European Union

A modern look at GRIN
an optimizing functional language back end

Podlovics Péter - Hruska Csaba, Kaposi Ambrus

Eötvös Loránd Tudományegyetem
Budapest

Védés
2019/20/2

SZÉCHENYI 2020

European Union
European Social
Fund

HUNGARIAN
GOVERNMENT

INVESTING IN YOUR FUTURE

Tartalom

GRIN áttekintés

Graph Reduction Intermediate Notation

```
main = sum (upto 0 10)

upto n m
  | n > m = []
  | otherwise = n : upto (n+1) m

sum []       = 0
sum (x:xs) = x + sum xs
```
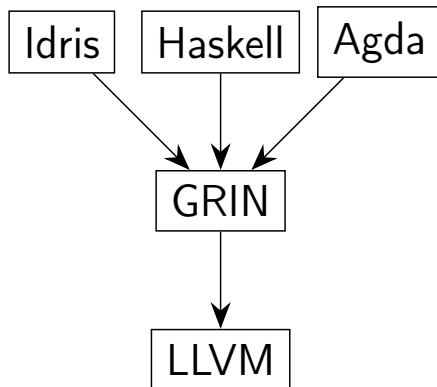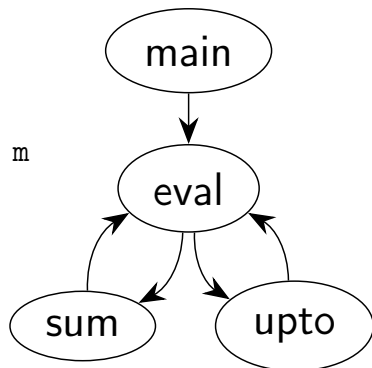
Front end kód

```
main = sum (upto 0 10)

upto n m
  | n > m = []
  | otherwise = n : upto (n+1) m

sum []     = 0
sum (x:xs) = x + sum xs
```

GRIN kód

```
                              eval p =
                                v <- fetch p
                                case v of
                                  (CInt n)     -> pure v
  grinMain =                      (CNil)       -> pure v
    t1 <- store (CInt 1)          (CCons y ys) -> pure v
    t2 <- store (CInt 10)         (Fupto a b) ->
    t3 <- store (Fupto t1 t2)       zs <- upto a b
    t4 <- store (Fsum t3)           update p zs
    (CInt r) <- eval t4             pure zs
    _prim_int_print r             (Fsum c) ->
                                    s <- sum c
                                    update p s
                                    pure s
```

Datalog áttekintés

Logikai programozás

$$c \leftarrow p_1 \land p_2 \land \cdots \land p_n$$

A GRIN nyelv Datalog modellje (részlet)

$$\frac{\texttt{p <- store n}}{\texttt{Store}(p, n)} \text{ (ER-Store)} \qquad \frac{\texttt{n <- fetch p}}{\texttt{Fetch}(n, p)} \text{ (ER-Fetch)}$$

$$\frac{\texttt{x <- update p n}}{\texttt{Update}(x, p, n)} \text{ (ER-Update)}$$

$$\frac{\texttt{k <- pure <lit>}}{\texttt{LitAssign}(k, \tau(\textit{lit}), \textit{lit})} \text{ (ER-Lit)}$$

$$\frac{\texttt{y <- pure x}}{\texttt{Move}(y, x)} \text{ (ER-Move)}$$

Points-to elemzés Datalog-ban

$$\frac{\texttt{Store}(p, n)}{\texttt{Heap}(p, n)} \ \text{(H-Store)} \qquad \frac{\texttt{Update}(*, p, n)}{\texttt{Heap}(p, n)} \ \text{(H-Update)}$$

Points-to elemzés Datalog-ban

$$\frac{\mathtt{Store}(p, n)}{\mathtt{Heap}(p, n)} \text{ (H-Store)} \qquad \frac{\begin{array}{c} \mathtt{Update}(*, p, n) \\ \mathtt{CreatedBy}(p, p') \\ \mathtt{Heap}(p', *) \end{array}}{\mathtt{Heap}(p', n)} \text{ (H-Update)}$$

Created-by elemzés (részlet)

$$\frac{\texttt{Node}(n, *)}{\texttt{CreatedBy}(n, n)} \text{ (C-Node)} \qquad \frac{\texttt{Store}(p, *)}{\texttt{CreatedBy}(p, p)} \text{ (C-Store)}$$

Created-by elemzés (részlet)

$$\frac{\texttt{Node}(n, *)}{\texttt{CreatedBy}(n, n)} \text{ (C-Node)} \qquad \frac{\texttt{Store}(p, *)}{\texttt{CreatedBy}(p, p)} \text{ (C-Store)}$$

$$\frac{\begin{array}{c}\texttt{Move}(v, n)\\ \texttt{CreatedBy}(n, n')\end{array}}{\texttt{CreatedBy}(v, n')} \text{ (C-Move)} \qquad \frac{\begin{array}{c}\texttt{Fetch}(v, p)\\ \texttt{CreatedBy}(p, p')\\ \texttt{Heap}(p', n)\\ \texttt{CreatedBy}(n, n')\end{array}}{\texttt{CreatedBy}(v, n')} \text{ (C-Fetch)}$$

Strukturális holt-kód eltávolítás

Idris példa

```
length : List a -> Int
length Nil = 0                    DDE
length (Cons x xs)            ======>
  = 1 + length xs
```

Idris példa

```
length : List a -> Int
length Nil = 0
length (Cons x xs)
  = 1 + length xs
```

$\overset{\text{DDE}}{\Longrightarrow}$

```
length : List a -> Int
length Nil = 0
length (Cons xs)
    = 1 + length xs
```

```
length : List a -> Int
length Nil = 0
length (Cons x xs)
  = 1 + length xs
```

A generált GRIN kód

```
length : List a -> Int
length Nil = 0
length (Cons x xs)
  = 1 + length xs
```

```
length p =
  xs <- fetch p
  case xs of
    (Cons y ys) ->
      l1 <- length ys
      l2 <- int_add l1 1
      pure l2
    (Nil) ->
      pure 0
```

A generált GRIN kód

```
length : List a -> Int
length Nil = 0
length (Cons x xs)
  = 1 + length xs
```

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) @ alt2 ->
      k0 <- pure 0
      pure k0
  pure r
```

A GRIN program Datalog reprezentációja

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) @ alt2 ->
      k0 <- pure 0
      pure k0
  pure r
```

A GRIN program Datalog reprezentációja

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) @ alt2 ->
      k0 <- pure 0
      pure k0
  pure r
```

```
FunParam(length,0,p)
Fetch(xs,p)
Case(r,xs)
Alt(r,alt1,CCons)
AltParam(r,CCons,0,y)
AltParam(r,CCons,1,ys)
Call(l1,length)
CallArgument(l1,0,ys)
LitAssign(k1,Int,1)
Call(l2,int_add)
CallArgument(l2,0,l1)
CallArgument(l2,1,k1)
ReturnValue(alt1,l2)

...
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) @ alt2 ->
      k0 <- pure 0
      pure k0
  pure r
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
        l1 <- length ys
        k1 <- pure 1
        l2 <- int_add l1 k1
        pure l2
      (Nil) @ alt2 ->
        k0 <- pure 0
        pure k0
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0 <- pure 0
      pure k0*
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1 <- length ys
      k1 <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1* <- length ys
      k1* <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p =
  xs* <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1* <- length ys
      k1* <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p* =
  xs* <- fetch p
  r <- case xs of
    (Cons y ys) @ alt1 ->
      l1* <- length ys
      k1* <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

# Az élőségi elemzés eredménye (részlet)

```
length p* =
  xs* <- fetch p
  r <- case xs of
    (Cons y ys*) @ alt1 ->
      l1* <- length ys
      k1* <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

Az élőségi elemzés eredménye (részlet)

```
length p* =
  xs* <- fetch p
  r <- case xs of
    (Cons y ys*) @ alt1 ->
      l1* <- length ys
      k1* <- pure 1
      l2* <- int_add l1 k1
      pure l2*
    (Nil) @ alt2 ->
      k0* <- pure 0
      pure k0*
  pure r*
```

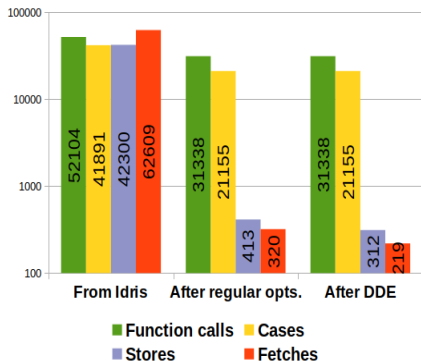| Var	Liveness
p	$\top$
y	$\bot$
xs	$Nil[], Cons[\bot, \top]$
ys	$\top$
l1	$\top$
k1	$\top$
l2	$\top$
k0	$\top$
r	$\top$(*feltetelezes*)

Mérési eredmények

Környezet

- Kis Idris programok:
  Type-driven Development with Idris - Edwin Brady

- Interpretált GRIN programok, és futtatott gépi kód is
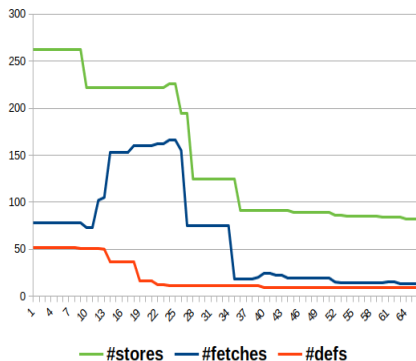
- Fordítási- és futásidejű mérések
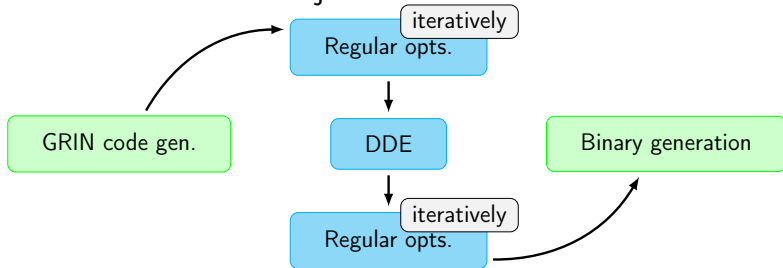
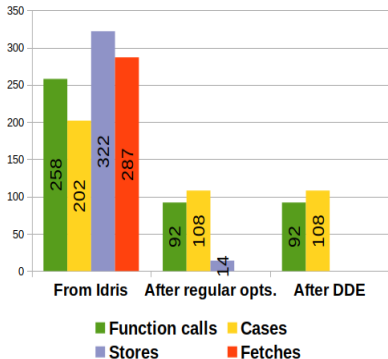# Length - GRIN statisztikák

Length - CPU bináris statisztikák

Stage	Size	Inst.	Stores	Loads	Mem.
idris	-	2822725	366880	1064977	9440
normal-O0	23928	769588	212567	233305	674080
normal-O3	23928	550065	160252	170202	674080
regular-opt	19832	257397	14848	45499	8200
dde-O0	15736	256062	14243	45083	5776
dde-O3	15736	284970	33929	54555	5776

Exact length - GRIN statisztikák
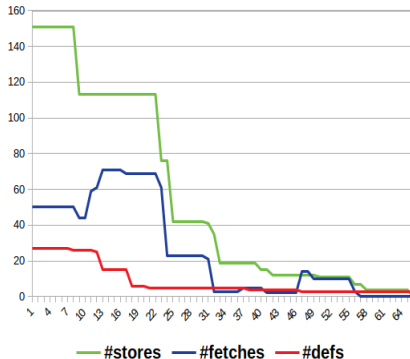
Exact length - CPU bináris statisztikák

Stage	Size	Inst.	Stores	Loads	Mem.
`idris`	-	260393	23320	68334	1888
`normal-O0`	18800	188469	14852	46566	4112
`normal-O3`	14704	187380	14621	46233	4112
`regular-opt`	10608	183560	13462	45214	112
`dde-O0`	10608	183413	13431	45189	0
`dde-O3`	10608	183322	13430	44226	0

Összefoglaló

- Újítasok:
  - új szintaxis
  - Datalog modell, Datalog elemzések
  - strukturális holt-kód eltávolítás

- Eredmények:
  - a strukturális holt-kód eltávolítás képes jelentősen csökkenteni a bináris méretét
  - a rendszer jól működik függőtípusos nyelvekre is
  - az optimalizált GRIN kód jelentősen hatékonyabb
  - a GRIN optimalizációk ortogonálisak az LLVM optimalizációkra

EFOP-3.6.2-16-2017-00013

European Union

KÖSZÖNÖM
A FIGYELMET!