

EFOP-3.6.2-16-2017-00013



European Union

# A modern look at GRIN

an optimizing functional language back end

Podlovics Péter - Hruska Csaba, Kaposi Ambrus

Eötvös Loránd Tudományegyetem  
Budapest

TDK-2020



HUNGARIAN  
GOVERNMENT

European Union  
European Social  
Fund



INVESTING IN YOUR FUTURE

# Tartalom

GRIN áttekintés

Strukturális holt-kód eltávolítás szemléltetése

Új szintaxis és Datalog modell

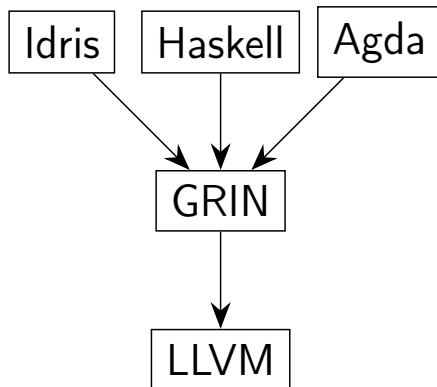
Created-by és élőségi elemzések

Strukturális holt-kód eltávolítás

Mérési eredmények

# GRIN áttekintés

# Graph Reduction Intermediate Notation



# A generált GRIN kód

```
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
```

# Strukturális holt-kód eltávolítás szemléltetése

# Idris példa a transzformációra

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



# Idris példa a transzformációra

<code>length : List a -&gt; Int</code>		<code>length : List a -&gt; Int</code>
<code>length Nil = 0</code>		<code>length Nil = 0</code>
<code>length (Cons x xs)</code>	$\xRightarrow{\text{DDE}}$	<code>length (Cons xs)</code>
<code>  = 1 + length xs</code>		<code>  = 1 + length xs</code>

# Új szintaxis és Datalog modell

# A generált GRIN kód (új szintaxis)

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

# A generált GRIN kód (új szintaxis)

```
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
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) ->
      k0 <- pure 0
      pure k0
```

# A generált GRIN kód (új szintaxis)

```
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) ->
      l1 <- length ys
      k1 <- pure 1
      l2 <- int_add l1 k1
      pure l2
    (Nil) ->
      k0 <- pure 0
      pure k0
  pure r
```

# A generált GRIN kód (új szintaxis)

```
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 nyelv Datalog modellje (részlet)

$$\frac{p \leftarrow \text{store } n}{\text{Store}(p, n)} \text{ (ER-Store)}$$

$$\frac{n \leftarrow \text{fetch } p}{\text{Fetch}(n, p)} \text{ (ER-Fetch)}$$

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

$$\frac{k \leftarrow \text{pure } \langle \text{lit} \rangle}{\text{LitAssign}(k, \tau(\text{lit}), \text{lit})} \text{ (ER-Lit)}$$

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

# 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)
```

...

# Created-by és előségi elemzések

$$c \vee \neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n$$

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

# Points-to elemzés Datalog-ban

$$\frac{\text{Store}(p, n)}{\text{Heap}(p, n)} \text{ (H-Store)}$$

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

# Points-to elemzés Datalog-ban

$$\frac{\text{Store}(p, n)}{\text{Heap}(p, n)} \text{ (H-Store)}$$

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

```
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
```

# Created-by elemzés eredménye

```
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	Producers
p	
y	
xs	
ys	
l1	
k1	
l2	
k0	
r	



# Created-by elemzés eredménye

```
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	Producers
p	...
y	...
xs	...
ys	...
l1	
k1	
l2	
k0	
r	

# Created-by elemzés eredménye

```
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	Producers
p	...
y	...
xs	...
ys	...
l1	
k1	{k1}
l2	
k0	{k0}
r	

# Created-by elemzés eredménye

```
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	Producers
p	...
y	...
xs	...
ys	...
l1	
k1	{k1}
l2	{l2}
k0	{k0}
r	

# Created-by elemzés eredménye

```
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	Producers
p	...
y	...
xs	...
ys	...
l1	
k1	{k1}
l2	{l2}
k0	{k0}
r	{l2, k0}

# Created-by elemzés eredménye

```
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	Producers
p	...
y	...
xs	...
ys	...
l1	{l2, k0}
k1	{k1}
l2	{l2}
k0	{k0}
r	{l2, k0}

# Az élősségi 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 élősségi 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 élősségi 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 élősségi 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 élősségi 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 élősségi 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 élősségi 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 élősségi 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 élősségi 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	$\perp$
xs	$Nil[], Cons[\perp, \top]$
ys	$\top$
l1	$\top$
k1	$\top$
l2	$\top$
k0	$\top$
r	$\top$ (feltetelezés)

# Strukturális holt-kód eltávolítás

# Idris példa a transzformációra

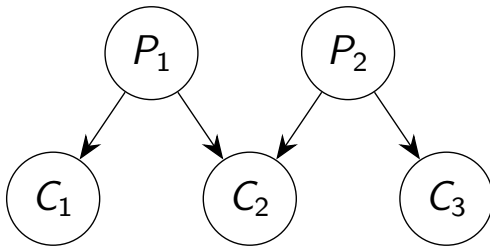
```
length : List a -> Int
length Nil = 0
length (Cons x xs)
  = 1 + length xs
```



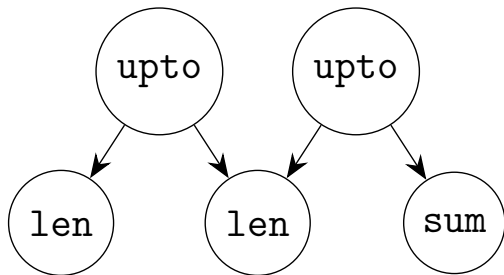
# Idris példa a transzformációra

<code>length : List a -&gt; Int</code>		<code>length : List a -&gt; Int</code>
<code>length Nil = 0</code>		<code>length Nil = 0</code>
<code>length (Cons x xs)</code>	$\xRightarrow{\text{DDE}}$	<code>length (Cons xs)</code>
<code>  = 1 + length xs</code>		<code>  = 1 + length xs</code>

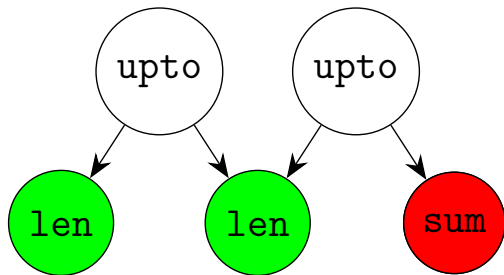
# Termelők és fogyasztók



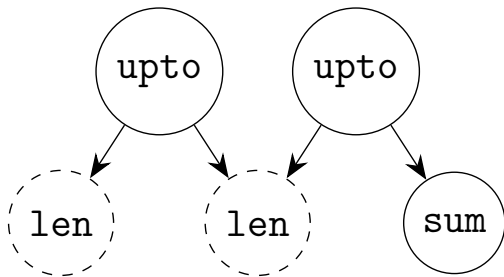
# Termelők és fogyasztók



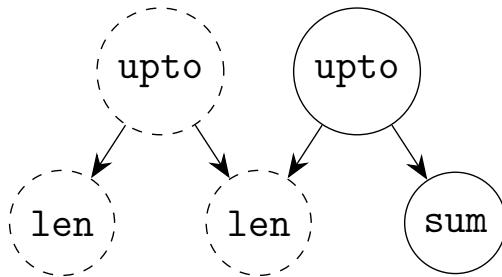
# Termelők és fogyasztók



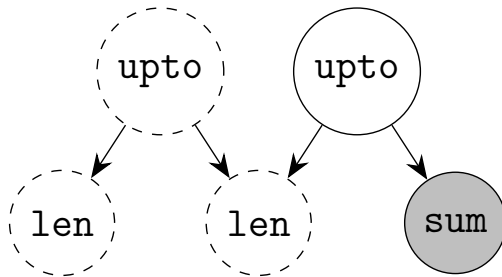
# Termelők és fogyasztók



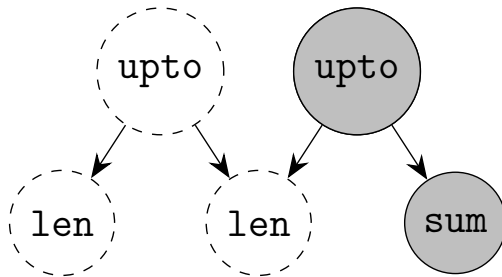
# Termelők és fogyasztók



# Termelők és fogyasztók

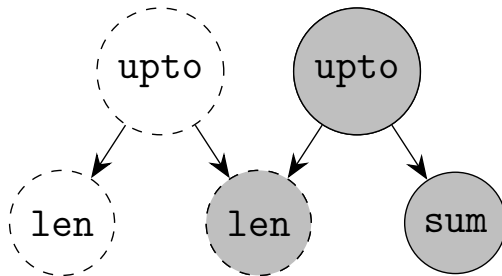


# Termelők és fogyasztók

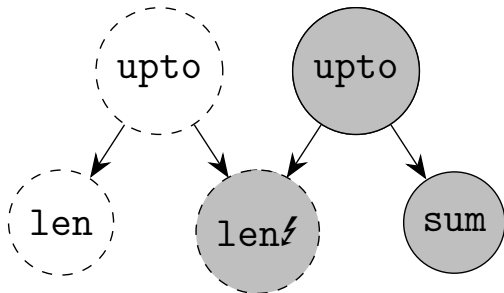




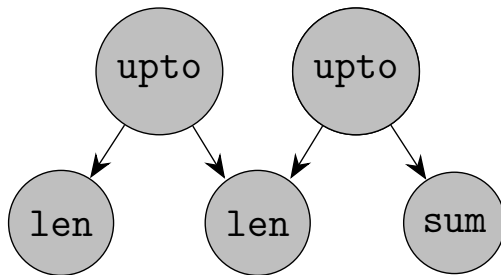
# Termelők és fogyasztók



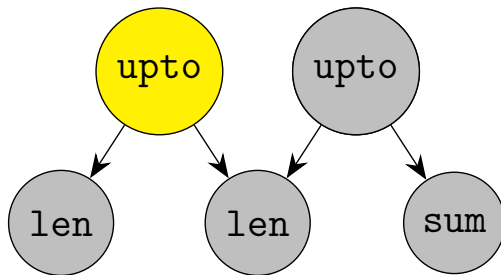
# Termelők és fogyasztók



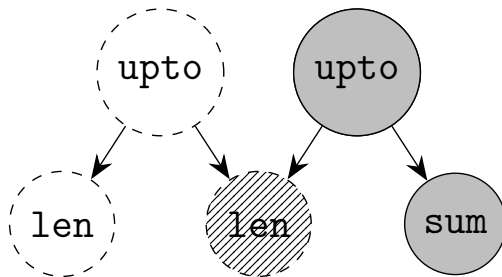
# Termelők és fogyasztók



# Termelők és fogyasztók

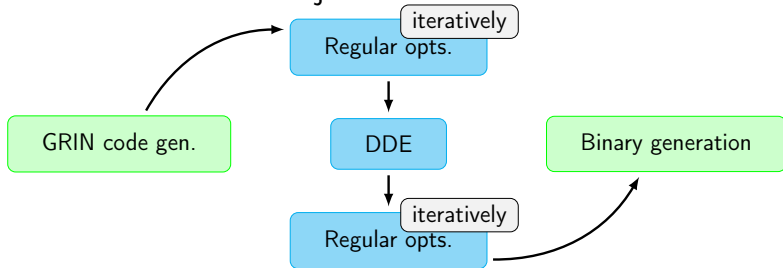


# Termelők és fogyasztók



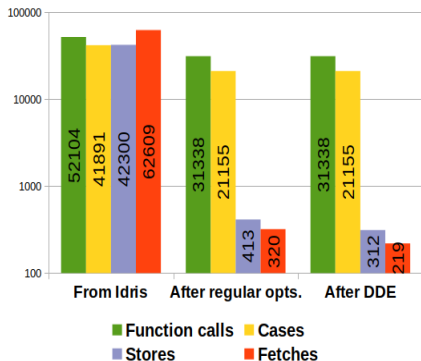
# Mérési eredmények

- Kis Idris programok:  
*Type-driven Development with Idris* - Edwin Brady
- Interpretált GRIN programok, és futtatott gépi kód is
- Fordítási- és futásidejű mérések

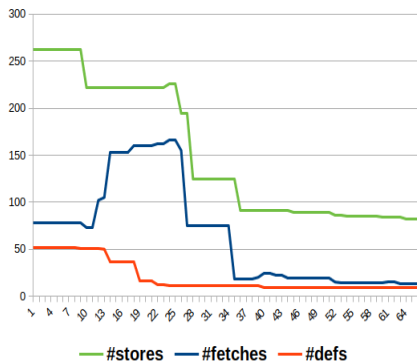


# Length - GRIN statisztikák

Runtime Statistics



Compile Time Statistics



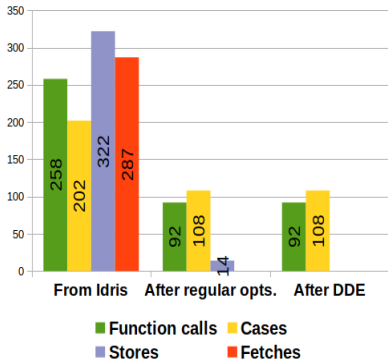


# Length - CPU bináris statisztikák

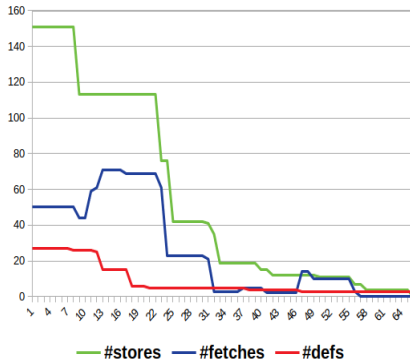
Stage	Size	Inst.	Stores	Loads	Mem.
idris	-	2822725	366880	1064977	9440
normal-00	23928	769588	212567	233305	674080
normal-03	23928	550065	160252	170202	674080
regular-opt	19832	257397	14848	45499	8200
dde-00	15736	256062	14243	45083	5776
dde-03	15736	284970	33929	54555	5776

# Exact length - GRIN statisztikák

Runtime Statistics



Compile Time Statistics



# Exact length - CPU bináris statisztikák

Stage	Size	Inst.	Stores	Loads	Mem.
idris	-	260393	23320	68334	1888
normal-00	18800	188469	14852	46566	4112
normal-03	14704	187380	14621	46233	4112
regular-opt	10608	183560	13462	45214	112
dde-00	10608	183413	13431	45189	0
dde-03	10608	183322	13430	44226	0

# Összefoglaló

- Újítások:
  - 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

# Publikációk, előadások

- Publikációk:
  - ActaCybernetica
- Előadások:
  - EUTypes 2019
  - Midlands Graduate School 2019
  - London Haskell Meetup 2019

EFOP-3.6.2-16-2017-00013



European Union

# KÖSZÖNÖM A FIGYELMET!

**SZÉCHENYI** 2020



HUNGARIAN  
GOVERNMENT

European Union  
European Social  
Fund



INVESTING IN YOUR FUTURE