

Generating Lambda Term Individuals in Typed Genetic Programming Using Forgetful A*

Tomáš Křen & Roman Neruda

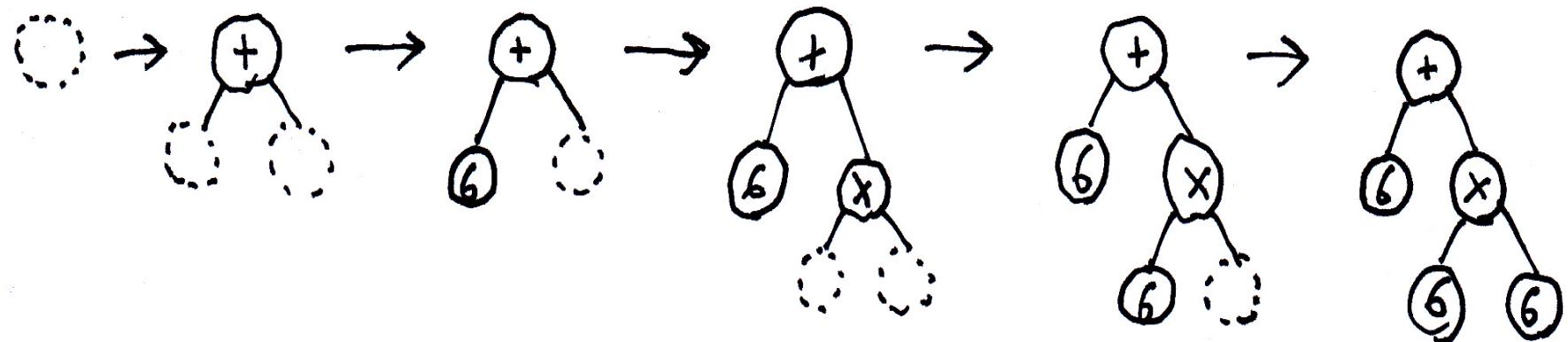
Úvodní propaganda

- Vzít z článku pár motivačních vět...

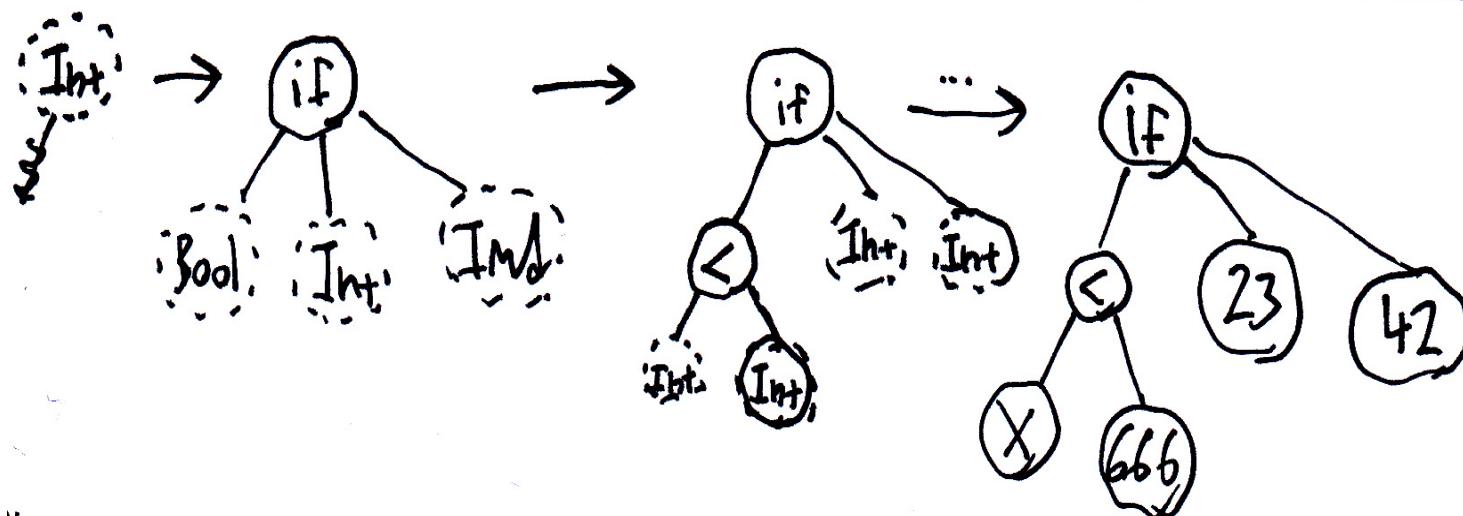
Standardní generování stromů v GP

- Obrázek jak se to generuje do hloubky
- A obrázek jak se to dělá, když je to typovaný (bez higher-orderismů, tzn argument nebude funkčního typu)
 - do „unfinished bubble“ se přidá info o typu, kterej pak musí volba stavebního symbolu akceptovat

Std. generatörüm



Typovare (jetzschte) generatörüm ($\text{if} : \text{Bool} \rightarrow \text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$)



Sériově vs. paralelně

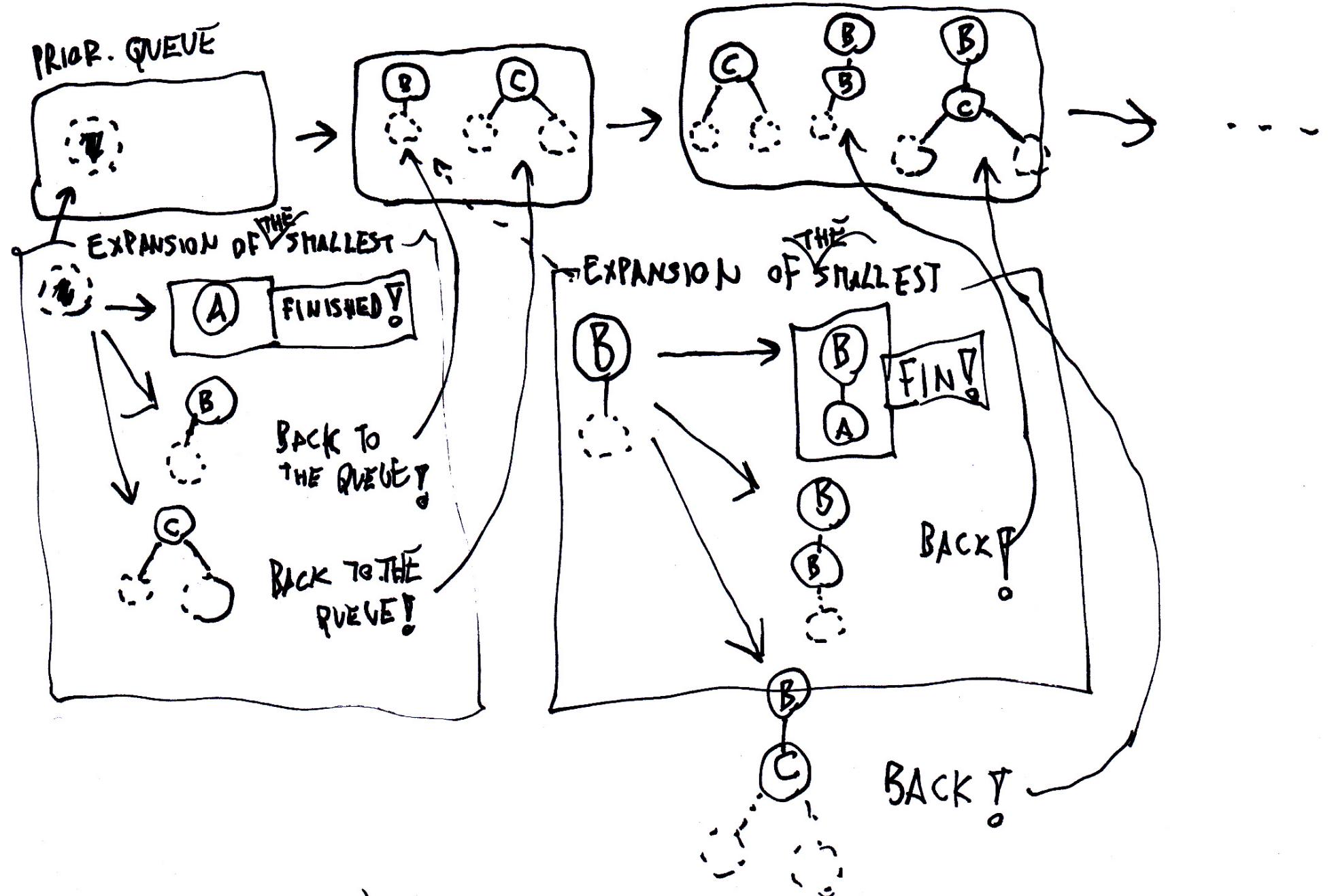
- Std metoda dělá stromy sériově, má 1 rozpracovaný ten dodělá a začne další
 - To dělá i ramped half-and-half
- Stromy můžeme dělat i paralelně – tzn mít jich rozpracovaných více



A* jako extrém paralelizmu

- Na jednou zpracováváme mnoho stromů
- Tuto kolekci držíme v prioritní frontě
 - priorita podle optimistického odhadu konečné velikosti stromu
- V jednom kroku vezmeme nejmenší strom, expandujeme ho a následníky vložíme zpět do fronty
 - Ilustrovat obrázkem

Illustrace 1st na střomech

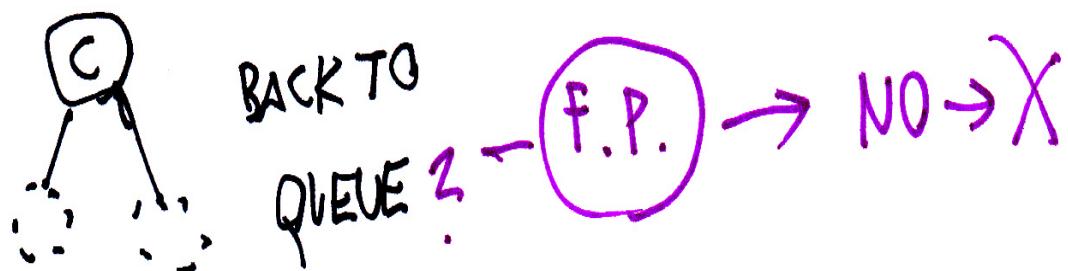
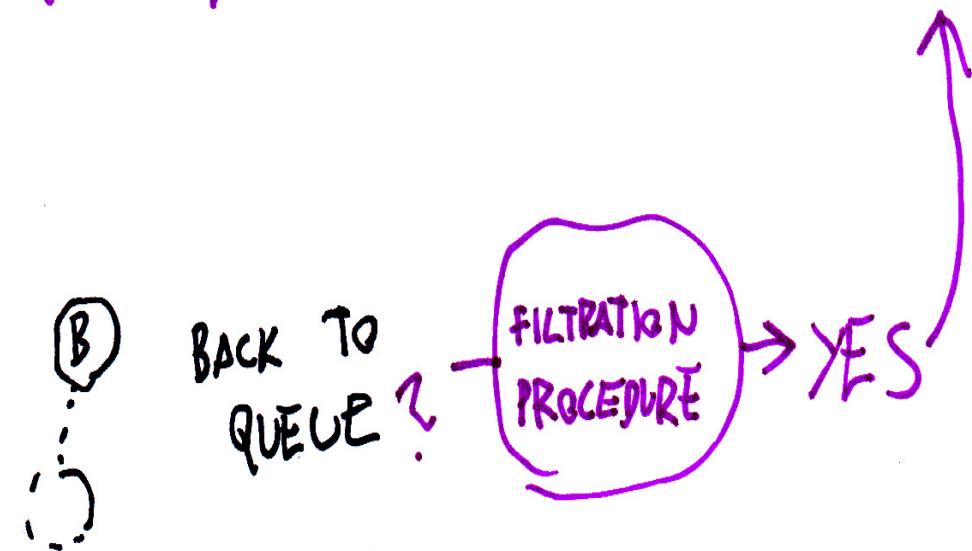


Jak znáhodnět A*

- Nastíněná metoda funguje jako systematická enumerace od nejmenšího po největší, ale základem GP generování je náhodnost
- Přidáme filtrační funkci, která na základě nějakého kriteria profiltaruje následníky
 - Naše geom ponechá následníka s pravděpodobností q^{depth} , kde q je parametr, my všude $q=0.75$ (**asi jí udelat zvlášť slajd**)
- Předělat předešlou ilustraci s tím, že tentokrát se někteří následníci zahodí
- Seriová verze se z týhle dostane tak, že filtr zahodí vše až na jednoho nasledníka

FORGETFUL A*

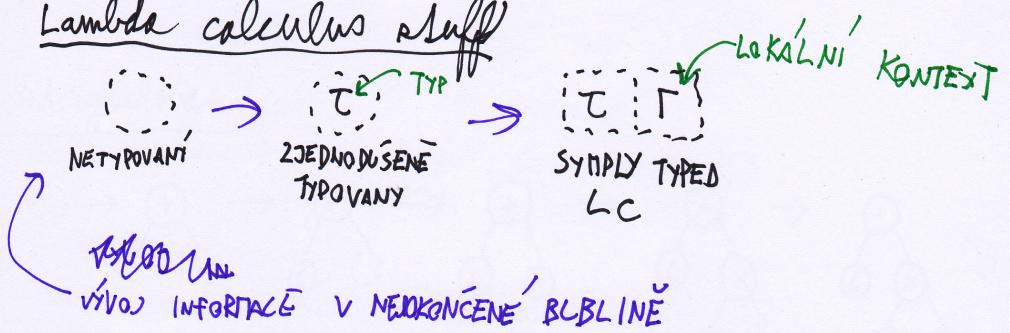
Jako předchozí, ale rád změnil EXPANSION podobně



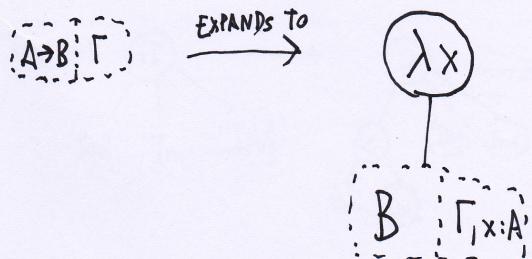
Zobecnění pro lambda calculus

- Dvě vety co je to LC
- Hlavní věc co s LC přichází jsou anonymní funkce, které zavádějí lokální proměnné
- Použití LC umožňuje použití stavebních symbolů se higher-order typy
- Unfinished leaf navíc obsahuje lokální context, rozšířený o lokální proměnné
- Dva obrázky
 - Expand šipkového typu
 - Expand nešipkového typu

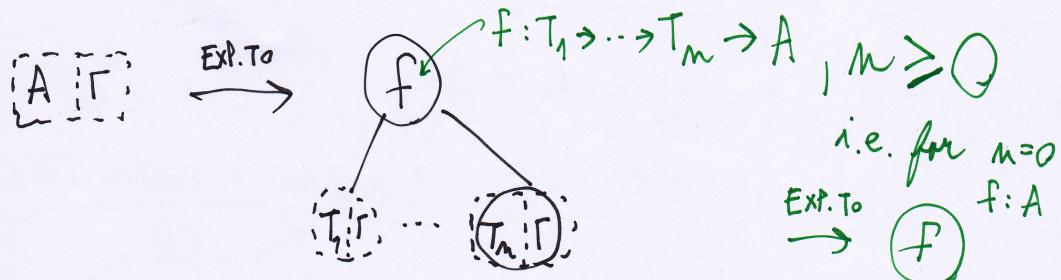
Lambda calculus stuff



EXPAZÉ ŠÍPKOVÉHO TYPU:



EXPAZÉ NE-ŠÍPKOVÉHO TYPU:



PŘÍKLAD VYGENEROVÁNÍ KOREKTNÍHO JEDINCE Z EVEN-PARTY.

TODO: $\lambda x. \text{foldr}(\lambda y z. \text{moc}(\text{and} y z) (\text{moc} y z))$
 $(\text{moc} (\text{head}' x) (\text{head}' x)) (\text{tail}' x)$

Výsledky...

- 2-3 podle času

Conclusions

- Tadá!