

Genetické programování v typovaných jazycích

Tomáš Křen

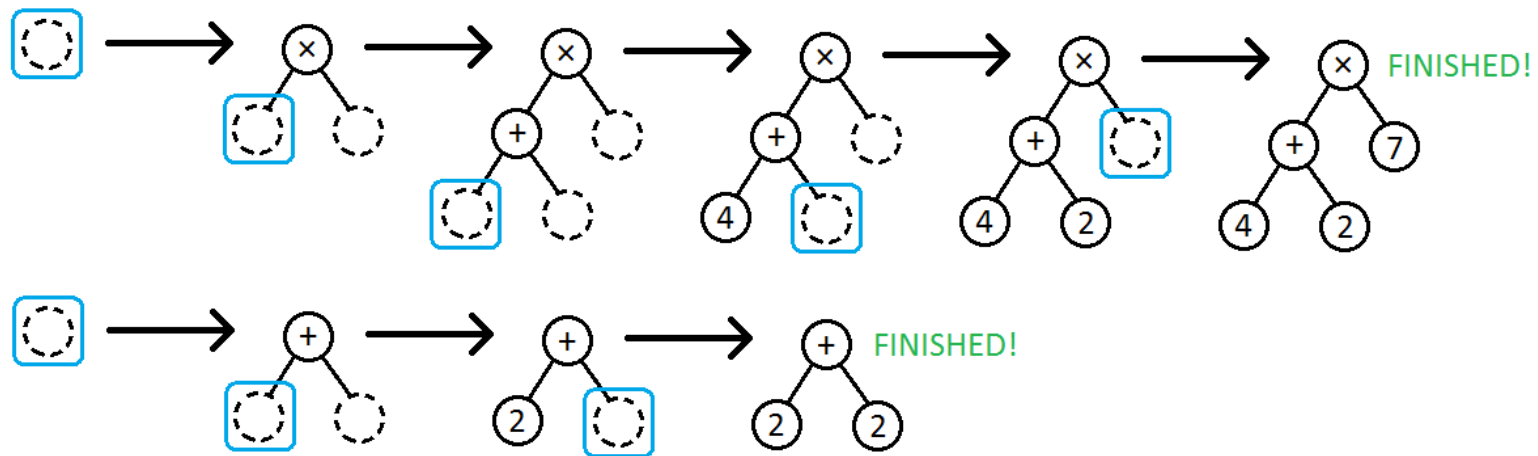
Vedoucí: Roman Neruda

Články

- Generating Lambda Term Individuals in Typed Genetic Programming Using Forgetful A*
 - Konference IEEE WCCI 2014, Peking
- Utilization of Reductions and Abstraction Elimination in Typed Genetic Programming
 - Konference GECCO 2014, Vancouver

Standardní generování jedinců

- Po jednom:



- Alternativní postup: Víc najednou.
 - “Generování sdílených částí může být sdíleno.”

Systematické generování jedinců

INPUT

Ⓐ

Ⓑ

Ⓒ

e.g. $T = \{a\}$, $F = \{b:1 \text{ arg}, c:2 \text{ args}\}$

PRIORITY QUEUE

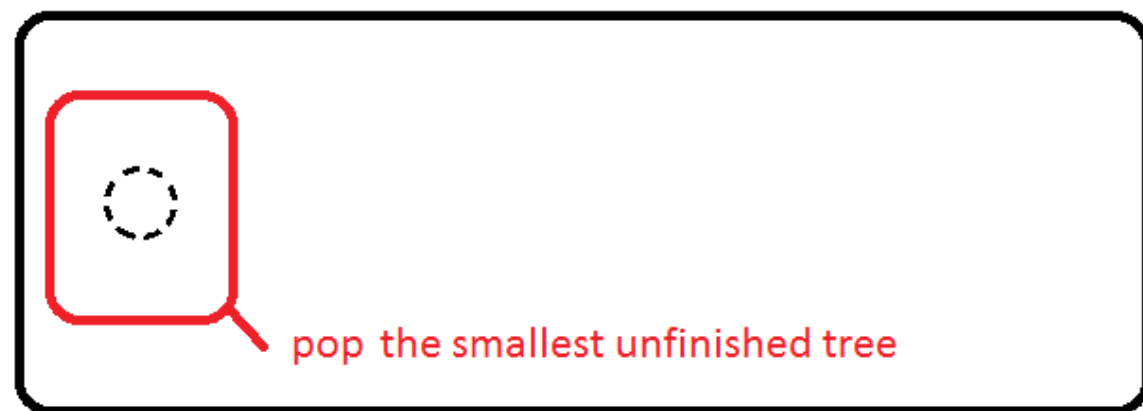


OUTPUT

INPUT



PRIORITY QUEUE

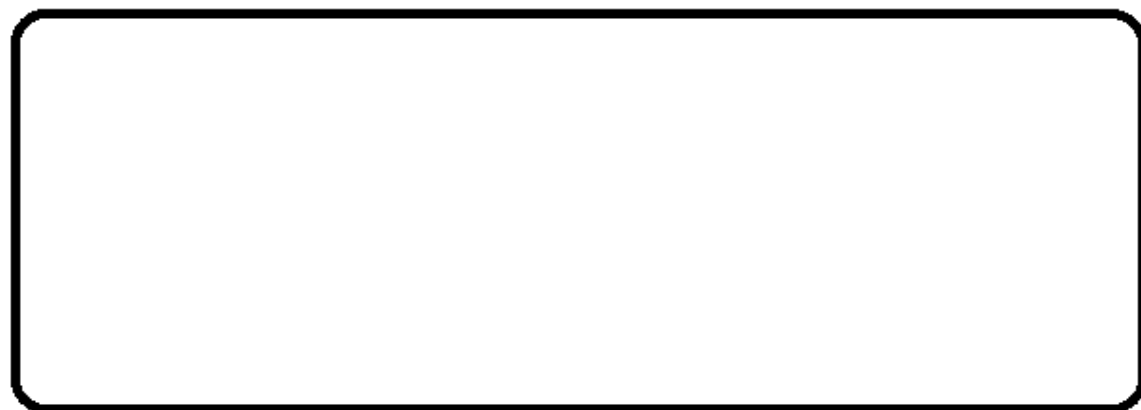


OUTPUT

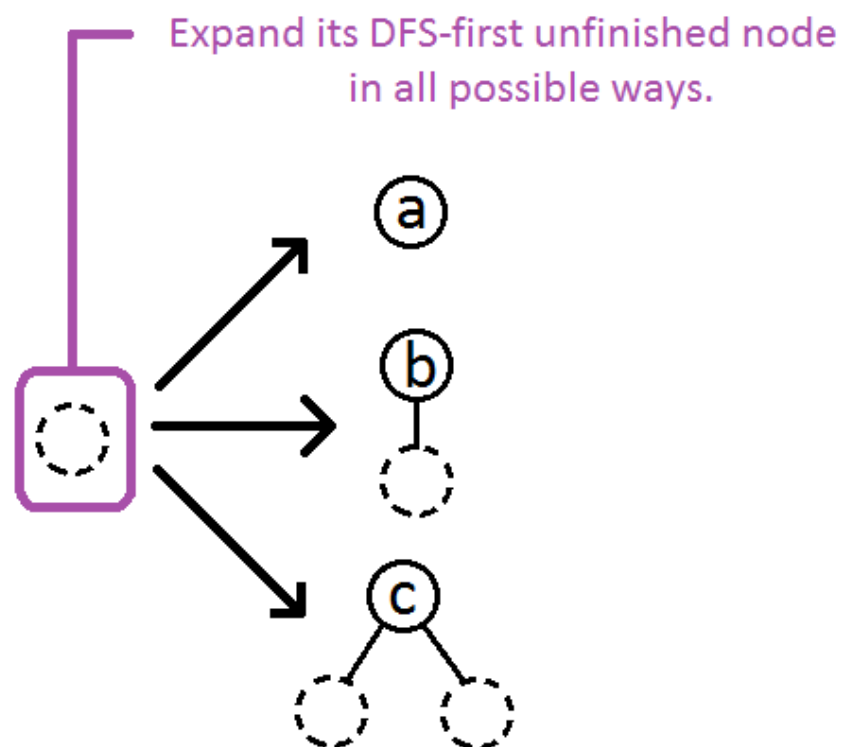
INPUT



PRIORITY QUEUE



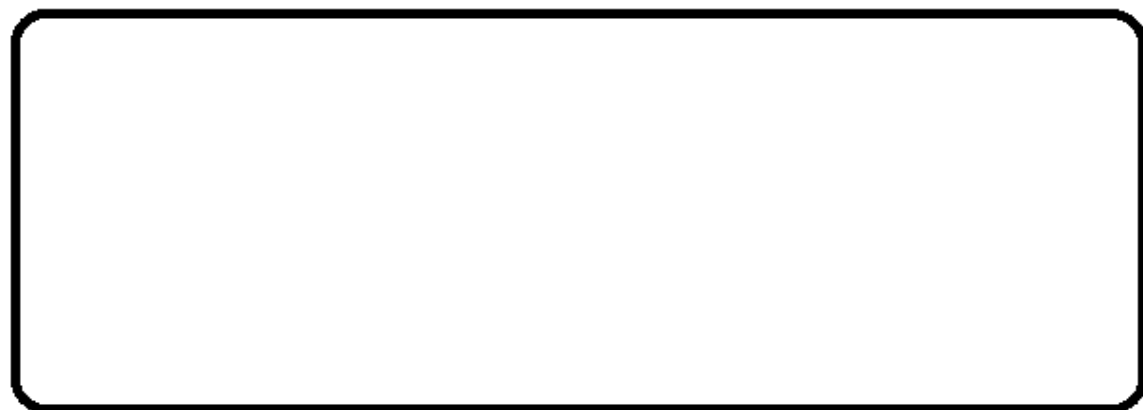
OUTPUT



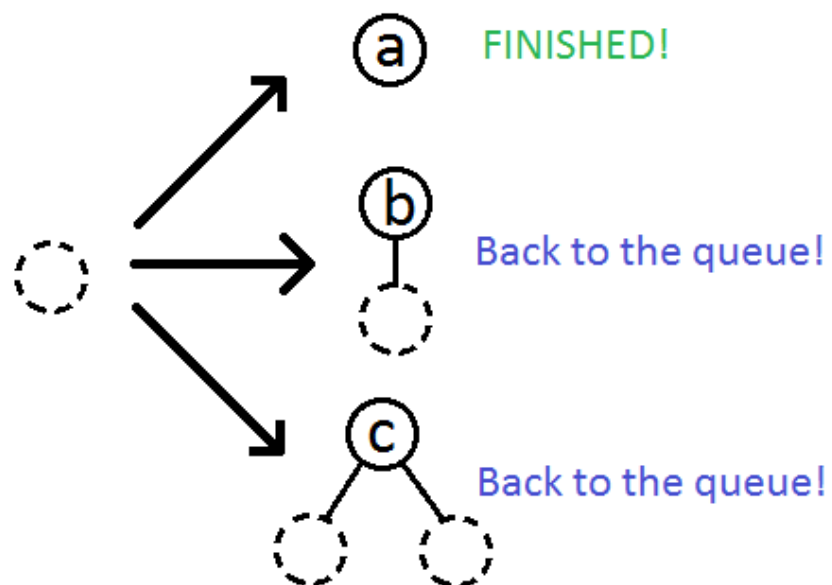
INPUT



PRIORITY QUEUE



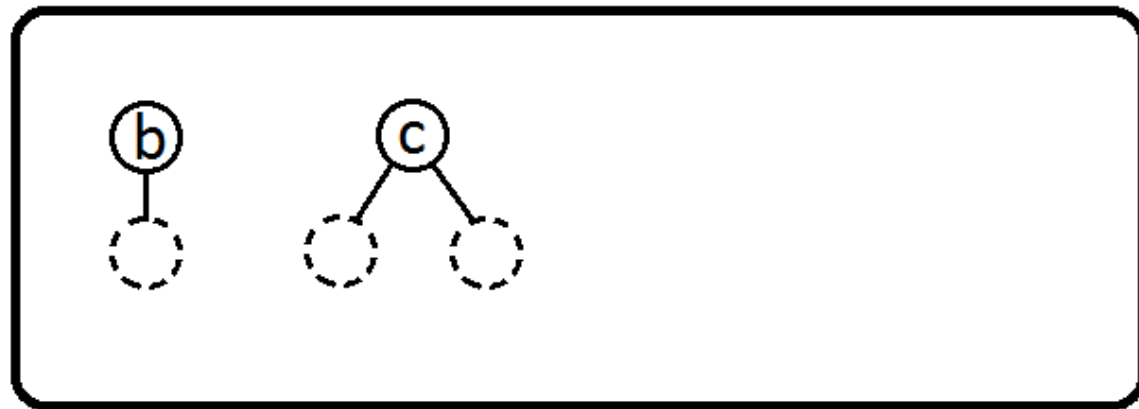
OUTPUT



INPUT



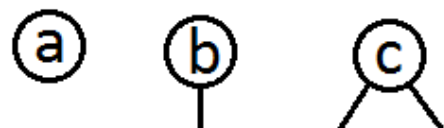
PRIORITY QUEUE



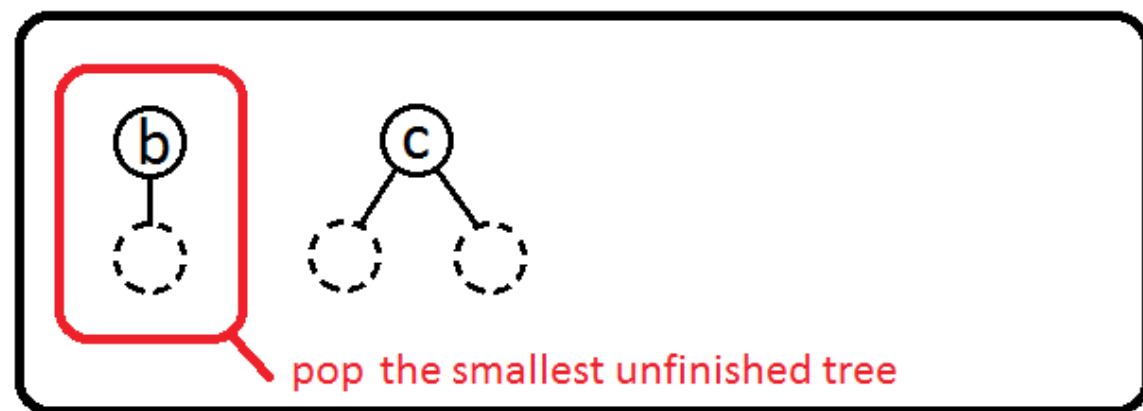
OUTPUT



INPUT



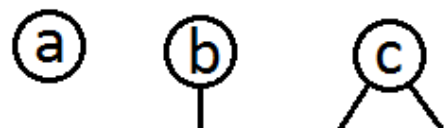
PRIORITY QUEUE



OUTPUT



INPUT



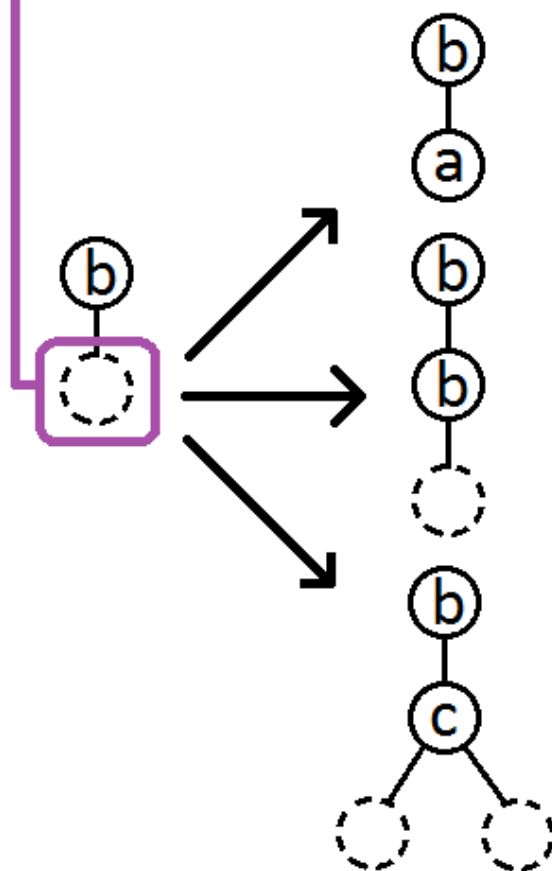
PRIORITY QUEUE



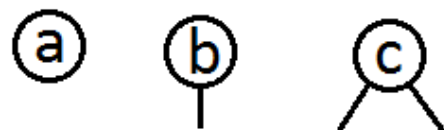
OUTPUT



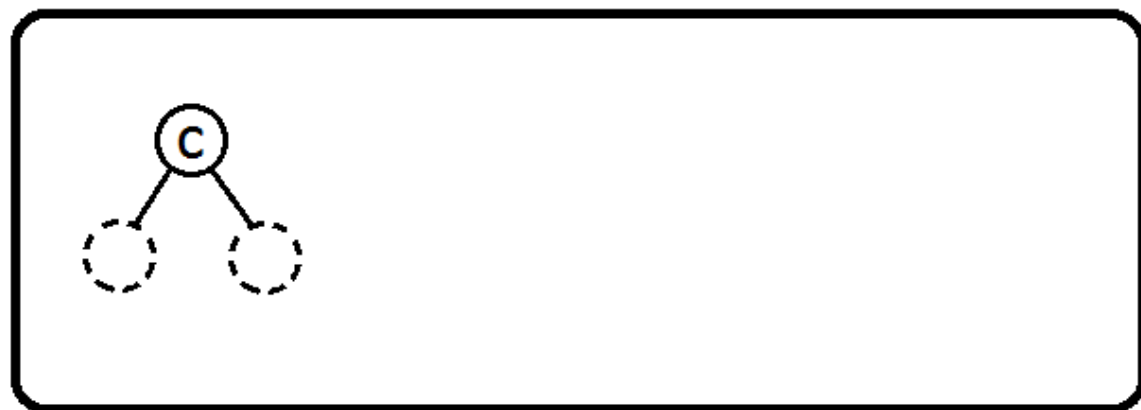
Expand its DFS-first unfinished node in all possible ways.



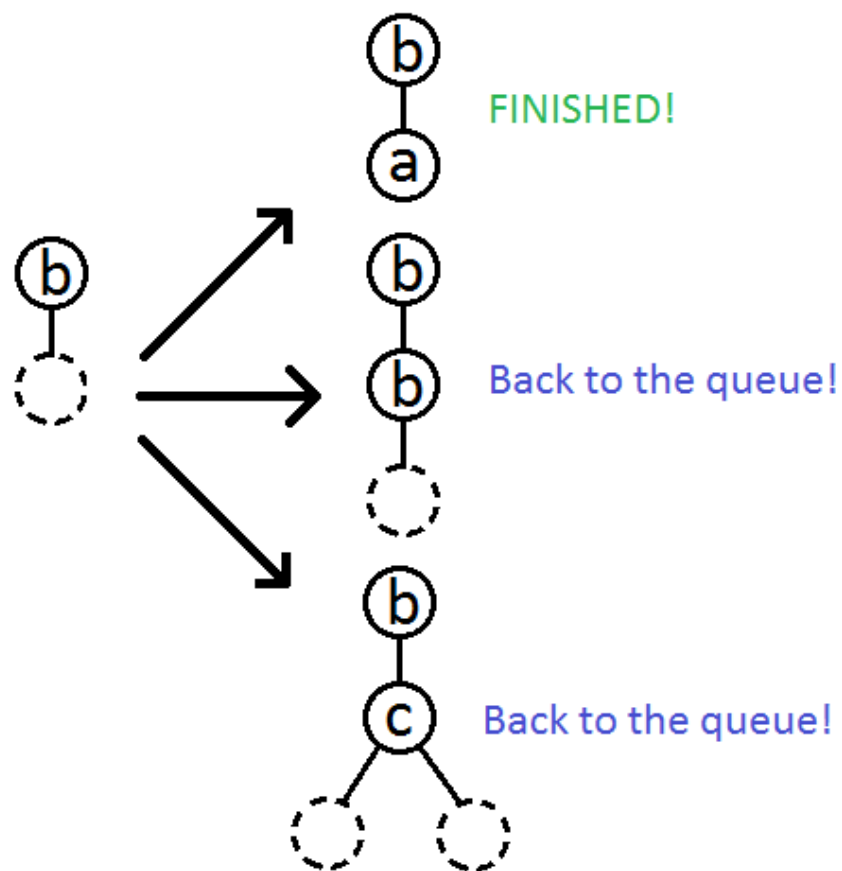
INPUT



PRIORITY QUEUE



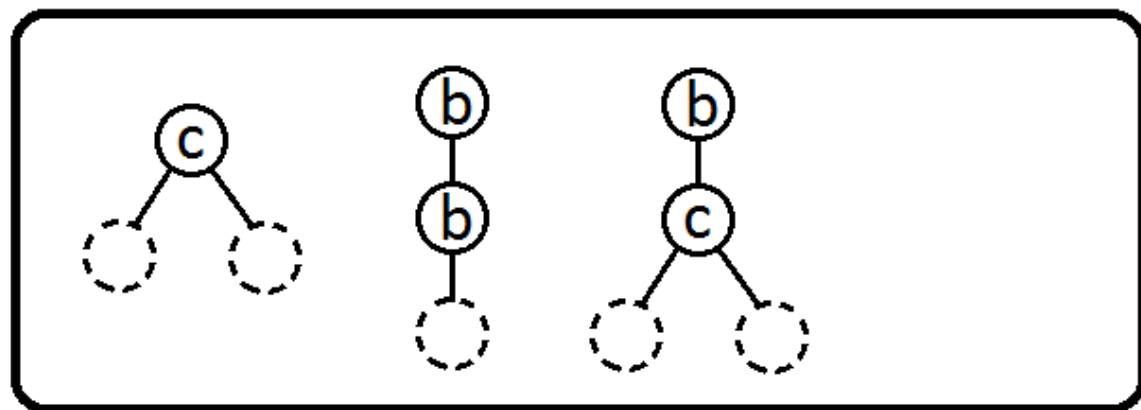
OUTPUT



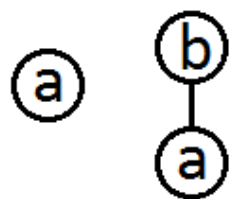
INPUT



PRIORITY QUEUE

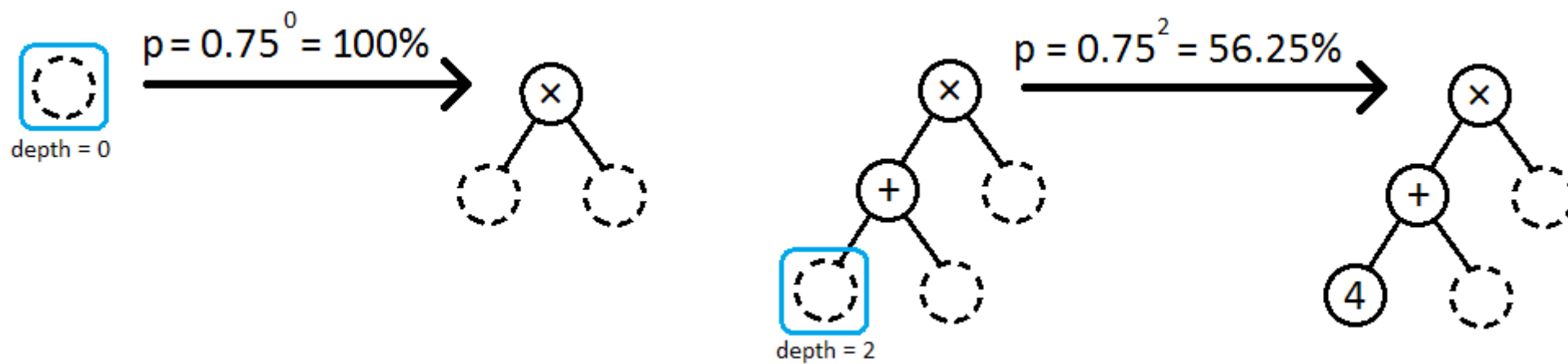


OUTPUT


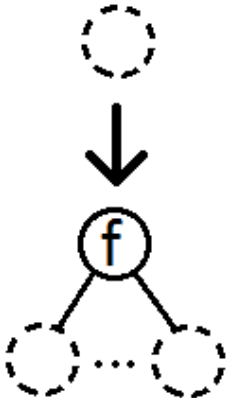
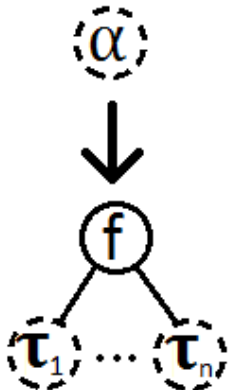
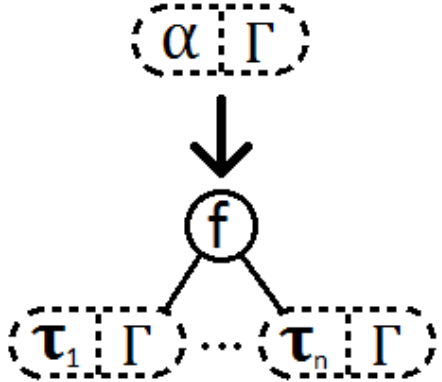
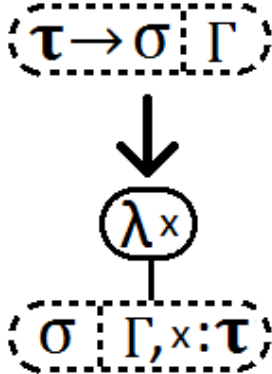


Naše *geometrická* strategie

- Dá expandovaný strom zpět do fronty s pstí $p = q^{\text{hloubka}}$
- Používáme $q = 0.75$
- A **hloubka** je hloubka expandovaného vrcholu.

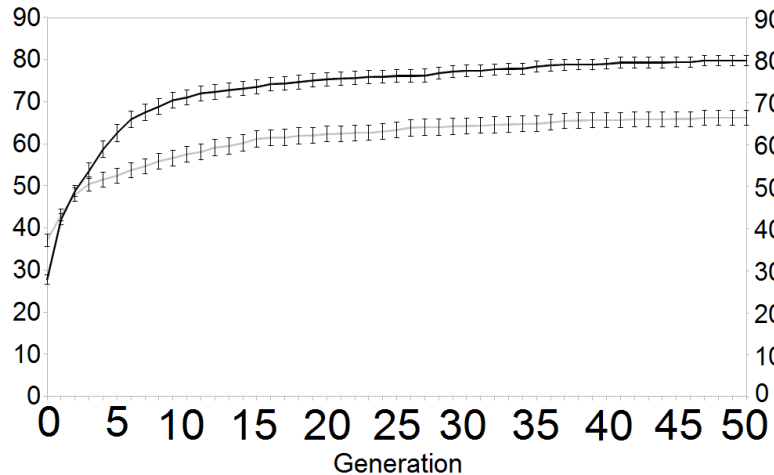


Zobecnění pro simply typed lambda calculus

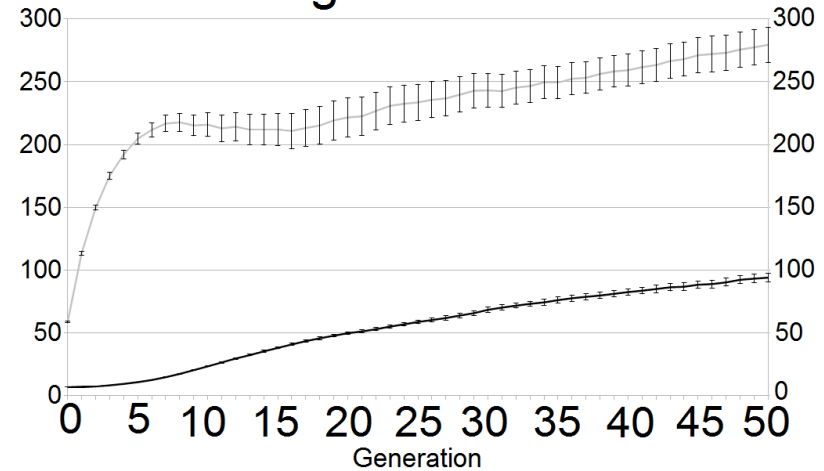
	No types	Types, no contexts	Simply typed lambda calculus	
Unfinished node		(α)	$(\tau \mid \Gamma)$	
Expansion(s)		 $f : \underbrace{\tau_1 \rightarrow \dots \rightarrow \tau_n}_{\text{inputs types}} \rightarrow \underbrace{\alpha}_{\text{ouput type}}$	<i>atomic types:</i>  $(f : \tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \alpha) \in \Gamma$	<i>function types:</i> 

Artificial ant problem

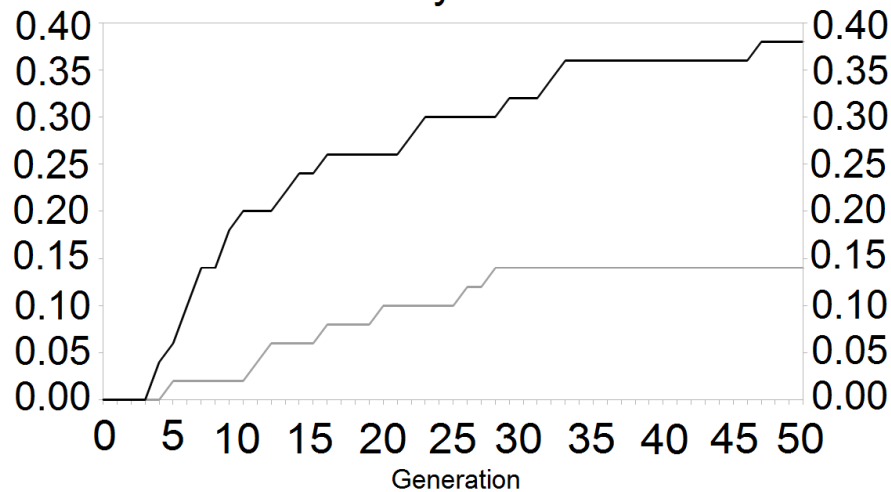
Fitness of the best individual



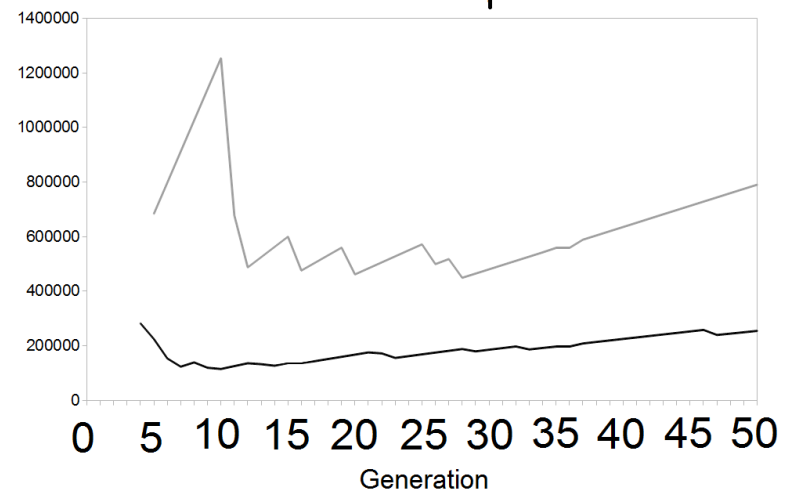
Average term size



Probability of success



Individuals to be processed

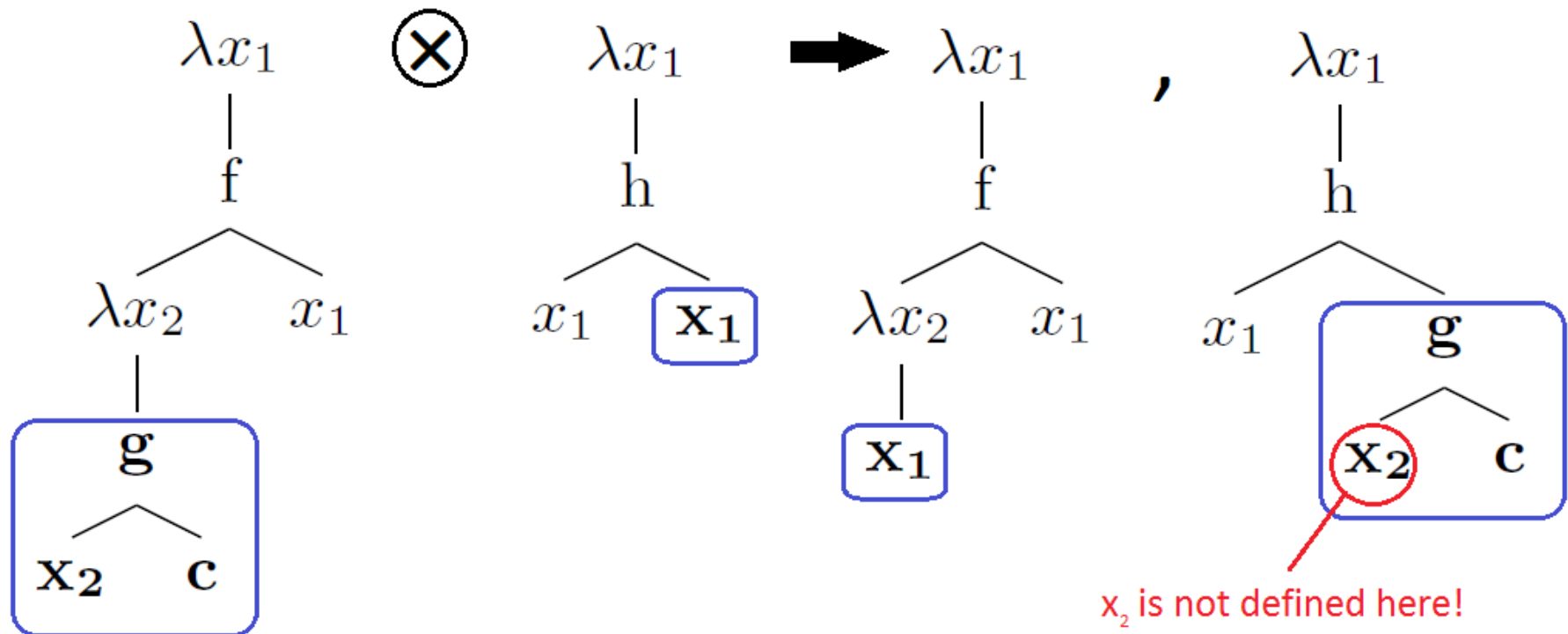


— Ramped half-and-half
— Geometric

Times: 265 minutes
107 minutes

Křížení pro typované lambda termy

- Je třeba prohazovat podstromy stejného typu
 - ..to je jednoduché
- Ale lokální proměnné zlobí!



Abstraction elimination

- Algoritmus pro zbavení se lokálních proměnných a lambda abstrakcí

$S = \lambda f g x . f x (g x)$

"function(f,g,x){ return f(x, g(x)) }"

$K = \lambda x y . x$

***i.e.* "function(x,y){ return x }"**

$I = \lambda x . x$

"function(x){ return x }"

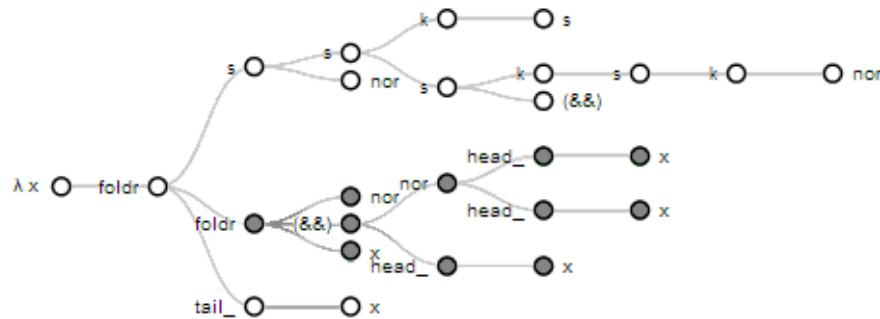
Hybrid crossover

- Každý vygenerovaný jedinec je transformován eliminačním algoritmem „po narození“

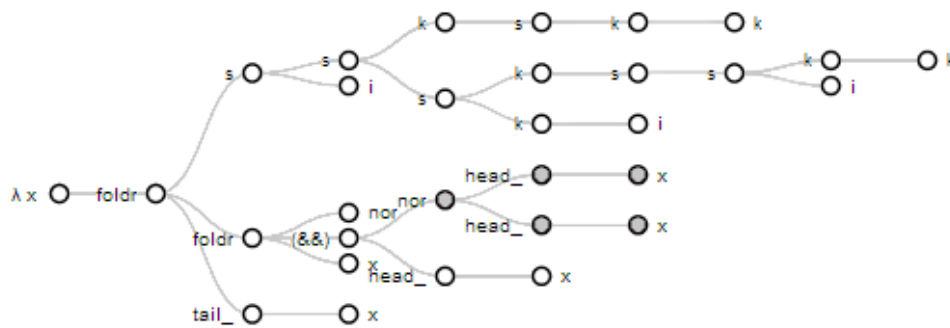
Unpacking crossover

- Jedinci drženy v $\beta\eta$ -normalní formě (zabalení).
- .. a transformujeme je (rozbalíme) těsně před křížením
- Po křížení zase normalizujeme (zabalíme).

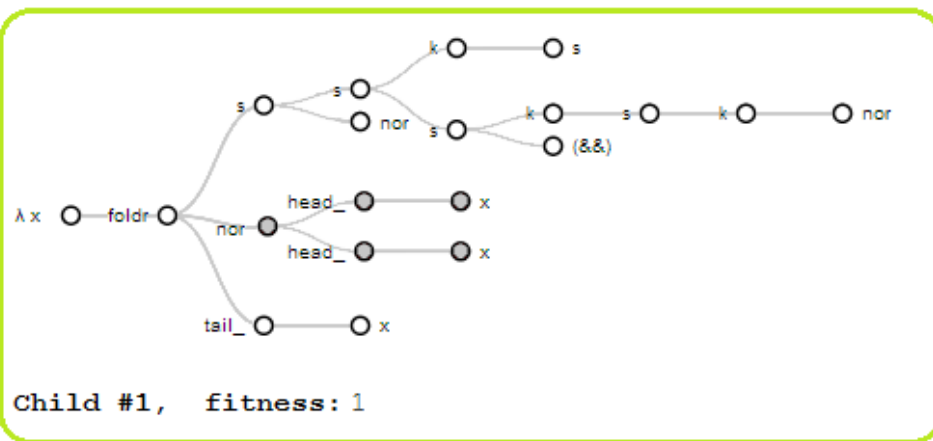
Even parity problem



Parent #1, fitness: 0.8125



Parent #2, fitness: 0.5



Child #1, fitness: 1

$$\lambda x . foldr (S(S(K S)(S(K(S(K nor))))and))nor) \\ (nor (head' x) (head' x)) (tail' x)$$

$=_{\beta\eta}$

$$\lambda x . foldr (\lambda y z . nor (and y z) (nor y z)) \\ (nor (head' x) (head' x)) (tail' x)$$

Which is equivalent to:

$$\lambda x . foldr xor (not (head' x)) (tail' x)$$

GP approach	$I(M, i, z)$
PolyGP	14,000
Our approach (hybrid)	28,000
GP with Combinators	58,616
GP with Iteration	60,000
Our approach (unpacking)	114,000
Generic GP	220,000
OOGP	680,000
GP with ADFs	1,440,000

Results comparison

Další (rozpracovaná) práce

- Další zobecnění typového systému
 - Hindley–Milnerův typový systém
 - Hindley–Milner navíc s typovými třídami
- Návrh zajímavého problému pro tento systém
 - Problémy okolo simulace jednoduché ekonomiky z multiagentního pohledu