1))
LET nome = newSuperName( )
LET livres = makeList(f)
LET s = s_1 || genSuper(nome,livres,ide,e_1)
LET e = nome || livres
IN (s,e,f)

Esquema *L*...

3- Abstração Lambda Atipada...

Exemplo:

LAMBASIC: LAM x. LAM y. (x PLUS y) EQ 0 → 0, 1

SUPER: ([$Y x y = IF EQ (PLUS x y) 0 0 1

$X x = $Y x],

[$X],

{ })

Resultados

- Exemplo 1

```
-- SCRIPT

MODULE Teste

DOMAINS Fdom = (N,N) -> N;
                x:N;
                f:Fdom

 DEFINITIONS

DEF f = LAM (x,7). x PLUS 7

END Teste
```

Resultados...

- Exemplo 1...

-- **LAMB**

```
LET Teste  =
   LET f   =   LAM   ( x   , 7 ) . x PLUS 7
   IN  <  >
```

-- **LAMBASIC**

```
LET Teste  =
 LET f   = ( LAM nn12 . ( nn12 EL 2 )   NE 7   ->   ? ,
   (  LAM x . x  PLUS 7 )  ( nn12  EL 1  )  )
 IN  <  >
```

-- **SUPER**

```
$S26 x = PLUS x 7

$S27 nn12 = IF NE ( EL nn12 2 ) 7 ? AP ( $S26 ) ( EL
       nn12 1 )

$S25 = ( $S27 )

$MOD_LAMB = ( )
```

Resultados...

- Exemplo 2

```
-- SCRIPT

MODULE Teste

DOMAINS x:N;
                f:N

DEFINITIONS

DEF f = LET (x,2) = (1,2) IN x PLUS 1

END Teste
```

Resultados...

- Exemplo 2…

-- **LAMB**

```
LET Teste  =
  LET f  =
   LET   ( x   , 2 )  =  ( 1 , 2  )
   IN x  PLUS 1
  IN  <  >
```

-- **LAMBASIC**

```
LET Teste  =
 LET f  =
  LET nn12   =   ( 1   , 2   )
  IN
  LET x   =   ( nn12   EL  1   )
  IN 2  NE  ( nn12  EL  2  )  ->  ?  , x  PLUS 1
 IN  <  >
```

-- **SUPER**

```
$S26 = ( 1 , 2 )

$S27 = ( EL $S26 1 )

$S25 = IF NE 2 ( EL $S26 2 ) ? PLUS $S27 1

$MOD_LAMB = ( )
```

Resultados...

- Exemplo 3

```
-- SCRIPT

MODULE TesteFive

DOMAINS Fdom = N -> N;
            a,b: Fdom

DEFINITIONS

DEF a = LAM n.3

DEF b = LAM n.4

DEF n = 5

END TesteFive
```

Resultados...

- Exemplo 3...

-- **LAMB**

```
LET TesteFive  =
 LET a   =   LAM n   . 3
 ALSO b   =   LAM n   . 4
 ALSO n   = 5
 IN  <  >
```

-- **LAMBASIC**

```
LET TesteFive  =
 LET a   =   (   LAM n   . 3   )
 IN
 LET b   =   (   LAM n   . 4   )
 IN
 LET n   = 5
 IN  <  >
```

-- **SUPER**

$S26 n = 3

$S25 = ($S26)

$S28 n = 4

$S27 = ($S28)

$S29 = 5

$MOD_LAMB = ( )

Resultados...

- Exemplo 4

```
-- SCRIPT

MODULE Teste

DOMAINS Fdom = N -> N;
            x:N;
            f:Fdom


DEFINITIONS

DEF f = LAM x. x PLUS 7

END Teste
```

Resultados...

- Exemplo 4...

-- **LAMB**

```
LET Teste  =
 LET f   =   LAM x   . x   PLUS 7
 IN  <  >
```

-- **LAMBASIC**

```
LET Teste  =
 LET f   =   (   LAM x   . x   PLUS 7   )
 IN  <  >
```

-- **SUPER**

```
$S26 x = PLUS x 7

$S25 = ( $S26 )

$MOD_LAMB = ( )
```

-- **CÓDIGO C**

```
#include "includes.h"
#include "globvars.h"

void $s26 (void) {
   pushbasic(7);
   push(1);
   eval();
   get();
   plus();
   updint(2);
   pop(1);
   returng();
}
void $s25 (void) {
   pushglobal(1, $s26);
   tuple(1);
   update(1);
   unwind();
}

void $mod_lamb (void) {
   tuple(0);
   update(1);
   unwind();
```

Conclusão

- Tradução de *LAMB* para *SUPER*
  - Casamento de Padrões
  - *Lambda-Lifting*
- Otimização no código gerado
  - Ordenação de variáveis pelo nível léxico

Conclusão...

- Integração do *Front-End* e *Lambda-Lifting*

- Domínio das técnicas para implementação de linguagens funcionais
  - Casamento de Padrões
  - *Lambda-Lifting*

Trabalhos Futuros

- Eliminar definições redundantes de supercombinadores
  - η-transformações nas definições de supercombinadores
- Incorporar *Garbage Collector*

Trabalhos Futuros...

- Ambiente integrado de programação: editor de texto, *help on-line*, outros
- Integração do *Back-End*

Outros Resultados

- Exemplo 5

```
-- SCRIPT

MODULE Teste

DOMAINS a,b: N;
            Fdom = N -> N;
            y:Fdom

DEFINITIONS

DEF y = LAM a. (LET b = 3 IN b) PLUS (LET b = 2 IN b)

END Teste
```

Outros Resultados...

- Exemplo 5...

```
-- LAMB

 LET Teste  =
  LET y   =   LAM a   .   (
   LET b    = 3
   IN b   )   PLUS   (
   LET b    = 2
   IN b   )
  IN  <  >
```

Outros Resultados...

- Exemplo 5...

```
-- LAMBASIC

 LET Teste  =
  LET y   =   (   LAM a    .   (
   LET b    = 3
   IN b    )   PLUS    (
   LET b    = 2
   IN b    )   )
  IN  <  >
```

Outros Resultados...

- Exemplo 5...

```
-- SUPER

$S26 a = PLUS (
 LET b  = 3
 IN b ) (
 LET b  = 2
 IN b )

$S25 = ( $S26 )


$MOD_LAMB = ( )
```

Outros Resultados...

- Exemplo 6

```
-- SCRIPT

MODULE Teste

DOMAINS Fdom = N -> N;
                f:Fdom


DEFINITIONS

DEF f = LAM ?. 1 PLUS 7

END Teste
```

Outros Resultados...

- Exemplo 6...

```
-- LAMB

 LET Teste  =
  DEF -FUNCAO   =   LAM -FF    .   LAM -LISTA    .
   LAM -FILTRO   . -LISTA   EQ   <   >   ->   <   >   ,
   ( -LISTA   IS -XX   PRE -XXSS   ->   (
   LET -XX    PRE -XXSS    = -LISTA
   IN   ( -FILTRO -XX   )   NE   TT   ->   <   >   CAT
    ( -FUNCAO -FF -XXSS -FILTRO   )   ,  ( -FF -XX   )
    CAT   ( -FUNCAO -FF -XXSS -FILTRO   )   )   ,   ?   )
  IN
```

Outros Resultados...

- Exemplo 6...

```
-- LAMB...

DEF -MAX-INDICE  =  LAM -LP  . -LP  IS  <  >  -> 0 ,
 -LP  IS  < -II  , -LL  >  PRE -LP1  ->
 (  LAM  < -II  , -LL  >  PRE -LP1  .
LET -I-MAX  = -MAX-INDICE  ( -LP1  )
IN -II  GR -I-MAX  -> -II  , -I-MAX  )  ( -LP  )  ,  ?
IN
```

Outros Resultados...

- Exemplo 6...

```
-- LAMB...

DEF -COPY-LIST  =  LAM -LORIGEM . LAM -LPARES .
   LAM -II  .  LAM -IMAX  .
  DEF -GET-PAIRS  =  LAM -LP1 .  LAM -KK .
    -LP1  IS  <  >  -> < 0 , 0 > , -LP1  IS  < -KK1  , -VV1  >  PRE -LP2  ->
    ( LAM  < -KK1  , -VV1  >  PRE -LP2  .  -KK  EQ -KK1  -> < -KK1  , -VV1  >  ,
    -GET-PAIRS  ( -LP2  , -KK  )  )  ( -LP1  ) , ?
  IN -II  GR -IMAX  -> -LORIGEM  , -LORIGEM  IS  <  >  ->
    <  >  , -LORIGEM  IS -AA  PRE -LO1  ->
    (  LAM -AA  PRE -LO1 .
  LET -VV  =
    LET  < -KK2 , -VV2 >  = -GET-PAIRS  ( -LPARES , -II  )
    IN -KK2  EQ -II  -> -VV2  , -AA
  IN -VV  PRE -COPY-LIST ( -LO1  , -LPARES  , -II  PLUS 1 , -IMAX  )  )  ( -LORIGEM  ) ,?
 IN
```

Outros Resultados...

- Exemplo 6...

```
-- LAMB...

LET f  =  LAM  ?  . 1  PLUS 7
IN  <  >
```

Outros Resultados...

- Exemplo 6...

```
-- LAMBASIC

LET Teste  =
 DEF -FUNCAO   =   (   LAM -FF   .   (   LAM -LISTA   .   (   LAM -FILTRO   .
  -LISTA   EQ   <   >   ->   <   >   ,
  ( $EMPTY -LISTA   ->   FF   ,   TT   AND   TT   ->   (
  LET nn1    = -LISTA
  IN
  LET -XX    =    ( $HEAD nn1    )
  IN
  LET -XXSS    =    ( $TAIL nn1    )
  IN   ( -FILTRO -XX   )   NE   TT   ->   <   >   CAT
   ( -FUNCAO -FF -XXSS -FILTRO   )   ,
   ( -FF -XX   )   CAT   ( -FUNCAO -FF -XXSS -FILTRO   ) ) ,
    ?   )   )   )   )
  IN
```

Outros Resultados...

- Exemplo 6...

```
DEF -MAX-INDICE = ( LAM -LP .
  ( SIZE -LP EQ 0 ) -> TT , FF -> 0 ,
  $EMPTY -LP -> FF , SIZE $HEAD -LP NE 2 ->
  FF , TT AND TT AND TT ->
  ( ( LAM nn4 . ( LAM -II . ( LAM -LL . ( LAM -LP1 .
  LET -I-MAX = -MAX-INDICE ( -LP1 )
  IN -II GR -I-MAX -> -II , -I-MAX ) ) )
   ( ( $HEAD nn4 ) EL 1 ) ( ( $HEAD nn4 ) EL 2 )
   ( $TAIL nn4 ) ) ) ( -LP ) , ? )
 IN
```

Outros Resultados...

- Exemplo 6...

```
DEF -COPY-LIST = ( LAM -LORIGEM .
  ( LAM -LPARES . ( LAM -II . ( LAM -IMAX .
  DEF -GET-PAIRS = ( LAM -LP1 . ( LAM -KK .
  ( SIZE -LP1 EQ 0 ) -> TT , FF ->
  < 0 , 0 > , $EMPTY -LP1 -> FF ,
  SIZE $HEAD -LP1 NE 2 -> FF , TT AND TT AND
  TT -> ( ( LAM nn6 . ( LAM -KK1 . ( LAM -VV1 .
  ( LAM -LP2 . -KK EQ -KK1 -> < -KK1 , -VV1 > ,
  -GET-PAIRS ( -LP2 , -KK ) ) ) )
  ( ( $HEAD nn6 ) EL 1 ) ( ( $HEAD nn6 ) EL 2 )
  ( $TAIL nn6 ) ) ) ( -LP1 ) , ? ) )
  IN -II GR -IMAX -> -LORIGEM , ( SIZE -LORIGEM EQ 0 ) ->
  TT , FF -> < > , $EMPTY -LORIGEM -> FF ,
  TT AND TT -> ( ( LAM nn10 . ( LAM -AA . ( LAM -LO1 .
```

Outros Resultados...

- Exemplo 6...

```
LET -VV    =
   LET nn8     = -GET-PAIRS     ( -LPARES     , -II     )
   IN
   LET -KK2     =     ( nn8     EL 1     )
   IN
   LET -VV2     =     ( nn8     EL 2     )
   IN -KK2    EQ -II    -> -VV2     , -AA
  IN -VV   PRE -COPY-LIST   ( -LO1    , -LPARES    ,
   -II   PLUS 1    , -IMAX    )   )   )   ( $HEAD nn10    )
   ( $TAIL nn10    )   )   )   ( -LORIGEM    )   ,   ?    )   )   )   )
 IN
 LET f   =   (   LAM nn12    . 1    PLUS 7    )
 IN  <  >
```

Outros Resultados...

- Exemplo 6...

```
-- SUPER

$S2-LISTA -FILTRO = IF EQ -LISTA ( ) ( )
 ( IF AP $EMPTY -LISTA FF IF AND TT TT (
LET nn1  = -LISTA
IN
LET -XX  =  ( AP $HEAD nn1  )
IN
LET -XXSS  =  ( AP $TAIL nn1  )
IN IF NE ( AP -FILTRO -XX ) TT CAT ( )
 ( AP AP AP -FUNCAO -FF -XXSS -FILTRO )
 CAT ( AP -FF -XX )
 ( AP AP AP -FUNCAO -FF -XXSS -FILTRO ) ) ? )

$S3 -LISTA = ( AP $S2 -LISTA )
```

Outros Resultados...

- Exemplo 6...

```
$S4 -FF = ( $S3 )

$S1 = ( $S4 )

$S6 -MAX-INDICE -LP1 =
 LET -I-MAX  = AP -MAX-INDICE  ( -LP1  )
 IN IF GR -II -I-MAX -II -I-MAX

$S7 -MAX-INDICE -LL = ( AP $S6 -MAX-INDICE )

$S8 -MAX-INDICE -II = ( AP $S7 -MAX-INDICE )

$S9 -MAX-INDICE nn4 = AP AP AP
  ( AP $S8 -MAX-INDICE ) ( EL ( AP $HEAD nn4 ) 1 )
  ( EL ( AP $HEAD nn4 ) 2 ) ( AP $TAIL nn4 )
```

Outros Resultados...

- Exemplo 6...

```
$S10 -MAX-INDICE -LP = IF ( EQ SIZE -LP 0 ) TT
  IF FF 0 IF AP $EMPTY -LP FF IF NE
  SIZE AP $HEAD -LP 2 FF IF AND AND TT TT TT
  AP ( ( AP $S9 -MAX-INDICE ) ) ( -LP ) ?

$S5 = ( AP $S10 -MAX-INDICE )

$S12 -GET-PAIRS -KK -KK1 -VV1 -LP2 =
  IF EQ -KK -KK1 ( -KK1 , -VV1 ) AP -GET-PAIRS ( -LP2 , -KK )

$S13 -GET-PAIRS -KK -KK1 -VV1 =
  ( AP AP AP AP $S12 -GET-PAIRS -KK -KK1 -VV1 )

$S14 -GET-PAIRS -KK -KK1 =
  ( AP AP AP $S13 -GET-PAIRS -KK -KK1 )
```

Outros Resultados...

- Exemplo 6...

```
$S15 -GET-PAIRS -KK nn6 =
  AP AP AP ( AP AP $S14 -GET-PAIRS -KK )
  ( EL ( AP $HEAD nn6 ) 1 )
  ( EL ( AP $HEAD nn6 ) 2 ) ( AP $TAIL nn6 )

$S16 -GET-PAIRS -LP1 -KK =
  IF ( EQ SIZE -LP1 0 ) TT IF FF ( 0 , 0 )
  IF AP $EMPTY -LP1 FF IF NE SIZE AP $HEAD -LP1 2 FF
  IF AND AND TT TT TT
  AP ( ( AP AP $S15 -GET-PAIRS -KK ) ) ( -LP1 ) ?

$S17 -GET-PAIRS -LP1 = ( AP AP $S16 -GET-PAIRS -LP1 )
```

Outros Resultados...

- Exemplo 6...

```
$S18 -LPARES -II -GET-PAIRS -LO1 =
 LET -VV  =
  LET nn8   = AP -GET-PAIRS   ( -LPARES    , -II   )
  IN
  LET -KK2   =   (   EL nn8 1   )
  IN
  LET -VV2   =   (   EL nn8 2   )
  IN IF  EQ -KK2 -II -VV2 -AA
 IN PRE -VV AP -COPY-LIST ( -LO1 , -LPARES , PLUS -II 1 , -IMAX )

$S19 -LPARES -II -GET-PAIRS -AA =
  ( AP AP AP $S18 -LPARES -II -GET-PAIRS )

$S20 -LPARES -II -GET-PAIRS nn10 =
  AP AP ( AP AP AP $S19 -LPARES -II -GET-PAIRS )
  ( AP $HEAD nn10 ) ( AP $TAIL nn10 )
```

Outros Resultados...

- Exemplo 6...

```
$S21 -LORIGEM -LPARES -II -IMAX =
 DEF -GET-PAIRS  =  ( AP $S17 -GET-PAIRS  )
 IN IF GR -II -IMAX -LORIGEM
  IF ( EQ SIZE -LORIGEM 0 ) TT IF FF ( )
  IF AP $EMPTY -LORIGEM FF IF AND TT TT
  AP ( ( AP AP AP $S20 -LPARES -II -GET-PAIRS ) ) ( -LORIGEM ) ?

$S22 -LORIGEM -LPARES -II =
  ( AP AP AP $S21 -LORIGEM -LPARES -II )

$S23 -LORIGEM -LPARES =
  ( AP AP $S22 -LORIGEM -LPARES )

$S24 -LORIGEM = ( AP $S23 -LORIGEM )

$S11 = ( $S24 )
```

Outros Resultados...

- Exemplo 6...

$S26 nn12 = PLUS 1 7

$S25 = ($S26)

$MOD_LAMB = ( )

Supercombinador

- Definição

Um supercombinador, $S, de aridade *n* é uma abstração lambda da forma

$$\text{LAM } x_1.\text{LAM } x_2. \ldots \text{LAM } x_n.\ E$$

onde:

1- E não é uma abstração lambda;
2- $S não tem variáveis livres;
3- qualquer abstração lambda em E é um supercombinador;
4- *n* >= 0.

Dados Técnicos

- Ferramentas utilizadas:
  - *YACC* - gerador de analisador sintático;
  - *gcc* - compilador C padrão.
- Número de linhas de código
  - 1ª etapa:
  - 2ª etapa:
- Tamanho do executável
  - 1ª etapa:
  - 2ª etapa: