Typované funkcionální genetické programování

aneb

Výlet do automatické syntézy programů

Tomáš Křen

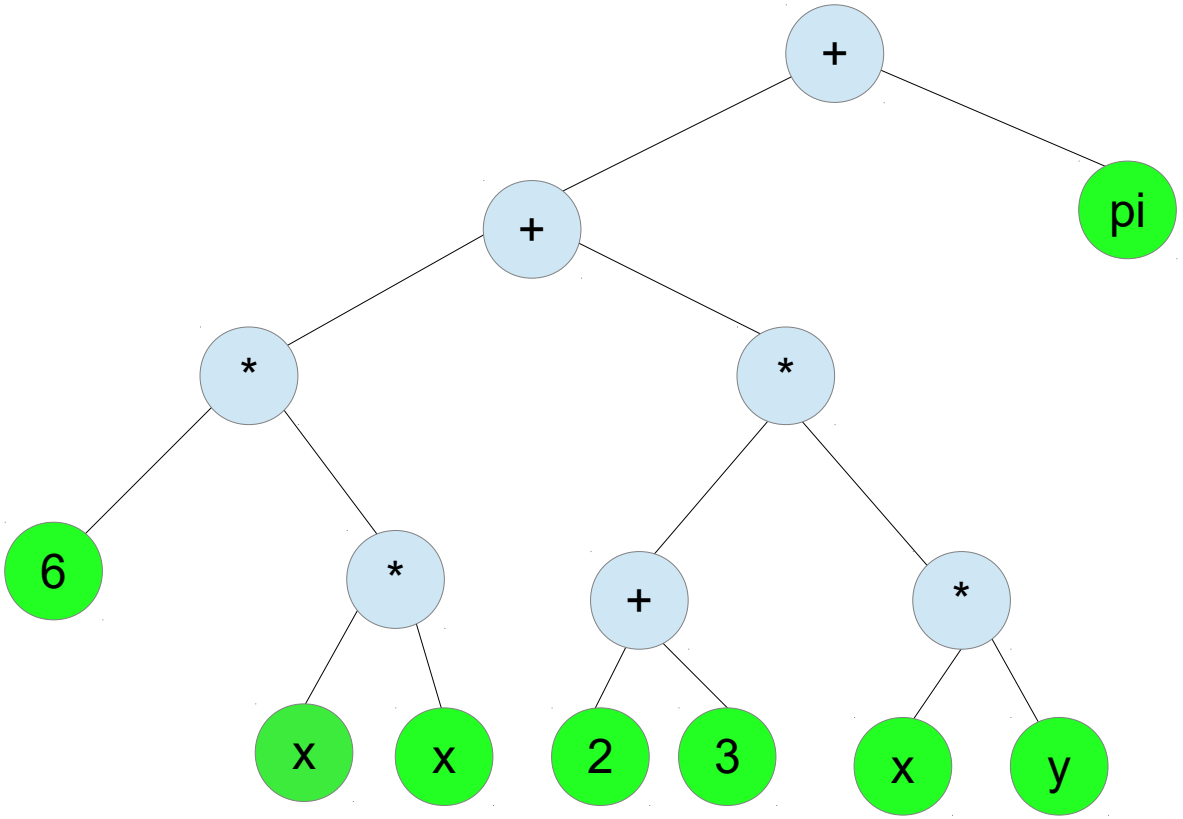
Ústav informatiky AV ČR & KTIML MFF UK

Co to je Genetické programování?

- Optimalizační metoda inspirovaná evolucí.
- Reprezentace řešení: **syntaktický strom programu**

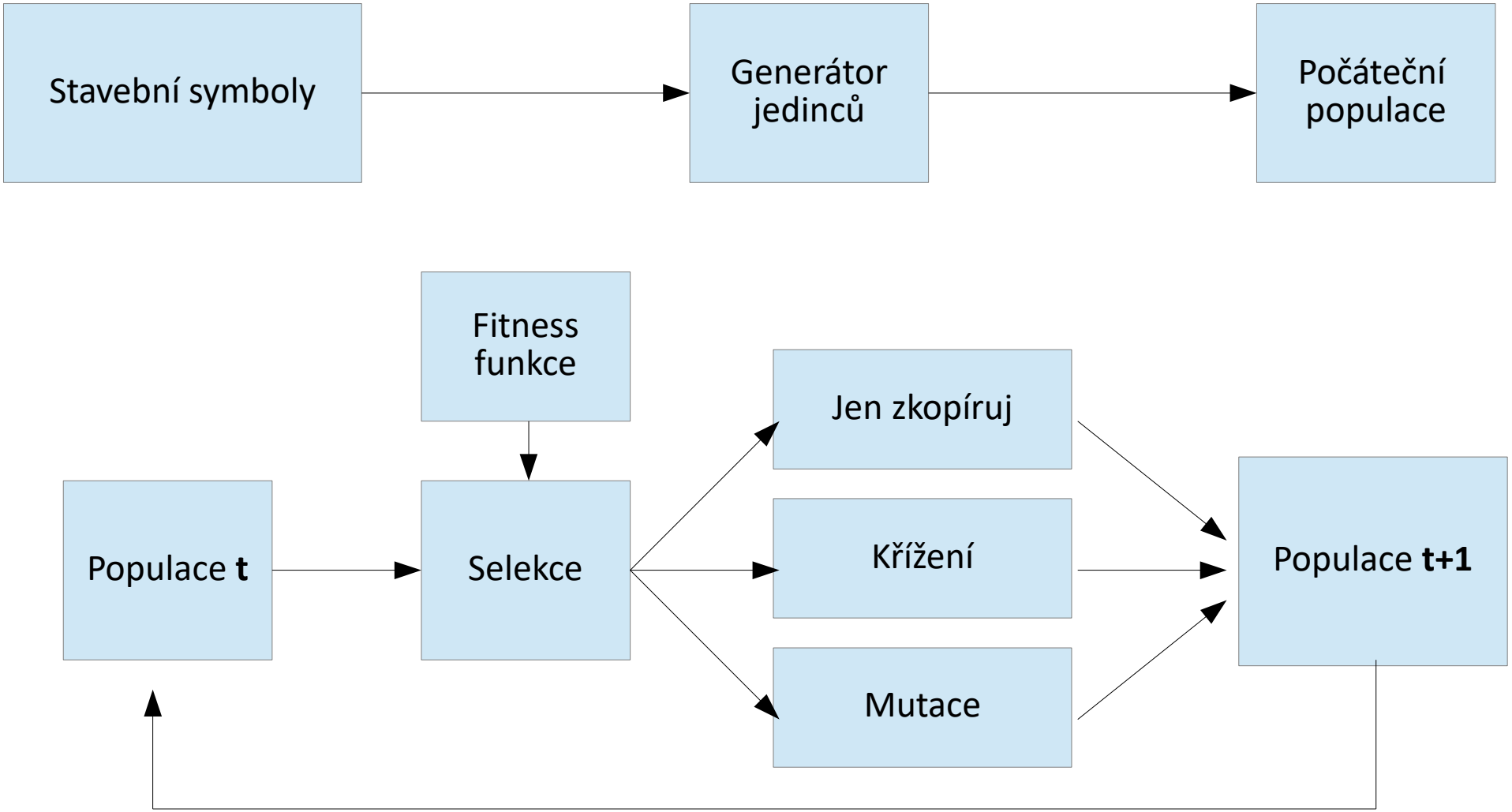
- **Hlavní vstupy**:
  - Fitness funkce
  - Množina stavebních symbolů
  - Genetické operátory (+ mechanizmus selekce)
- **Výstup:**
  - Programy

GP jedinec



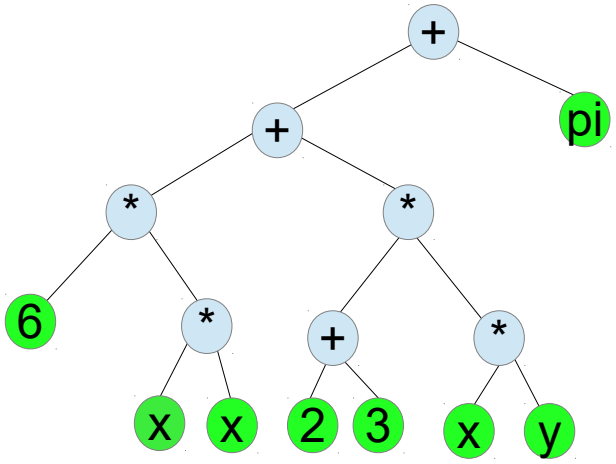

```
function(x,y){
  return ((6*(x*x))+((2+3)*(x*y)))+pi;
}
```

Jak to funguje?

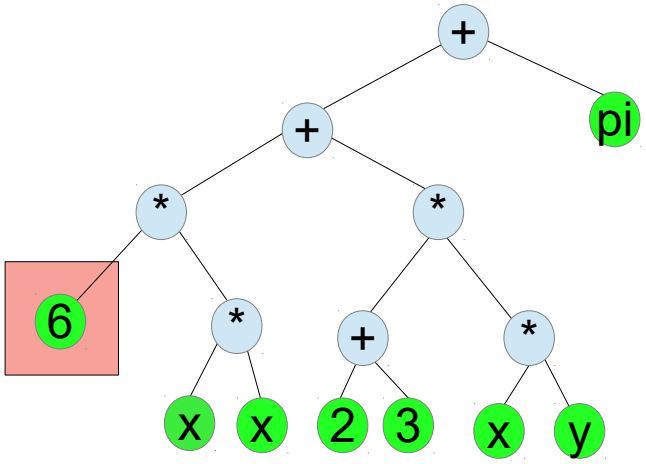


Mutace

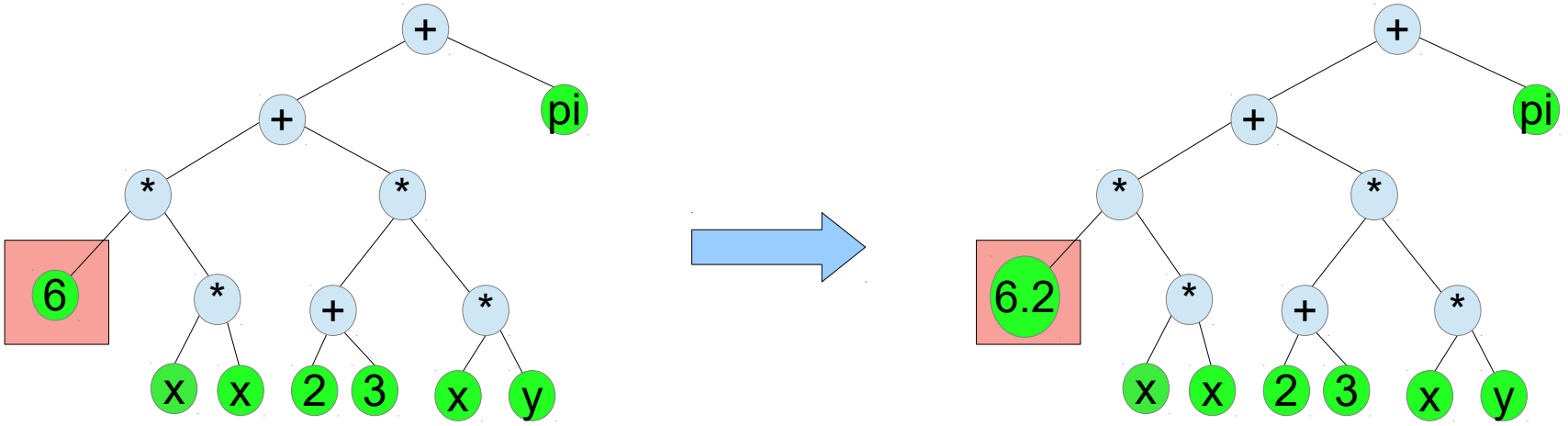
Mutace



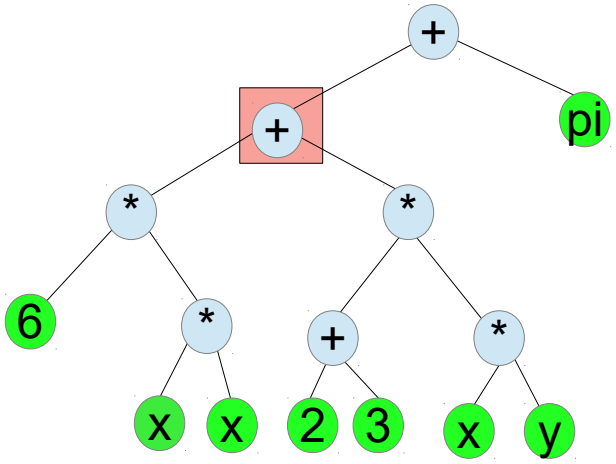
Mutace



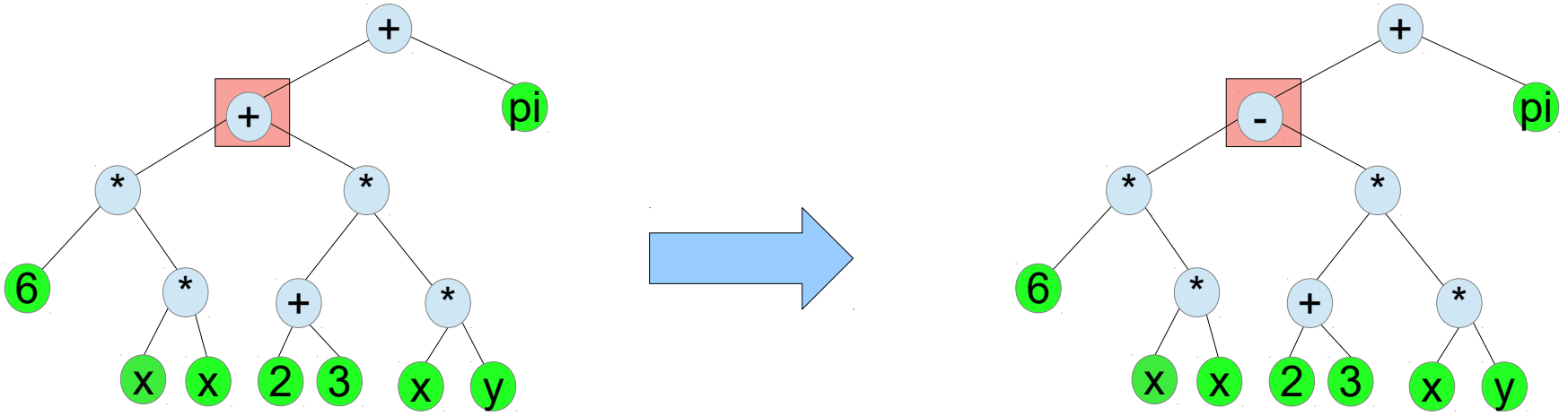
Mutace



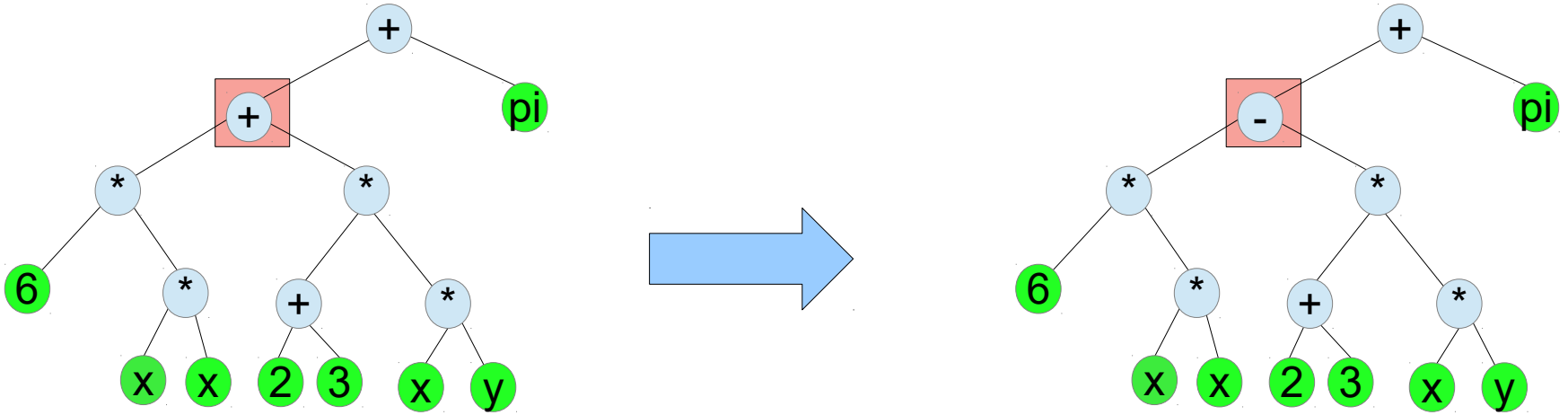
Mutace

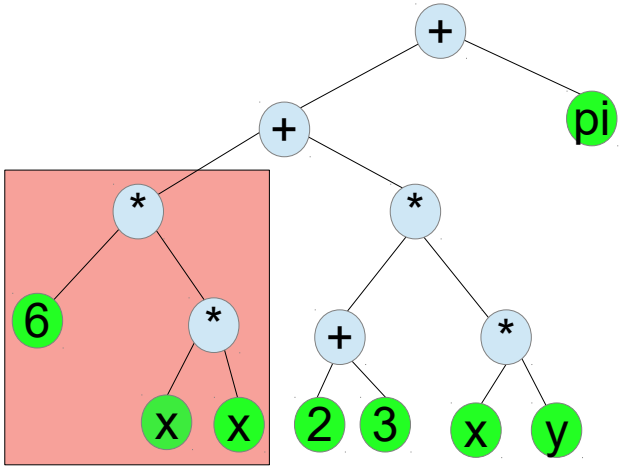


Mutace

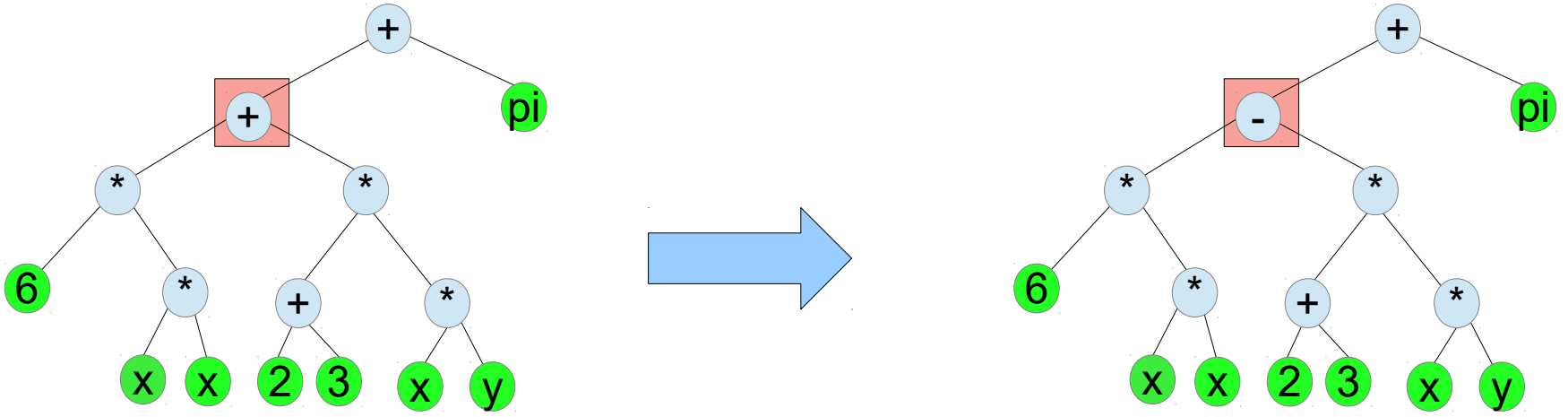


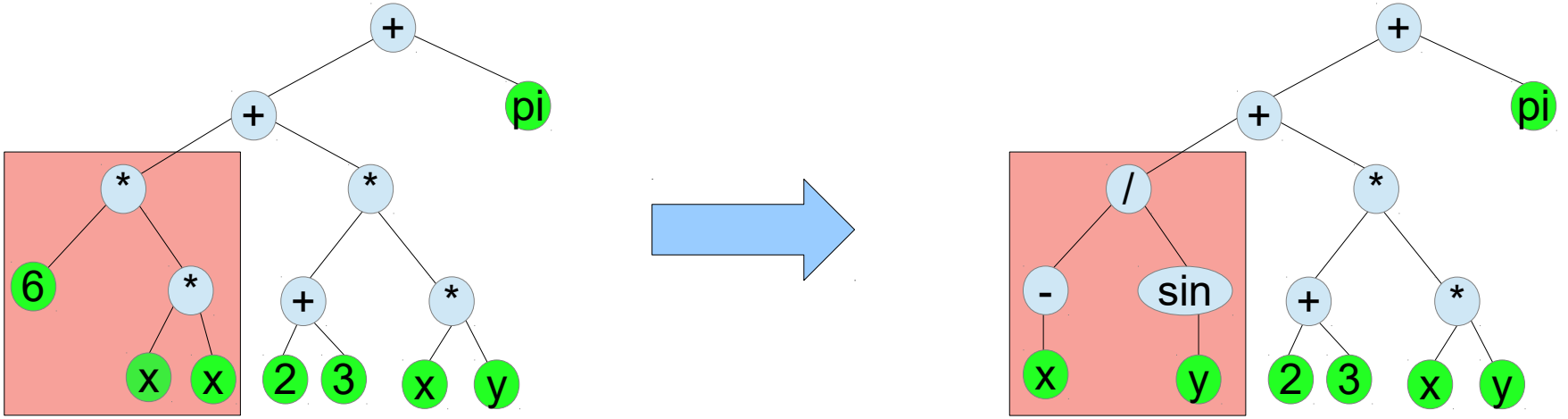
Mutace





Mutace

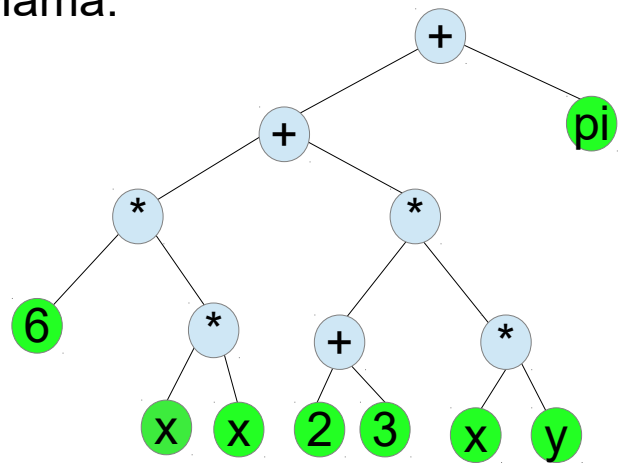




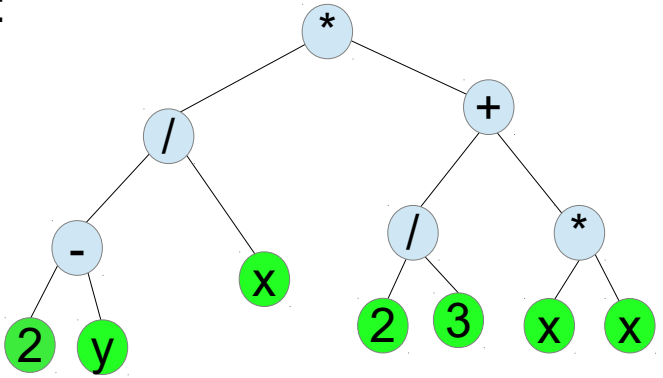
Křížení

Křížení

Máma:

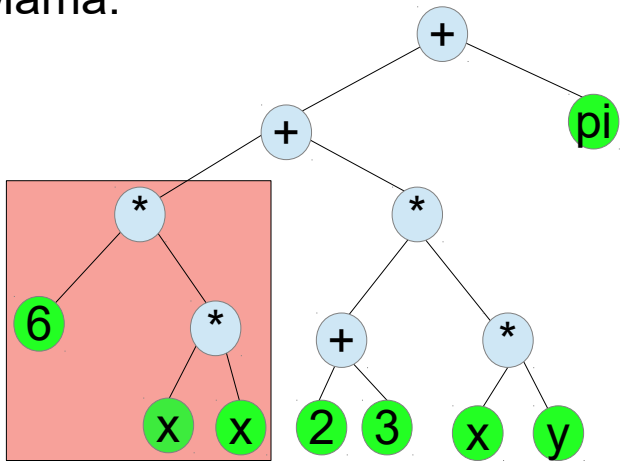


Táta:

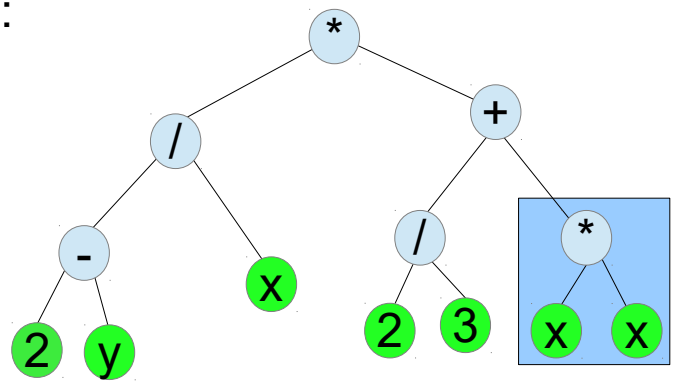


Křížení

Máma:

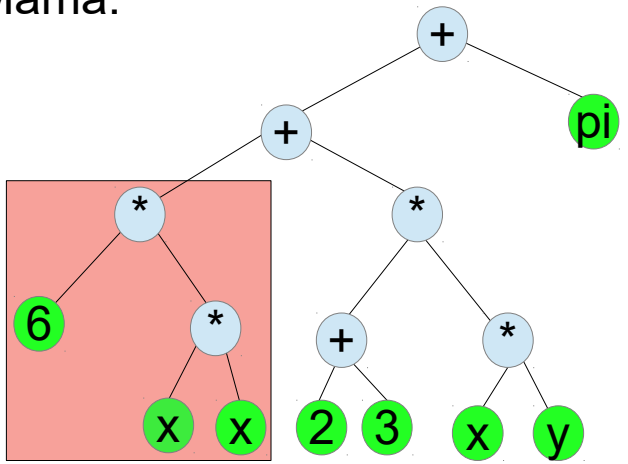


Táta:

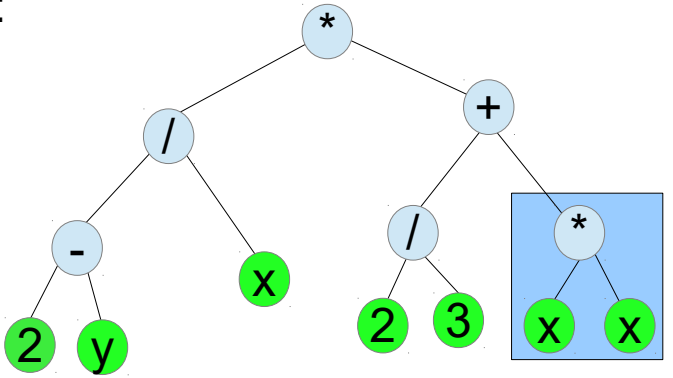


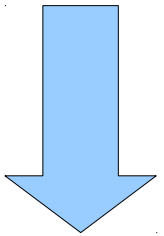
Křížení

Máma:

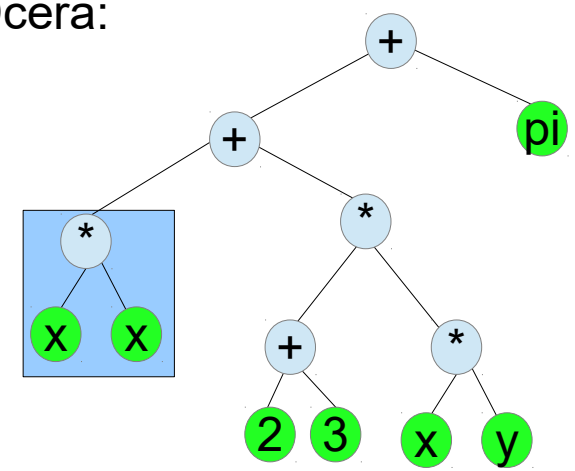


Táta:

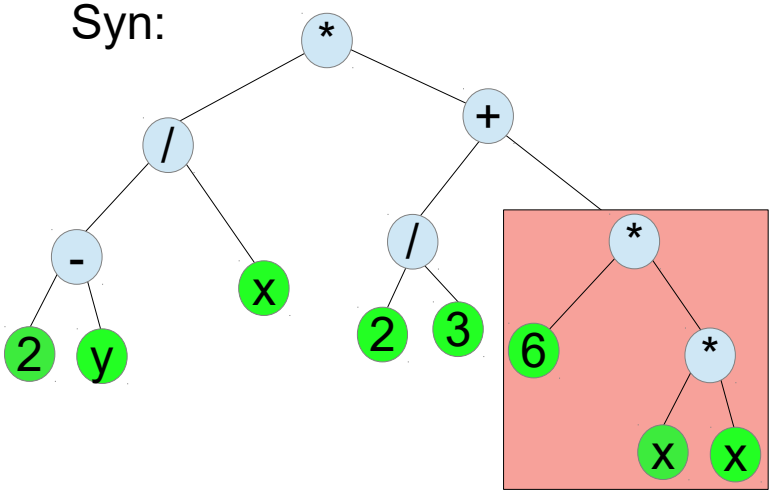




Dcera:



Syn:



Typy v GP

- Výhody
  - Do sady stavebních symbolů můžeme přidávat dle libosti
  - Vhodně volená typová omezení činí programy smysluplnější
  - Prohledávaný prostor je menší

- Nevýhody
  - Algoritmus GP se zkomplikuje (generování, gen. operátory)
  - Pro moc silný typový systém může být problém vůbec najít korektního jedince

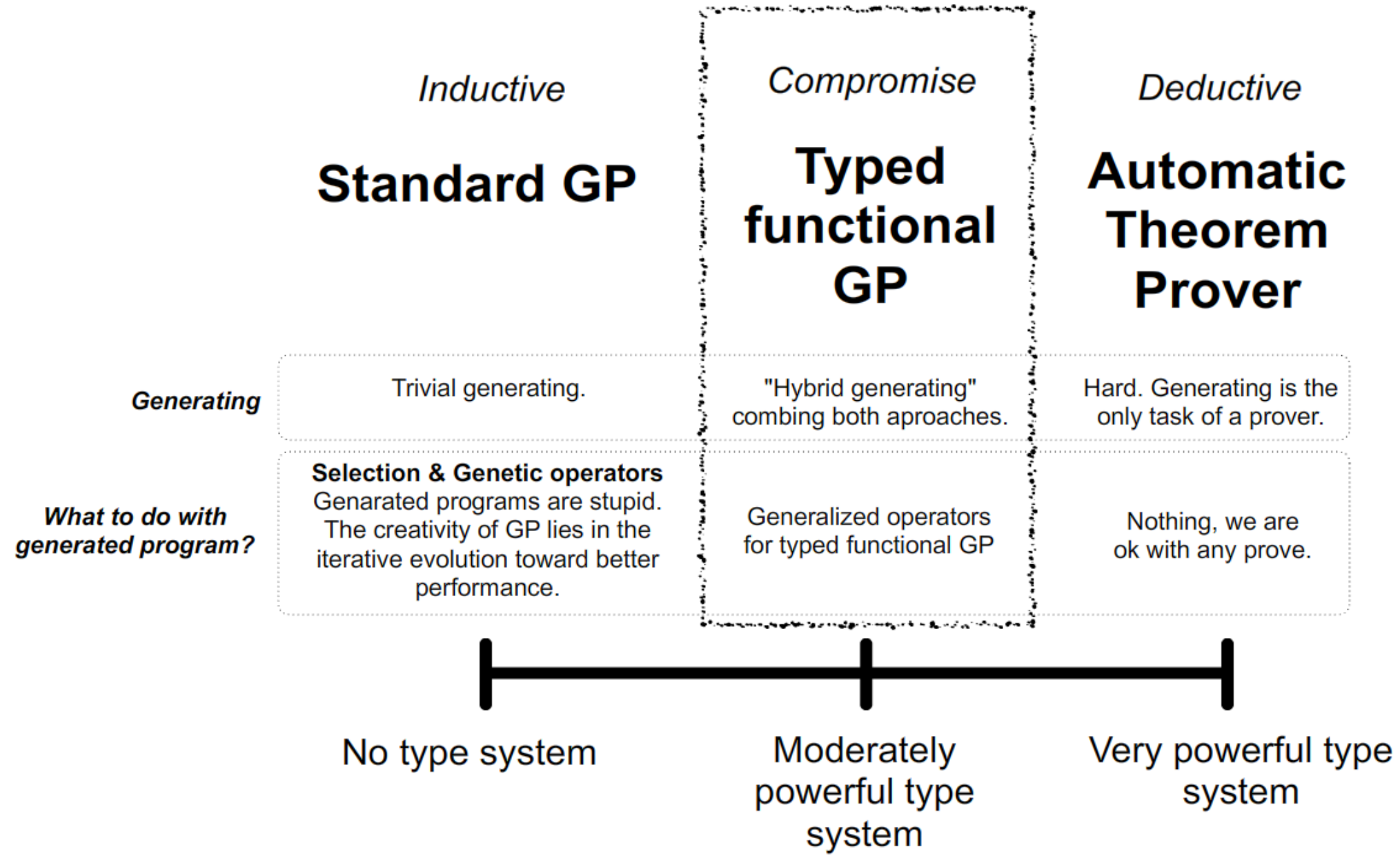
Výhody použití přístupů z funkcionálního programování v GP

- Komplexní nebo obecné programovací konstrukty mohou být reprezentovány jako higher-order funkce (které můžeme dát do množiny stavebních symbolů)
- Typy umožňují precizní způsob jak vynucovat tvrdá omezení na strukturu generovaných programů.
- *Curry-Howardův izomorfismus ...*

Curry-Howardův izomorfismus

Svět logiky	Druh logiky	Tvrzení	Důkaz
Svět typů	Typový systém	Typ	Program

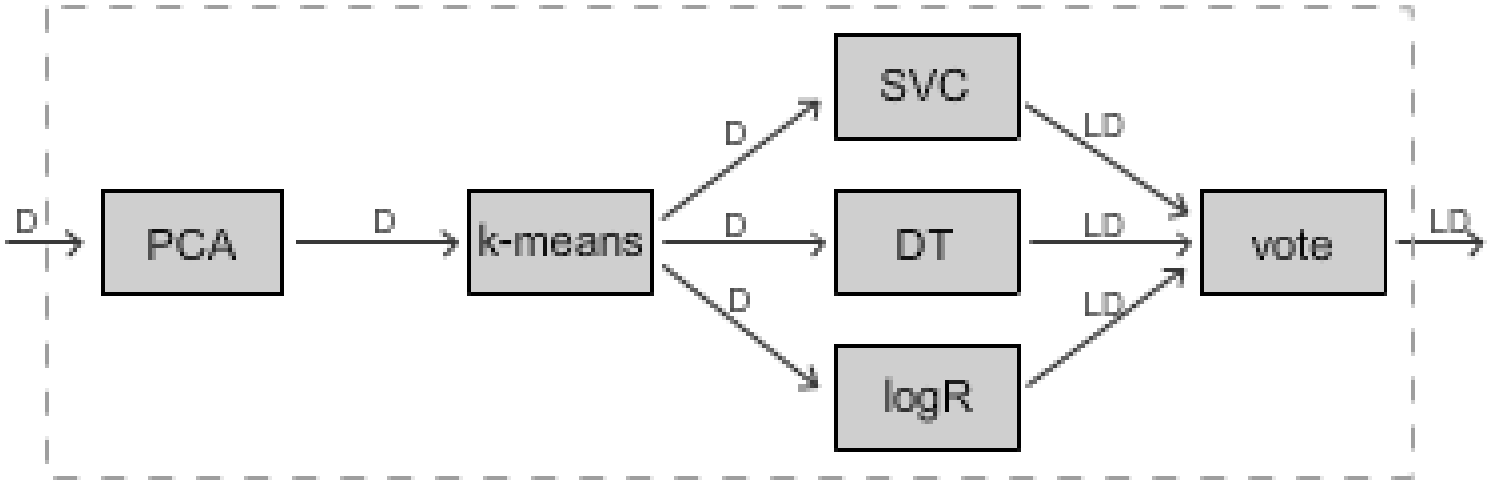
Curry-Howardův izomorfismus



První příklad aplikace TFGP: **Evoluce ML workflows**

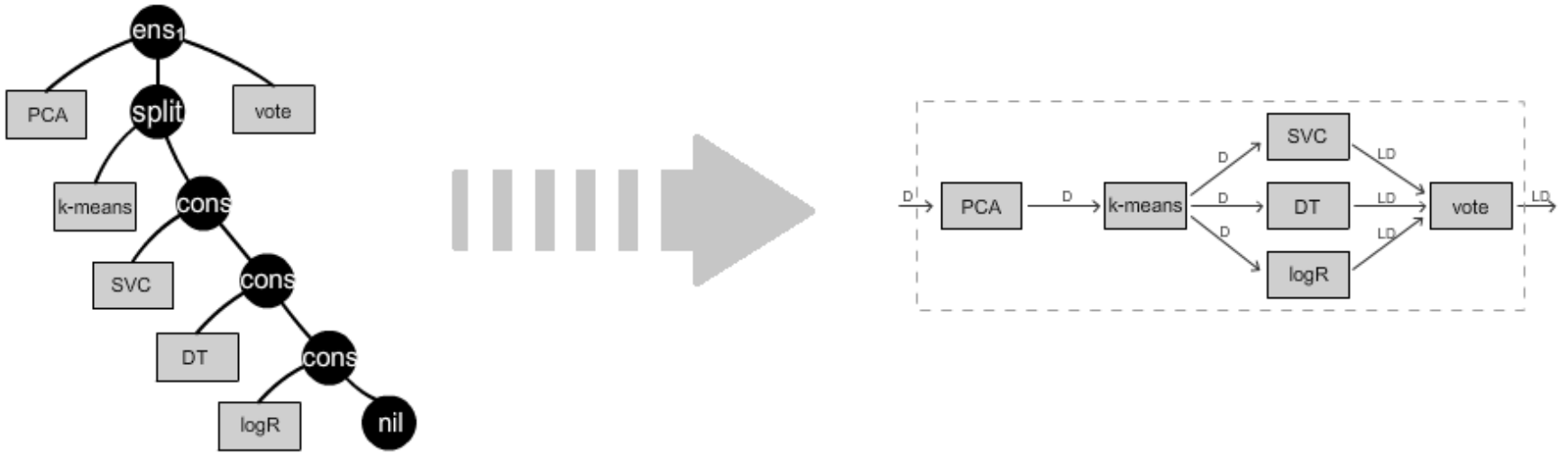
- Existuje mnoho *Machine Learning* metod
  - A každá se funguje různě kvalitně na různých datasetech

- **Workflow** : Kombinace několika jednoduchých metod do jedné komplikovanější.
  - Může výrazně zlepšit výsledky



Naše workflow reprezentace

- ML workflow lze přirozeně reprezentovat jako **Directed Acyclic Graph** (DAG)
- … který reprezentujeme jako strom programu popisující konstrukci onoho DAGu
  - Operacemi ***kombinujícími menší DAGy do větších***



- **Listy** odpovídají ML metodám
  - Tedy v zásadě funkcím
- A tedy **vnitřní vrcholy** odpovídají “higher order funkcím”
  - Typy nám pomohou zvládnout, aby vznikaly jen smysluplné kombinace

Klasifikátor

- Natrénovaný klasifikátor lejbluje data
  - transformuje Data na Labeled Data

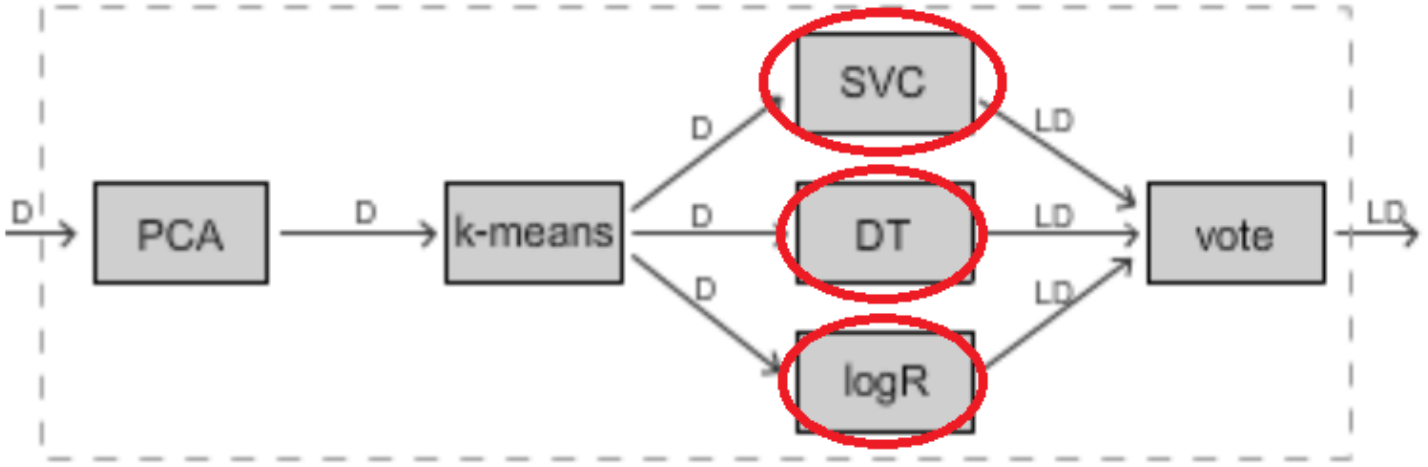
- používáme:
  - Support vector classifier
  - Logistic regression
  - Gaussian naive Bayes
  - Decision tree
  - Perceptron
  - MLP
  - LDA
  - QDA
  - PAC
  - SGD

Data (D)

volatile	chlorid	pH	alcoł
0,26	0,049	3,5	10,5
0,67	0,074	3,1	9,3
0,27	0,052	3,4	10
0,27	0,047	3,2	11,6
0,32	0,033	3,1	12,3
0,24	0,044	3,2	10

Labeled Data (LD)

volatile	chlorid	pH	alcoł	quality
0,26	0,049	3,5	10,5	8
0,67	0,074	3,1	9,3	5
0,27	0,052	3,4	10	6
0,27	0,047	3,2	11,6	6
0,32	0,033	3,1	12,3	7
0,24	0,044	3,2	10	6



typ:

D => LD

Preprocessing

- Preprocessing transformuje data na trochu jiná data

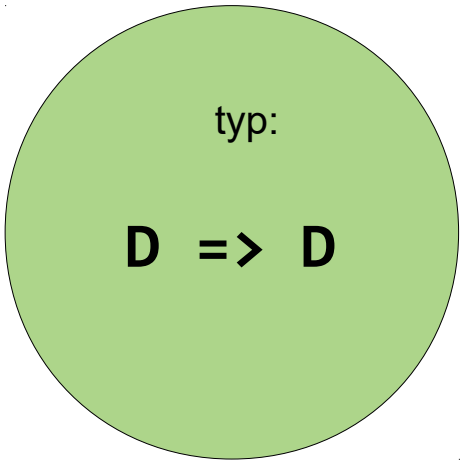
- používáme:
  - k-best
  - PCA

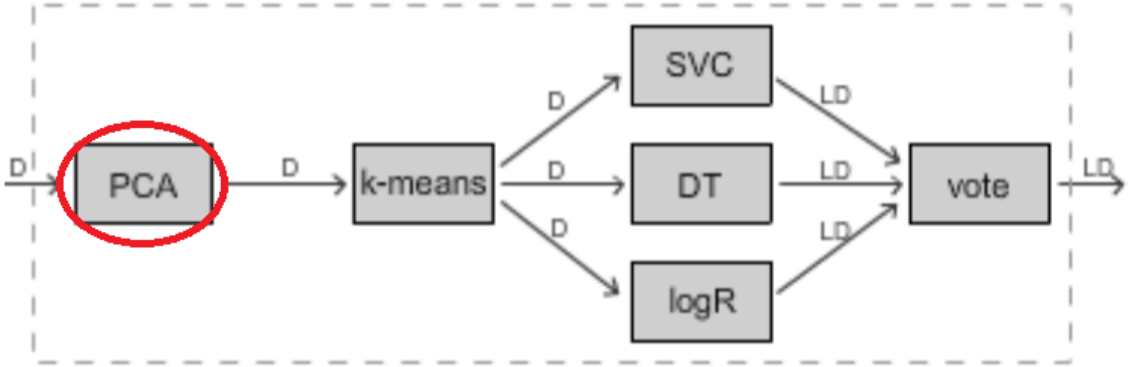
Data (D)

fixed	volatile	citric	residi	chlorid	free	total	density	pH	sulpha	alcol
6,8	0,26	0,42	1,7	0,049	41	122	0,993	3,5	0,48	10,5
7,6	0,67	0,14	1,5	0,074	25	168	0,9937	3,1	0,51	9,3
6,6	0,27	0,41	1,3	0,052	16	142	0,9951	3,4	0,47	10
7,4	0,27	0,48	1,1	0,047	17	132	0,9914	3,2	0,49	11,6
7,2	0,32	0,36	2	0,033	37	114	0,9906	3,1	0,71	12,3
8,5	0,24	0,39	10,4	0,044	20	142	0,9974	3,2	0,53	10

Data (D)

volatile	chlorid	density	alcol
0,26	0,049	0,993	10,5
0,67	0,074	0,9937	9,3
0,27	0,052	0,9951	10
0,27	0,047	0,9914	11,6
0,32	0,033	0,9906	12,3
0,24	0,044	0,9974	10





Splitter

- Splitter rozděluje tok dat do několika větví.

- We use:
  - k-means
  - copy

Data (D)

volatile	chlorid	density	alcol
0,26	0,049	0,993	10,5
0,67	0,074	0,9937	9,3
0,27	0,052	0,9951	10
0,27	0,047	0,9914	11,6
0,32	0,033	0,9906	12,3
0,24	0,044	0,9974	10

Data (D)

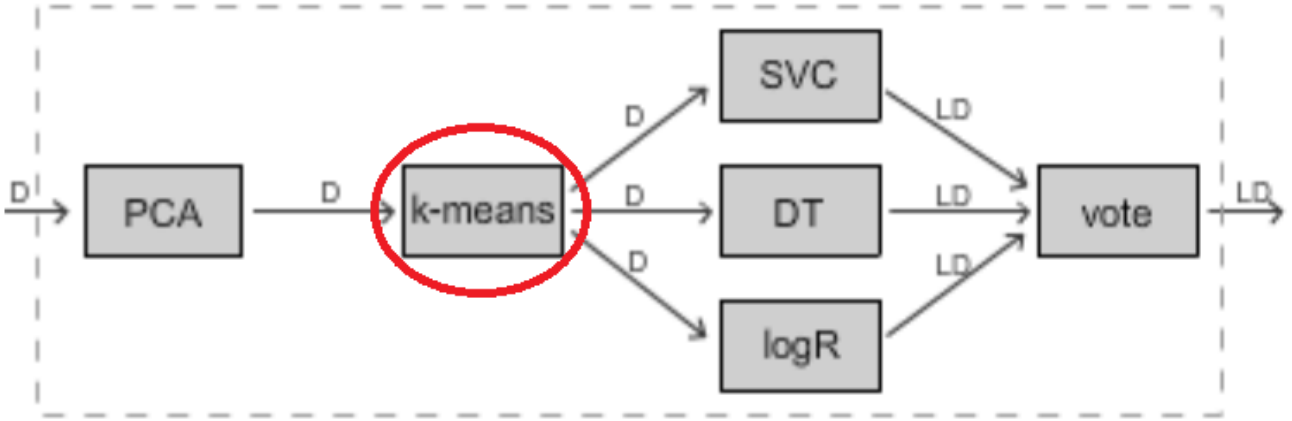
volatile	chlorid	density	alcol
0,26	0,049	0,993	10,5
0,67	0,074	0,9937	9,3
0,27	0,047	0,9914	11,6
0,32	0,033	0,9906	12,3

Data (D)

volatile	chlorid	density	alcol
0,27	0,052	0,9951	10
0,24	0,044	0,9974	10

typy:

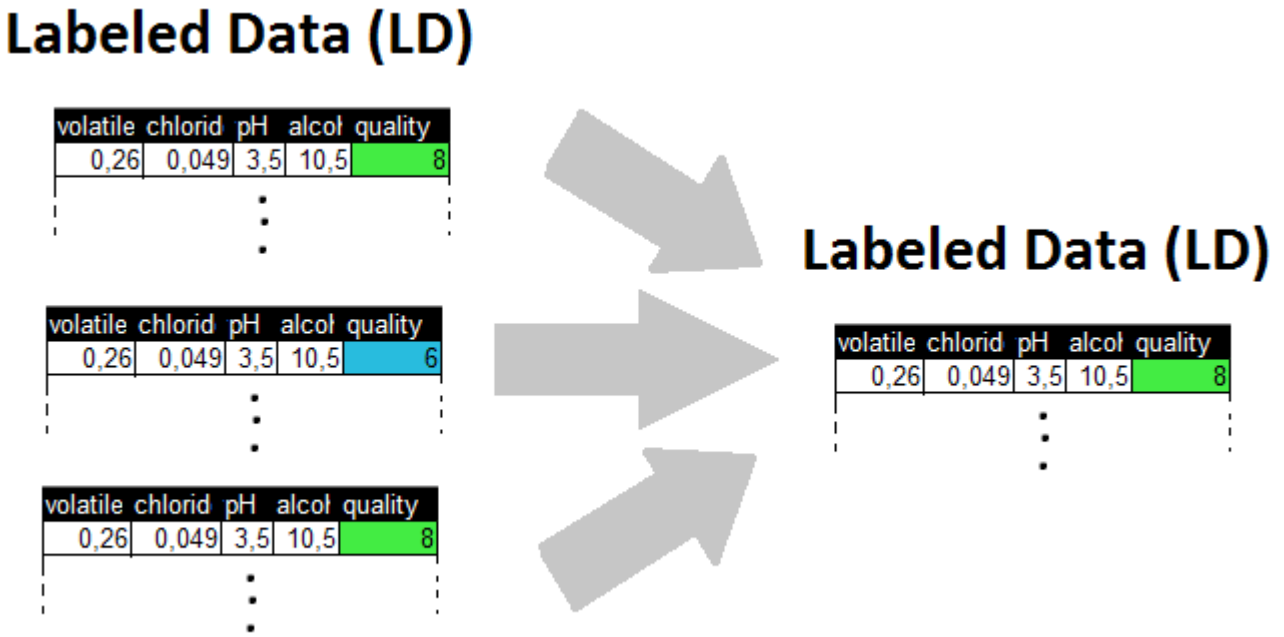
```
kMeans : D => (V D (S(S n)) Disj)
copy   : D => (V D (S(S n)) Copy)
```

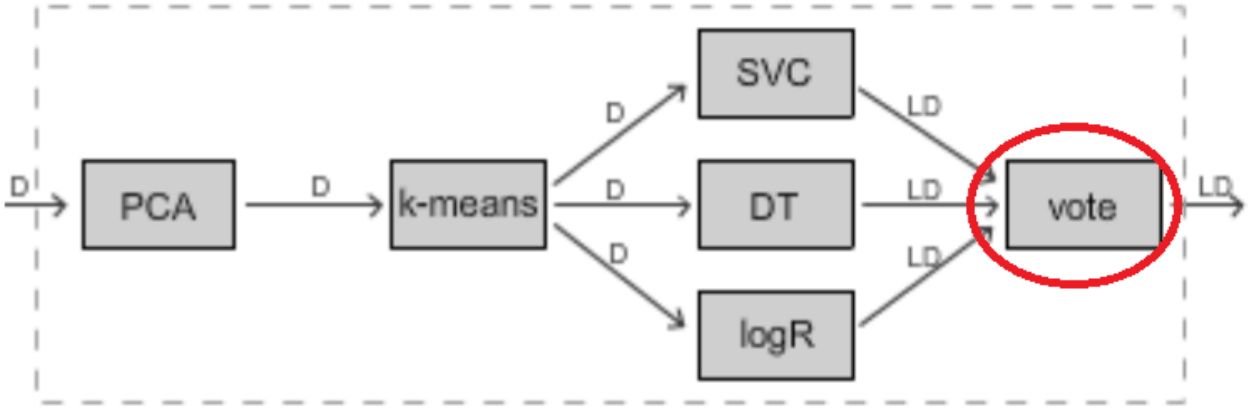


Merger

- A merger je taky dat zase spojuje dohromady.

- We use:
  - Union / Voting
  - Stacking

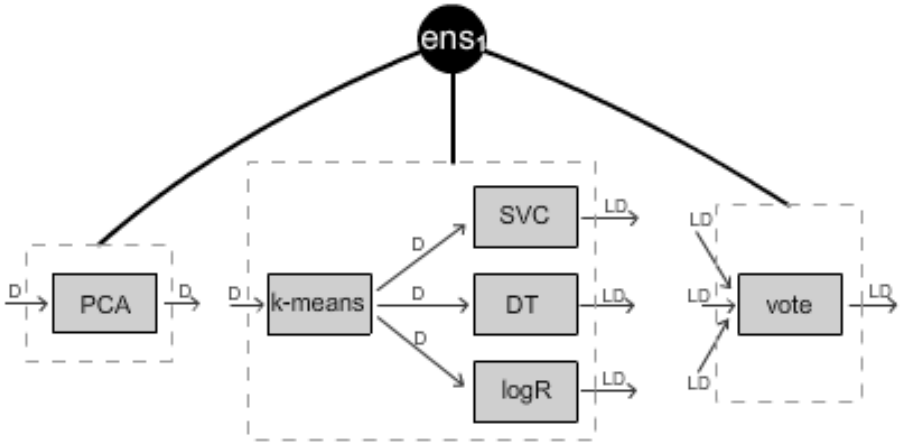


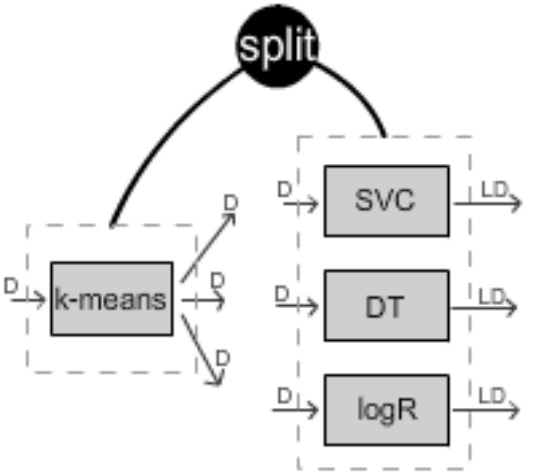


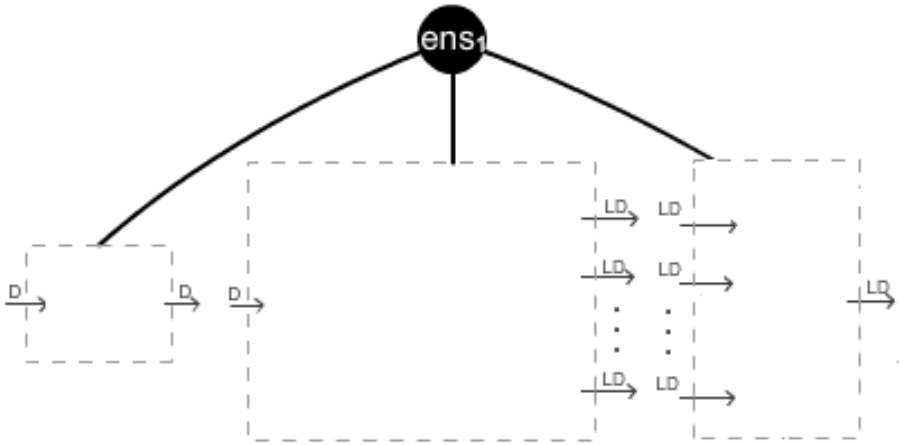
typy:

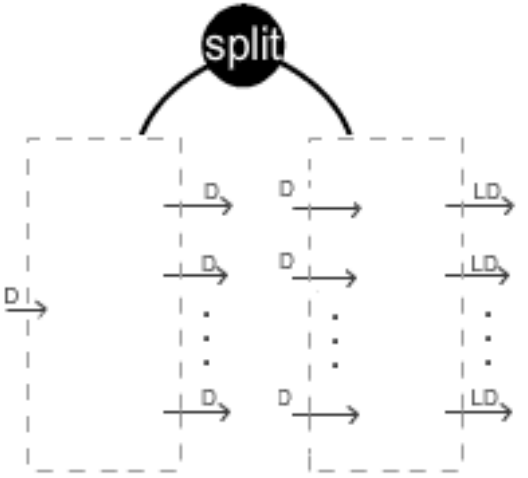
```
vote    : (V LD (S(S n)) an  ) => LD
Stacker : (V LD (S(S n)) Copy) => D
```

Sériové zapojení



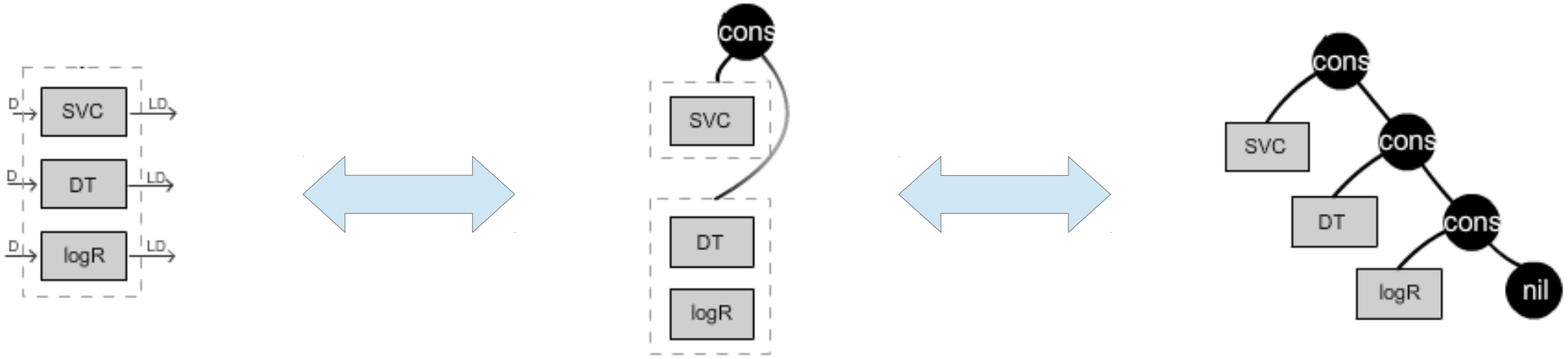




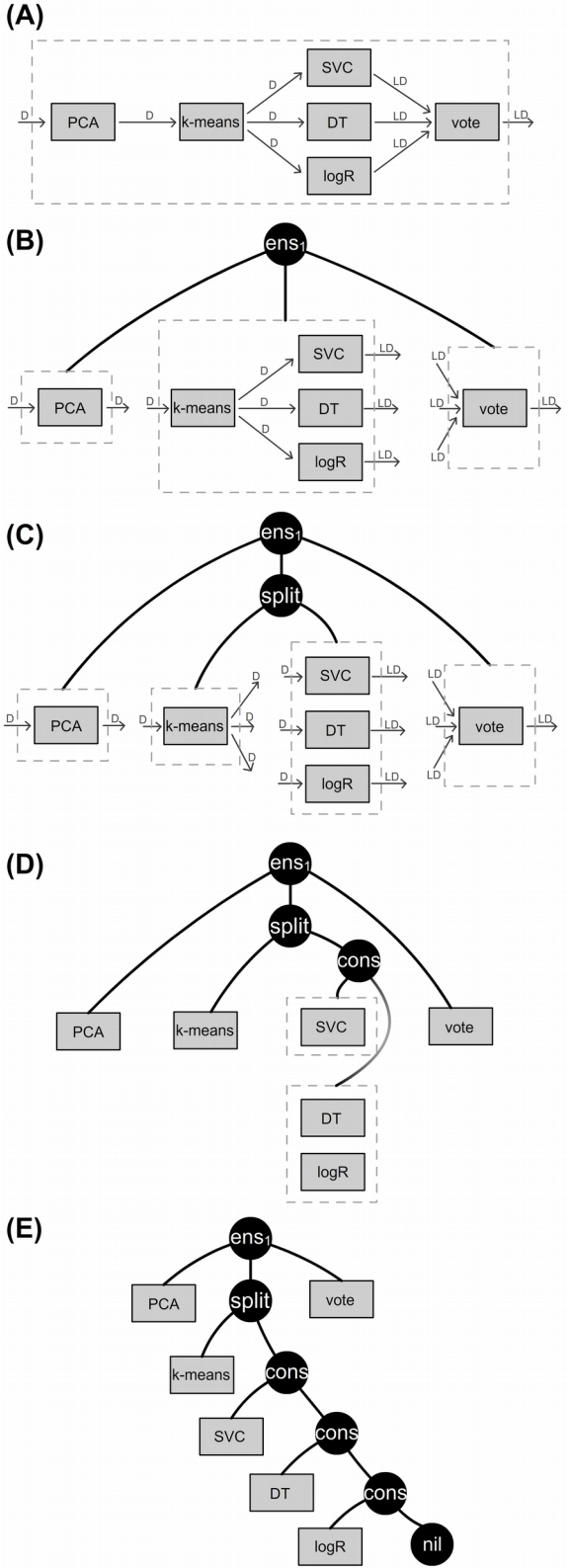



```
ens1 : ( (D => D) × (D => (V LD n an)) × ((V LD n an) => LD)) ) -> (D => LD)

split : ( (D => (V D n an)) × (V (D => LD) n an) )    ->    (D => (V LD n an))
```

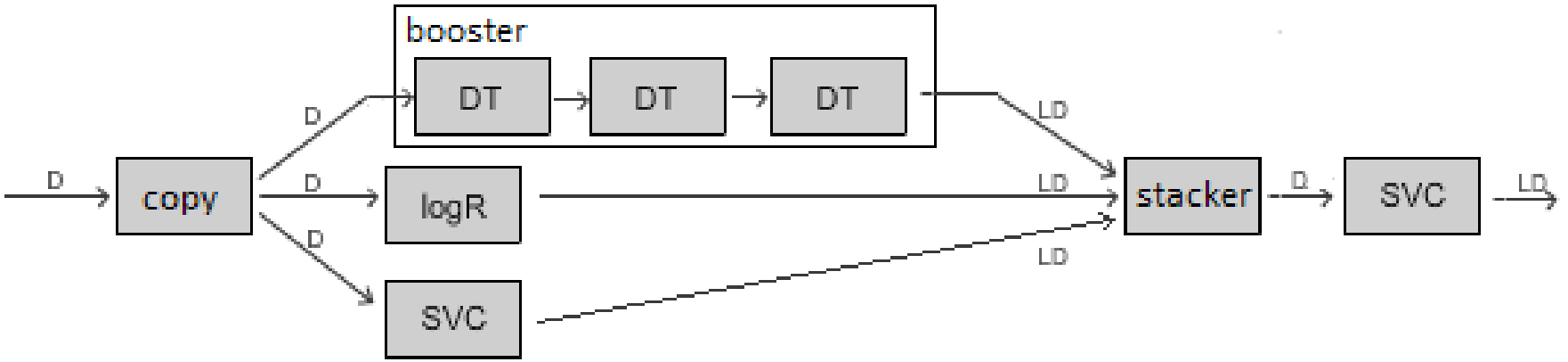
.. a **paralelní zapojení**




```
cons  :  (a × (V a n an)) -> (V a (S n) an)

 nil  :  V a 0 an
```


(A)
PCA
k-means
SVC
DT
logR
vote
D
LD
(B)
ens1
(C)
split
(D)
cons
(E)
nil

… a taky
Stacking & **Boosting**
ensemblové metody



- **Stacking** – Kombinuje predikce předchozích klasifikátorů jako nový dataset.
- **Boosting** – Ensemble tvořený serií klasifikátorů, kde každej další klasifikátor se soustředí na chybně klasifikované předpovědi klasifikátorů před ním.

Stavební symboly

Vnitřní vrcholy

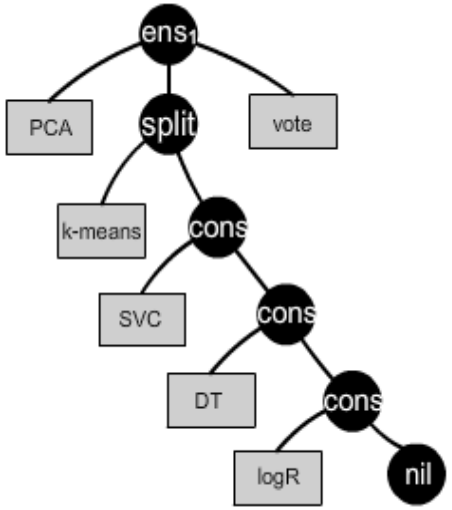
```
ens1
ens0
split
cons
stacking
boosting
booster
```

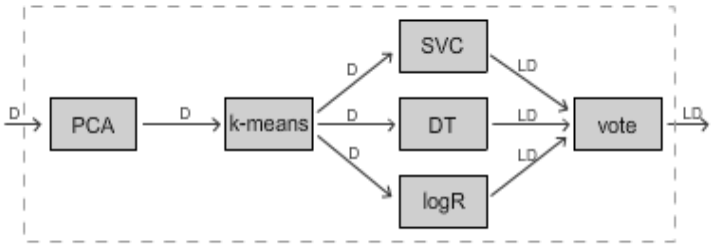
Listy

```
PCA
kBest
kMeans
copy
vote
booBegin
booEnd
stacker
nil
```

```
SVC
logR
gaussianNB
DT
Perc
LDA
QDA
PAC
SGD
MLP
```

+ set of parameters for the ML methods





Stavební symboly

Vnitřní vrcholy

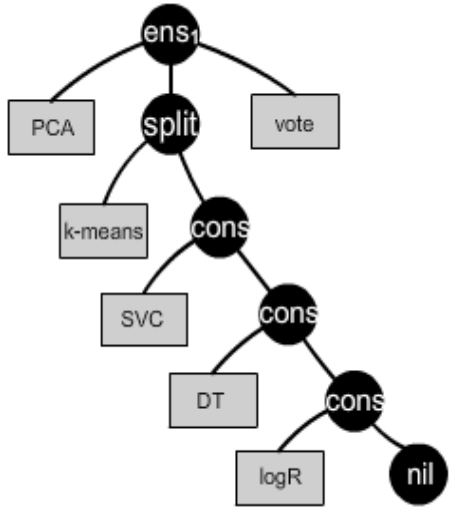
```
ens1     : ((D => D) × (D => (V LD n an)) × ((V LD n an) => LD)) -> (D => LD)
ens0     : ((D => (V LD n an)) × ((V LD n an) => LD)) -> (D => LD)
split    : ((D => (V D n an)) × (V (D => LD) n an)) -> (D => (V LD n an))
cons     : (a × (V a n an)) -> (V a (S n) an)
stacking : ((V LD n Copy) => D) × (D => LD)) -> ((V LD n Copy) => LD)
boosting : ((D => Boo) × (V (Boo => Boo) (S(S n)) an) × (Boo => LD)) -> (D => LD)
booster  : (D => LD) -> (Boo => Boo)
```

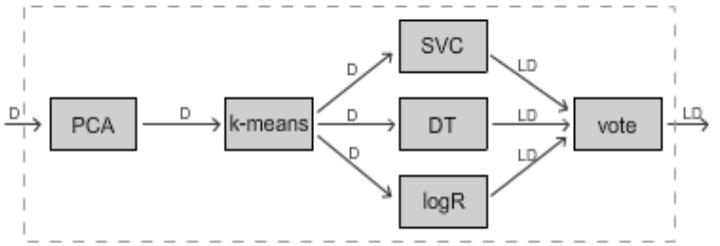
Listy

```
PCA        : D => D
kBest      : D => D
kMeans     : D => (V D (S(S n)) Disj)
copy       : D => (V D (S(S n)) Copy)
vote       : (V LD (S(S n)) an) => LD
booBegin   : D => Boo
booEnd     : Boo => LD
stacker    : (V LD (S(S n)) Copy) => D
nil        : V a 0 an
```

```
SVC        : D => LD
logR       : D => LD
gaussianNB : D => LD
DT         : D => LD
Perc       : D => LD
LDA        : D => LD
QDA        : D => LD
PAC        : D => LD
SGD        : D => LD
MLP        : D => LD
```

+ set of parameters for the ML methods





Stavební symboly

Vnitřní vrcholy

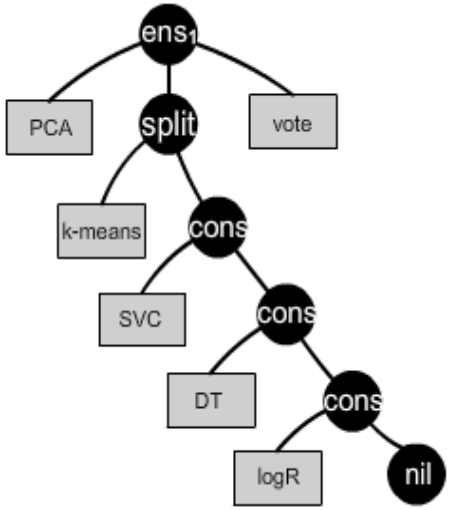
```
ens1     : ((D => D) × (D => (V LD n an)) × ((V LD n an) => LD)) -> (D => LD)
ens0     : ((D => (V LD n an)) × ((V LD n an) => LD)) -> (D => LD)
split    : ((D => (V D n an)) × (V (D => LD) n an)) -> (D => (V LD n an))
cons     : (a × (V a n an)) -> (V a (S n) an)
stacking : ((V LD n Copy) => D) × (D => LD)) -> ((V LD n Copy) => LD)
boosting : ((D => Boo) × (V (Boo => Boo) (S(S n)) an) × (Boo => LD)) -> (D => LD)
booster  : (D => LD) -> (Boo => Boo)
```

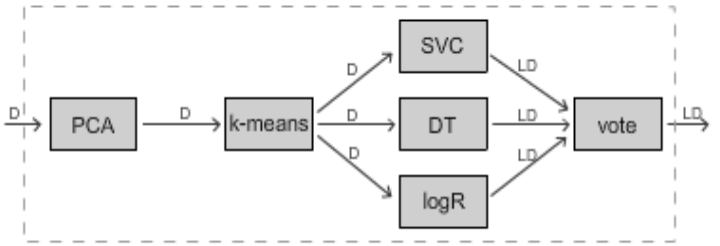
Listy

```
PCA        : D => D
kBest      : D => D
kMeans     : D => (V D (S(S n)) Disj)
copy       : D => (V D (S(S n)) Copy)
vote       : (V LD (S(S n)) an) => LD
booBegin   : D => Boo
booEnd     : Boo => LD
stacker    : (V LD (S(S n)) Copy) => D
nil        : V a 0 an
```

```
SVC        : D => LD
logR       : D => LD
gaussianNB : D => LD
DT         : D => LD
Perc       : D => LD
LDA        : D => LD
QDA        : D => LD
PAC        : D => LD
SGD        : D => LD
MLP        : D => LD
```

+ set of parameters for the ML methods

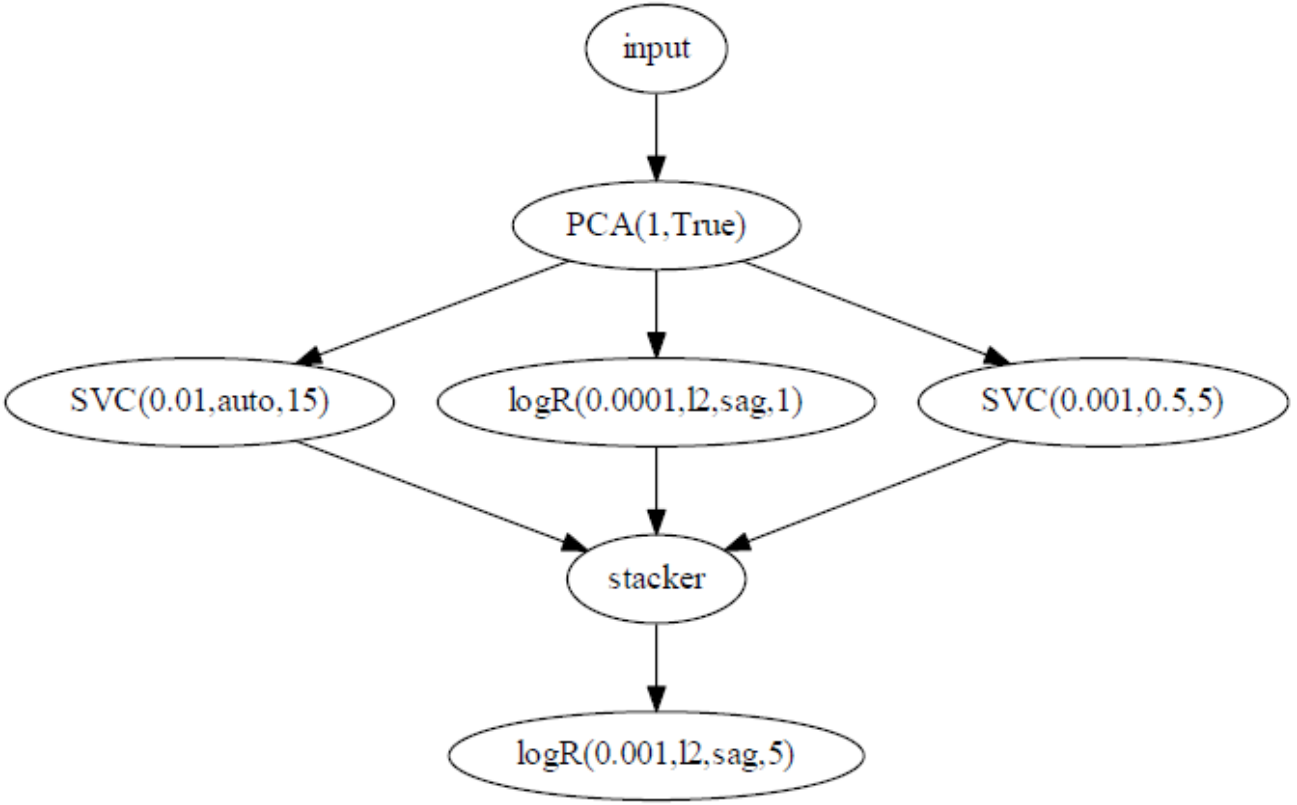




Genetické operátory: **Generování, Křížení & Mutace**

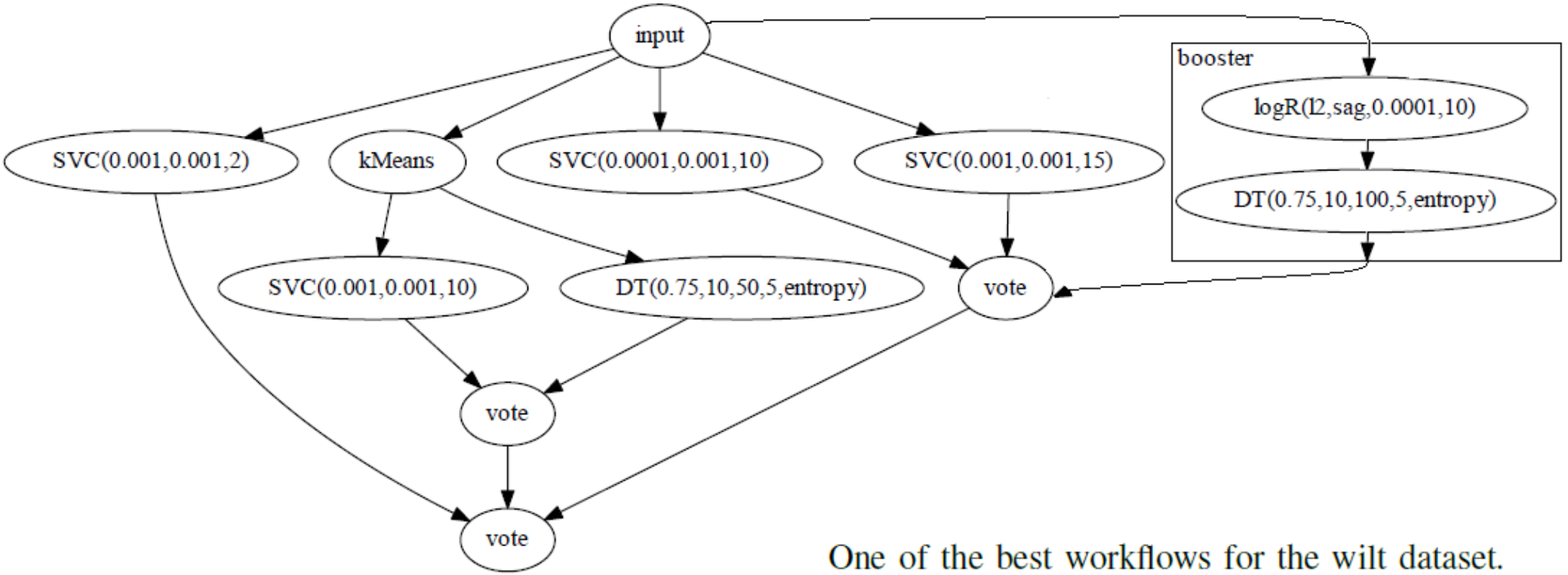
- Používáme náš nový algoritmus pro rychle generování stromů
  - Vhodný pro parametricky polymorfický typový systém
  - Schopný uniformního generování
- Typovaná verze tree-swapping křížení
- A několik mutací
  - Same size tree mutation
  - Mutation of single parameter
    (of machine learning method)

Příklady jedinců



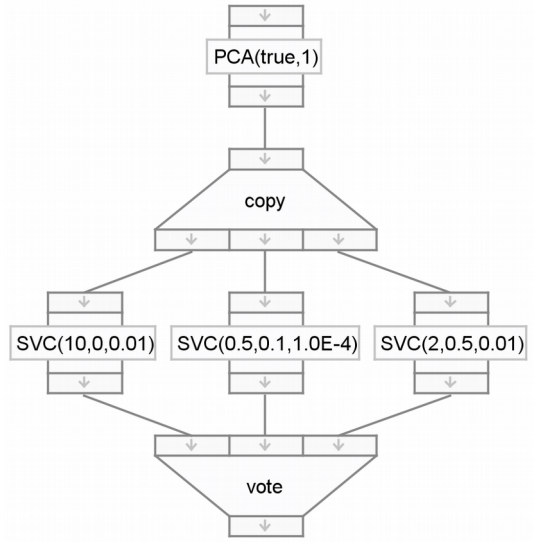
One of the best workflows for the magic dataset.

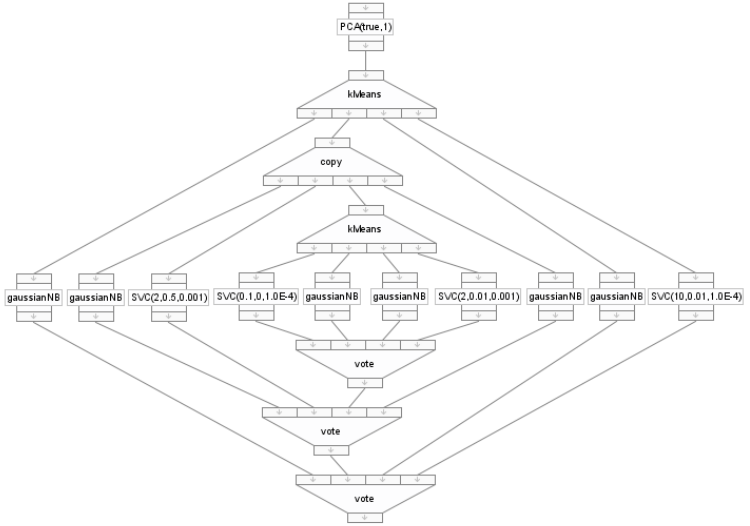
Příklady jedinců

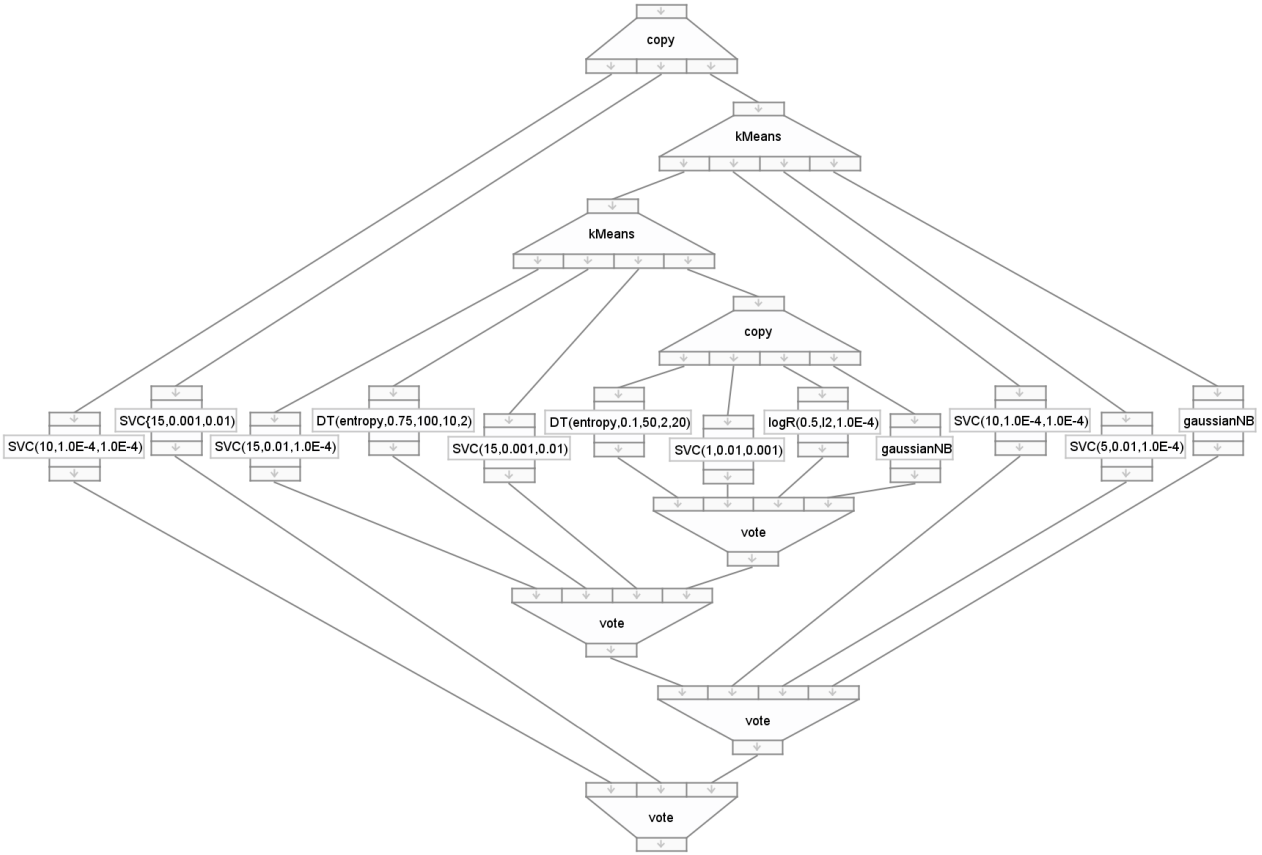


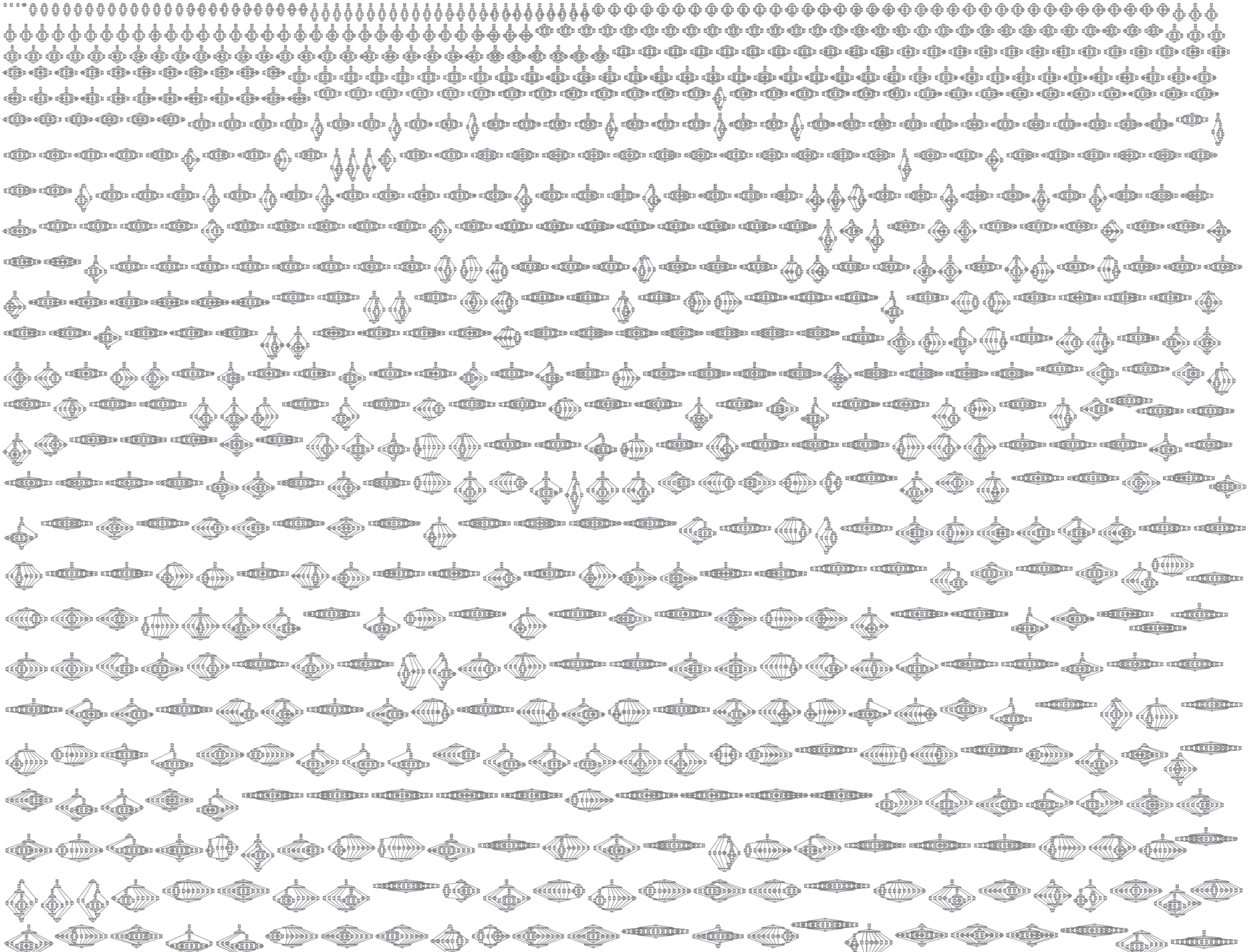
One of the best workflows for the wilt dataset.

Příklady jedinců

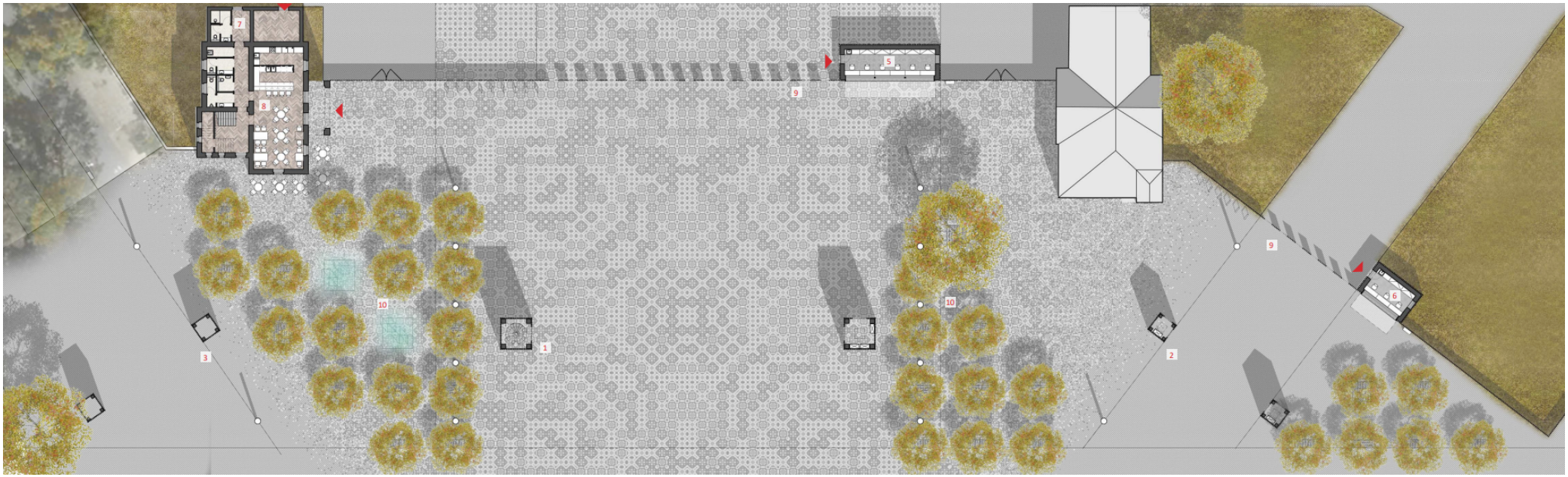




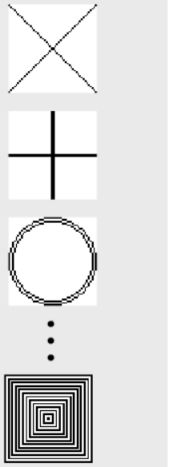




Druhý Příklad:
Interaktivní evoluce návrhů dlažby



Makro semínka
(počáteční *makro návrhy)*



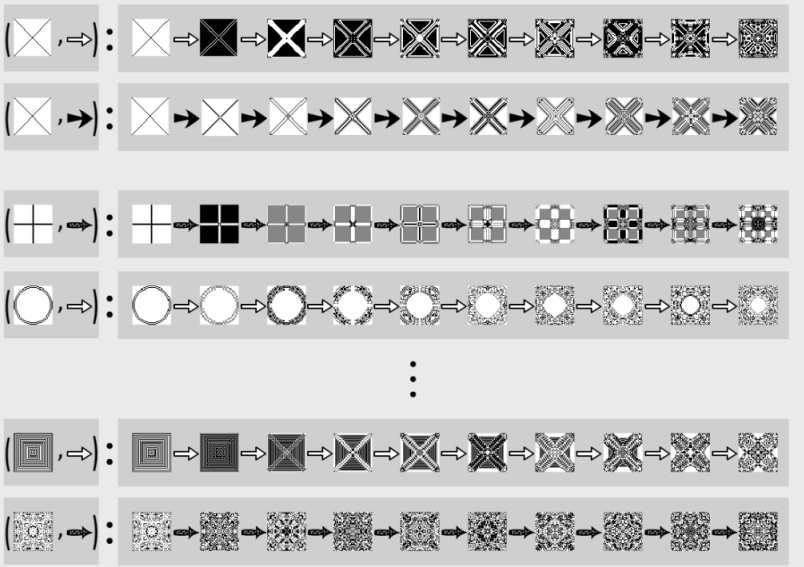
×

Prostor *pravidel*
celulárních automatů

=

Populace makro návrhů

dvojice (*makro semínko, pravidlo)*



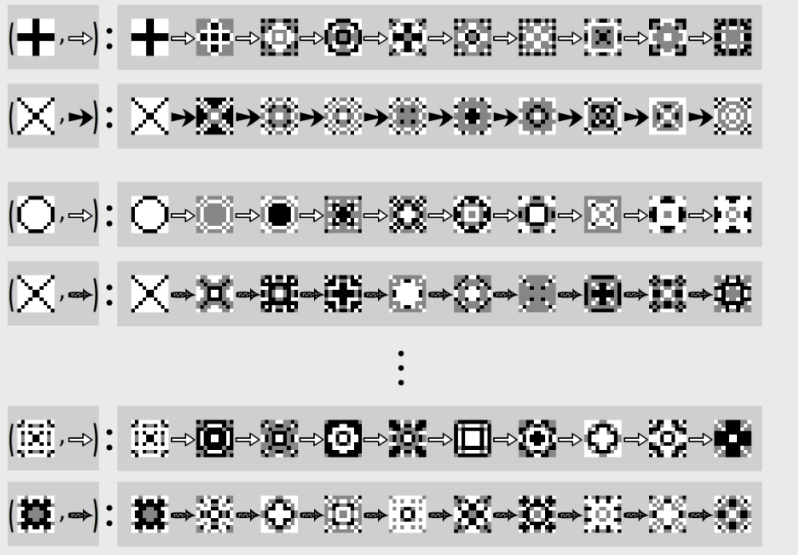
Mikro semínka
(počáteční *mikro návrhy)*

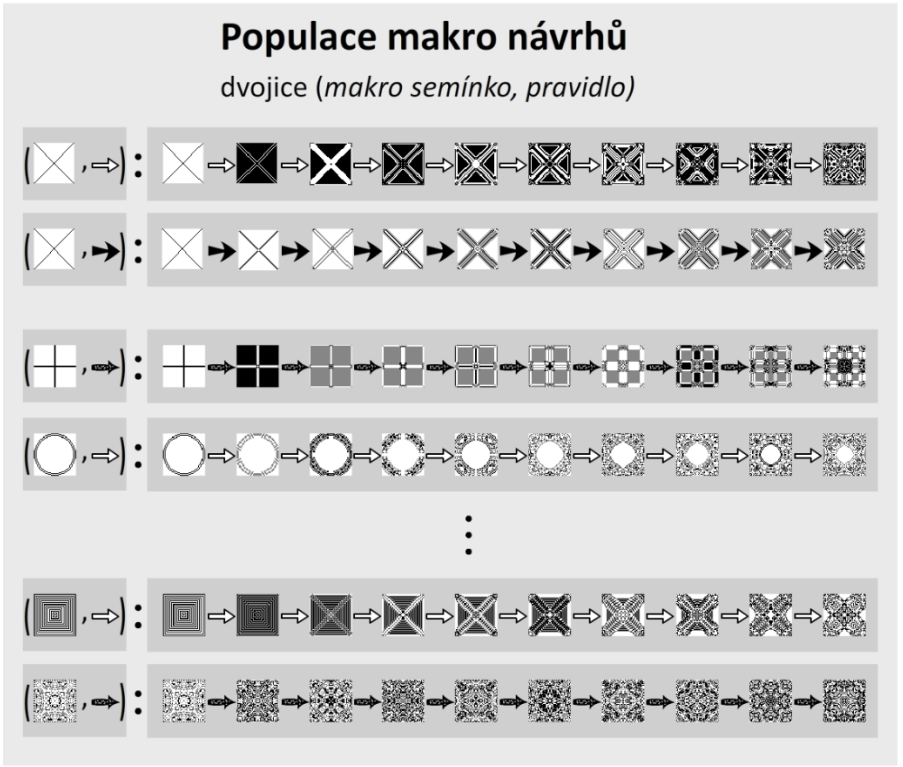
×

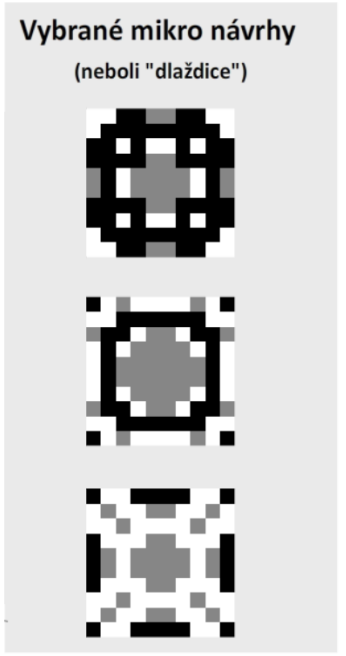
=

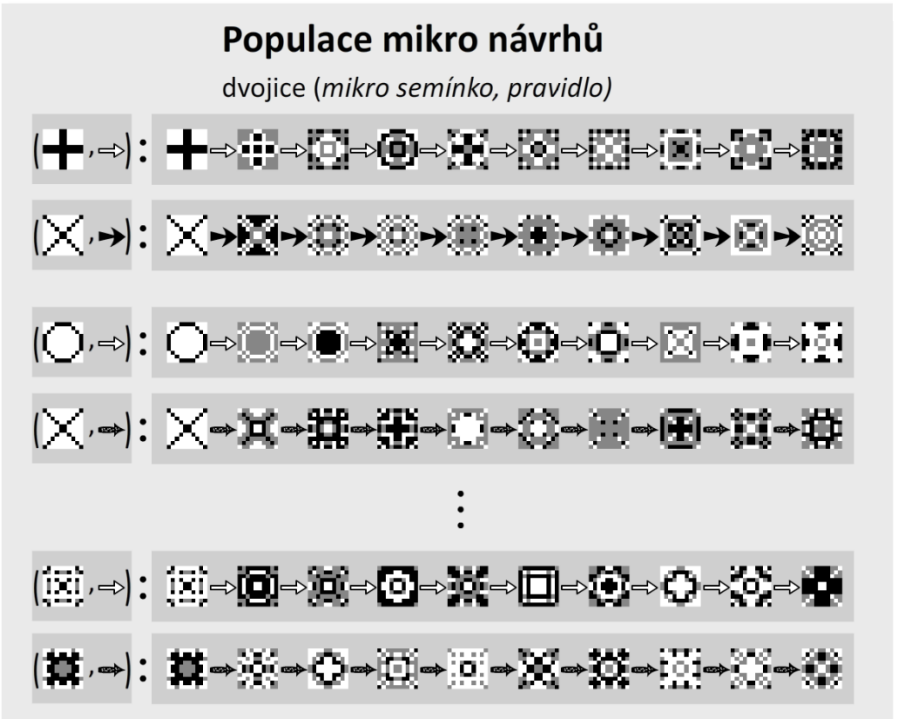
Populace mikro návrhů

dvojice (*mikro semínko, pravidlo)*




Populace makro návrhů
dvojice (*makro semínko, pravidlo)*


Vybrané mikro návrhy
(neboli "dlaždice")

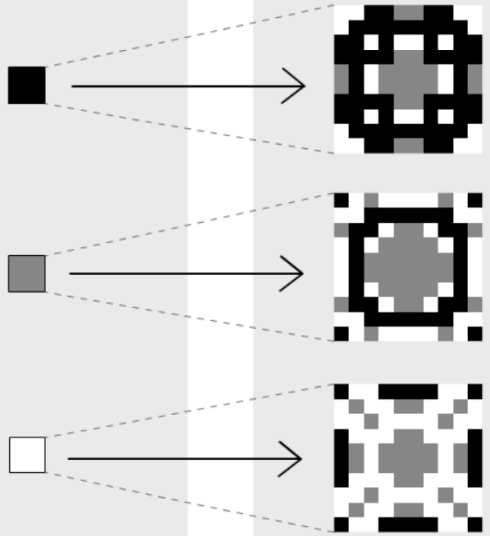

Populace mikro návrhů
dvojice (*mikro semínko, pravidlo)*


Vybraný makro návrh

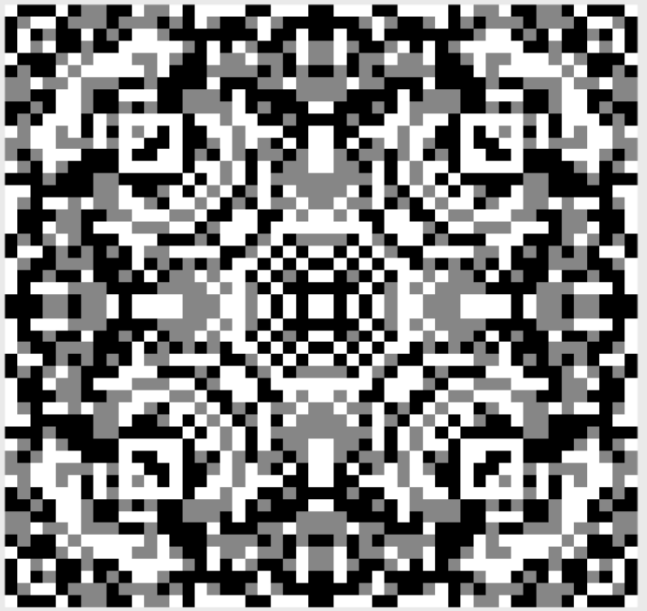
Na makro návrh aplikujeme dlaždice...

Pixely makro návrhu
(stavy jednotlivých buněk)

Vybrané mikro návrhy
(neboli "dlaždice")

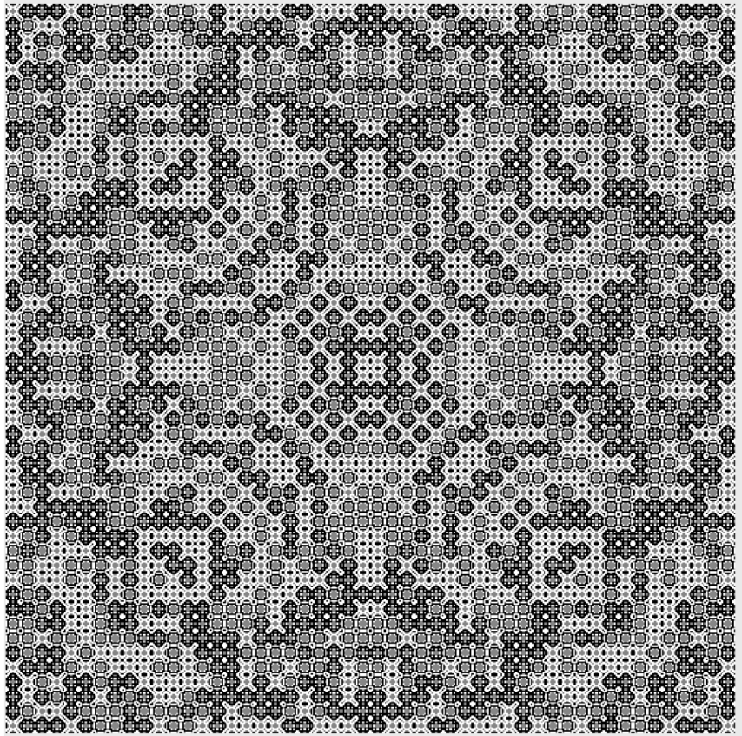


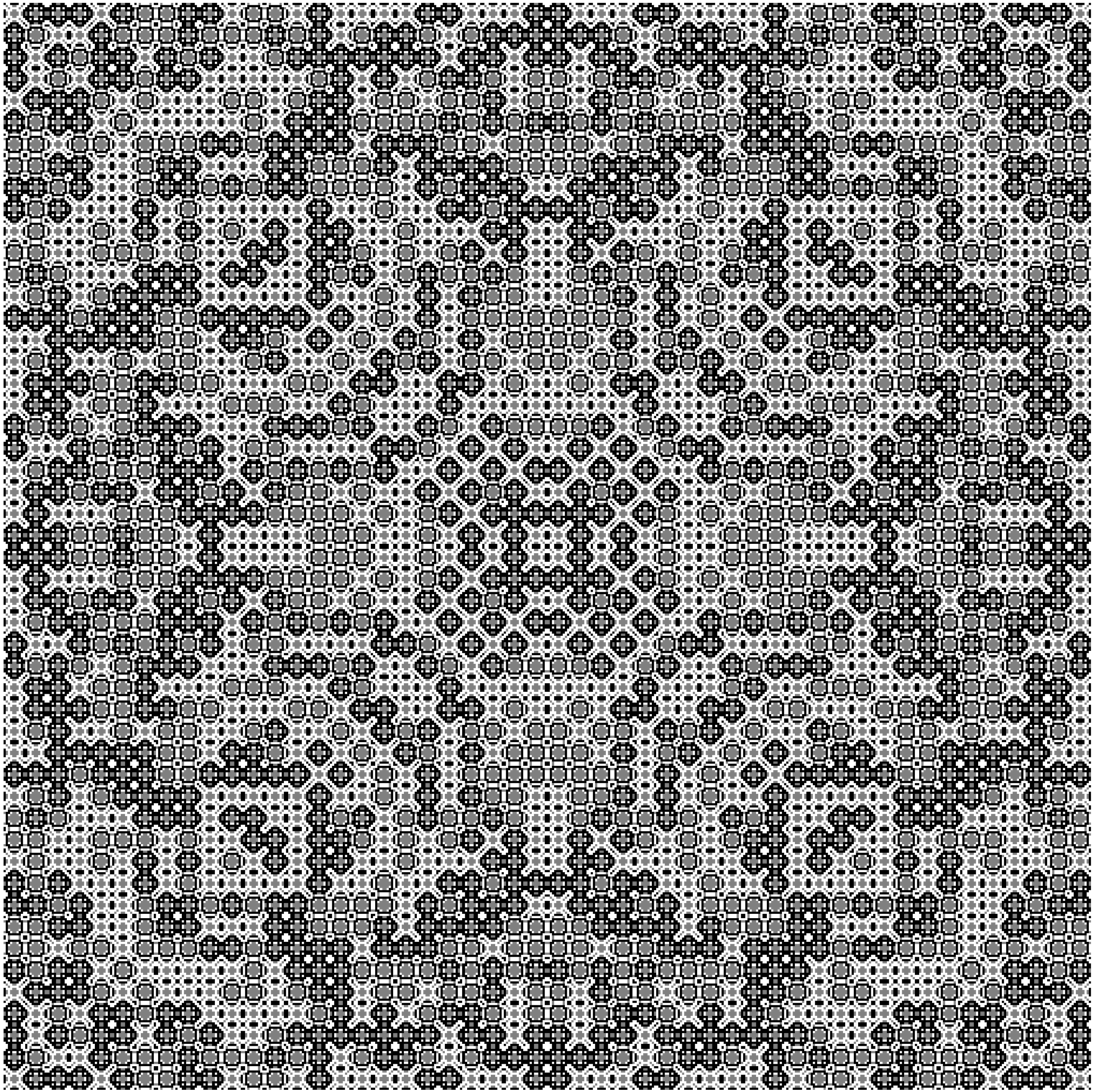
Vybraný makro návrh



Finální návrh

Vzniklý nahrazanením buněk v makro návrhu příslušnými mikro návrhy.





Substituce dlažek pomocí typů

Goal …"Img 500" = Img 2 * 2 * 5 * 5 * 5

```
(Img (T 2 (T 2 (T 5 (T 5 5)))))
```

```
replace : (Mul a b c) -> (Img a) -> (Img b) -> (Img c)

a_a1  :  Mul a 1 a
a_1a  :  Mul 1 a a
a_xab : (Mul a b c) -> (Mul (T x a) b (T x c))
a_axb : (Mul a b c) -> (Mul a (T x b) (T x c))
```

.
.
.

Goal = (Img (T 2 (T 2 (T 5 (T 5 5)))))

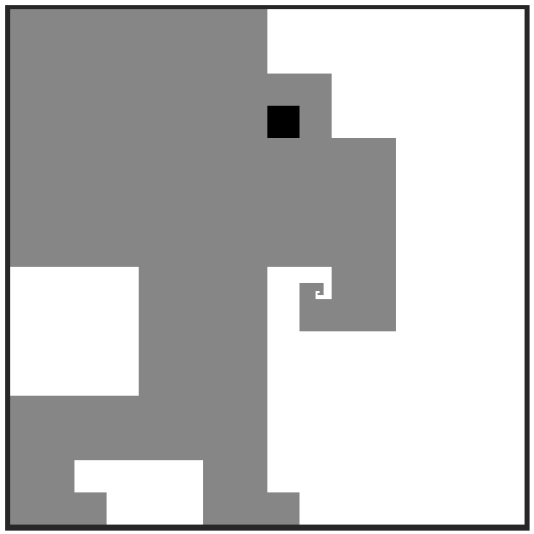
(1) 500 = img
(2) N/A
(3) N/A
(4) <500,1> = (replace a1 img img)
(5) <250,2> = (replace (axb a1) img img)
(6) <2,250> = (replace (axb (xab 1a)) img img)
(7) <5,100> = (replace (axb (axb (xab 1a))) img img)
(8) <50,10> = (replace (xab (axb (xab (axb a1)))) img img)
(9) <<500,1>,1> = (replace (xab a1) (replace (xab a1) img img) img)
(10) <50,<1,10>> = (replace (xab (axb (axb a1))) img (replace 1a img img))
(11) <<100,1>,5> = (replace (xab (xab (xab (xab 1a)))) (replace a1 img img)
(12) <<1,1>,<2,250>> = (replace (axb 1a) (replace a1 img img) (replace (xab
(13) <<<2,1>,125>,2> = (replace (axb (xab a1)) (replace (xab 1a) (replace a1
(14) <<<1,1>,<2,250>>,1> = (replace a1 (replace 1a (replace 1a img img) (repl
(15) <<2,50>,<1,5>> = (replace (xab (xab (axb a1))) (replace (axb (xab 1a))
(16) <2,<2,<1,125>>> = (replace (xab (axb (axb 1a))) img (replace (xab (axb
(17) <25,<<1,<1,1>>,20>> = (replace (axb (axb (xab (axb a1)))) img (replace
(18) <<4,<1,<1,1>>>,<125,1>> = (replace (xab (xab 1a)) (replace a1 img (repla
(19) <1,<1,<50,<5,2>>>> = (replace 1a img (replace (axb 1a) img (replace (xa
(20) <<1,<<1,1>,<2,1>>>,250> = (replace (xab 1a) (replace (axb a1) img (repla
(21) <<<<1,2>,250>,1>,<<1,1>,1>> = (replace a1 (replace a1 (replace (xab 1a)
(22) <1,<<10,<<1,1>,<50,1>>>,1>> = (replace 1a img (replace a1 (replace (xab
(23) <2,<5,<2,<1,<5,5>>>>> = (replace (axb (xab 1a)) img (replace (axb (xab 1a)
(24) <2,<1,<5,<<2,5>,5>>>> = (replace (axb (xab (axb 1a))) img (replace 1a img
(25) <<250,<1,<<2,<1,1>>,1>>>,1> = (replace (xab a1) (replace (axb (xab a1))
(26) <<<<5,25>,<<2,1>,2>>,1>,1> = (replace a1 (replace (xab a1) (replace (ax
(27) <<<1,1>,1>,<<1,<250,<2,1>>>,1>> = (replace 1a (replace 1a (replace a1 im
(28) <<1,<<<1,<<1,<1,1>>,1>>,1>,<1,1>>>,500> = (replace 1a (replace a1 img (r
(29) <1,<<1,<1,1>>,<<<1,1>,<1,2>>,250>>> = (replace 1a img (replace (axb 1a)
(30) <<<<1,<<1,1>,1>>,<25,2>>,1>,10> = (replace (axb (xab (axb a1))) (replace
(31) <<<<<1,5>,<25,1>>,1>,2>,1>,2> = (replace (axb (xab (xab a1))) (replace
(32) <<<2,1>,1>,<<<1,1>,5>,<<1,50>,1>>> = (replace (xab (axb (axb 1a))) (rep

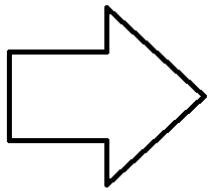
Work in Progress:

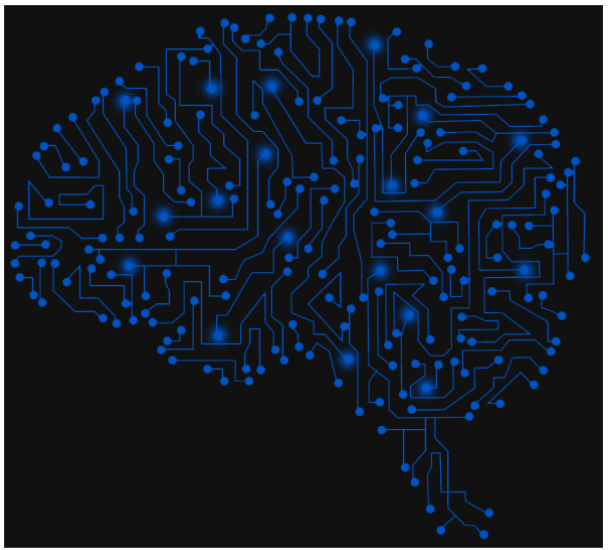
i2c : Images to Code

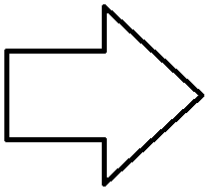
- **Cíl:** Natrénovat hlubokou síť, která převádí:
  - **Obrázky** na **Program popisující obrázek**
  - .. tak aby výsledný obrázek byl co nejpodobnější vstupu

- „Natrénovat síť, která vidí kód.“









```
q(G, q(q(W, W, q(G,
G, B, G), W), W, G,
W), q(W, G, q(G, G,
G, q(W, W, G, W)),
q(G, G, W, G)),
q(q(q(W, q(W, W, G,
q(G, W, q(q(W, W, W,
G), G, W, W)),
W, G), G, W, W), W,
q(W, W, q(W, W, G,
W), W), W))
```

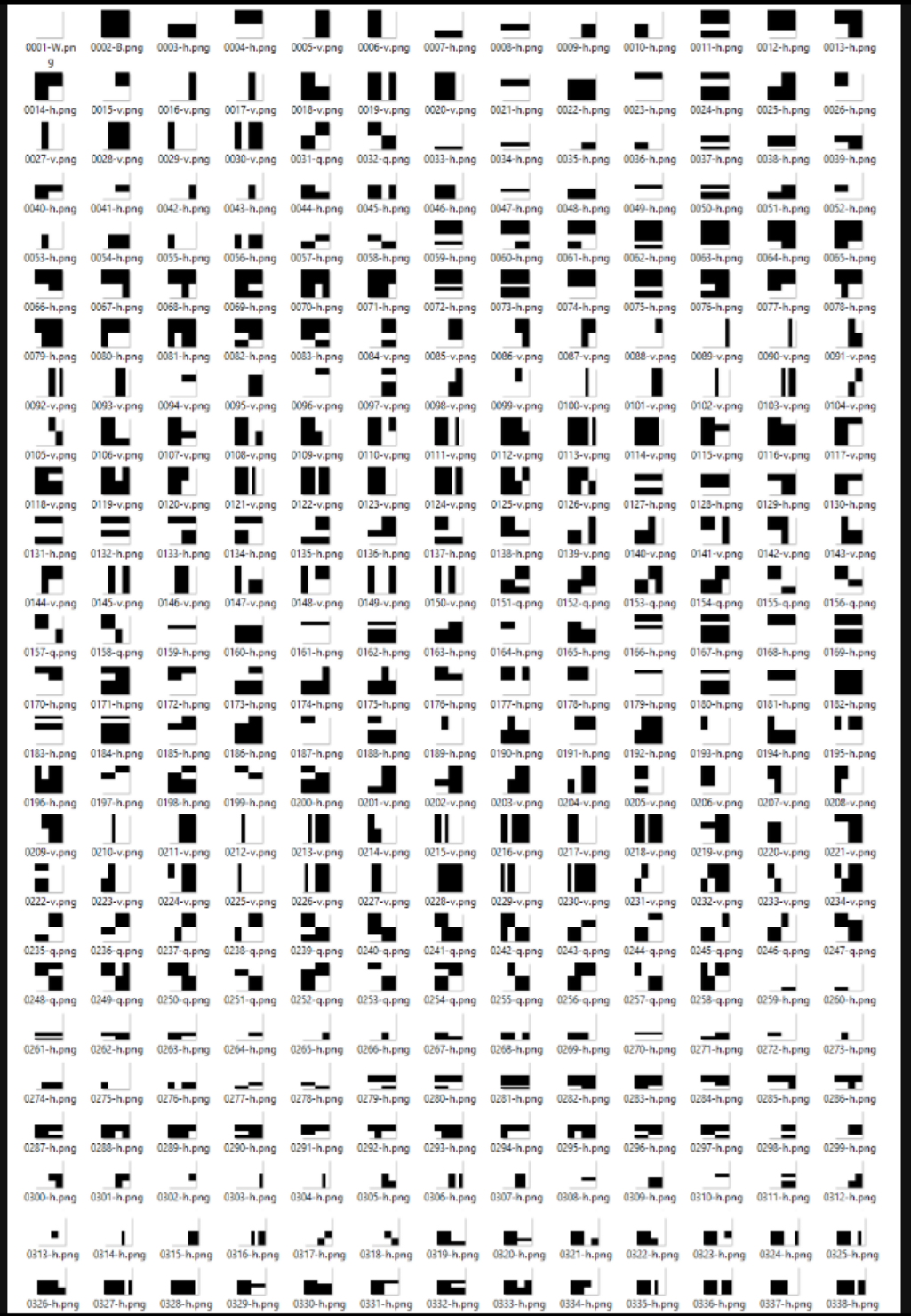
i2c : Jazyk pro popis obrázků

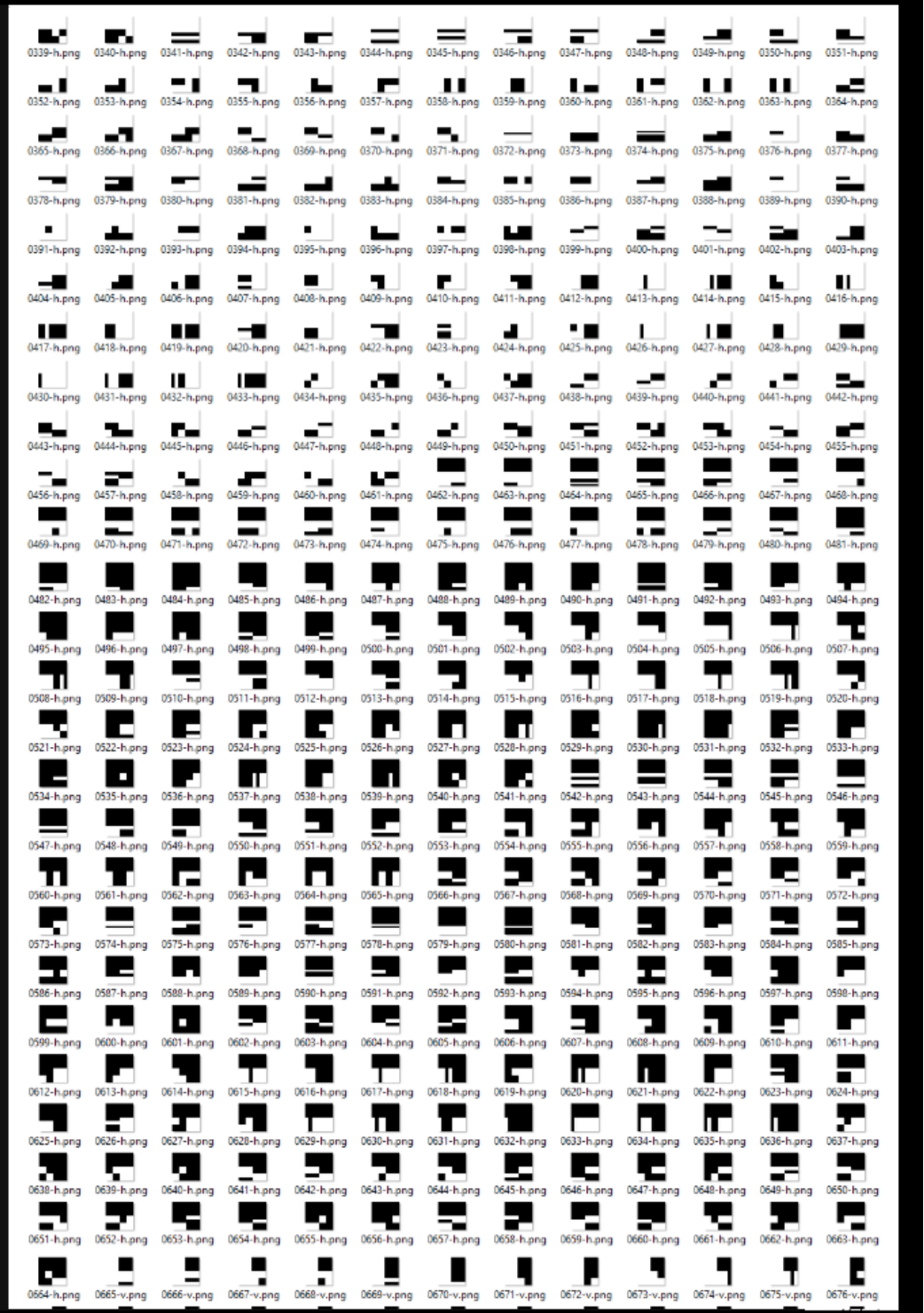
- Sada schopná popisovat hierarchické struktury
- Co nejjednodušší
- ⟶ Zvolené stavební symboly pro první iteraci experimentů:
  - **h** : horizontální split
  - **v** : vertikální split
  - **q** : split na 4 kvadranty
  - **B** : černá výplň
  - **W** : bílá výplň

i2c : Generování datasetu

- Postupně podle velikosti
  - Do určitého limitu exhaustivně
  - Nad limit uniformní samplování

- „Occamova břitva“
  - necháváme si jen nejmenší reprezentaci
  - *ImageHash* *https://github.com/JohannesBuchner/imagehash*


0001-W.png
0002-B.png
0003-h.png
0004-h.png
0005-v.png
0006-v.png
0007-h.png
0008-h.png
0009-h.png
0010-h.png
0011-h.png
0012-h.png
0013-h.png
0014-h.png
0015-v.png
0016-v.png
0017-v.png
0018-v.png
0019-v.png
0020-v.png
0021-h.png
0022-h.png
0023-h.png
0024-h.png
0025-h.png
0026-h.png
0027-v.png
0028-v.png
0029-v.png
0030-v.png
0031-q.png
0032-q.png
0033-h.png
0034-h.png
0035-h.png
0036-h.png
0037-h.png
0038-h.png
0039-h.png
0040-h.png
0041-h.png
0042-h.png
0043-h.png
0044-h.png
0045-h.png
0046-h.png
0047-h.png
0048-h.png
0049-h.png
0050-h.png
0051-h.png
0052-h.png
0053-h.png
0054-h.png
0055-h.png
0056-h.png
0057-h.png
0058-h.png
0059-h.png
0060-h.png
0061-h.png
0062-h.png
0063-h.png
0064-h.png
0065-h.png
0066-h.png
0067-h.png
0068-h.png
0069-h.png
0070-h.png
0071-h.png
0072-h.png
0073-h.png
0074-h.png
0075-h.png
0076-h.png
0077-h.png
0078-h.png
0079-h.png
0080-h.png
0081-h.png
0082-h.png
0083-h.png
0084-v.png
0085-v.png
0086-v.png
0087-v.png
0088-v.png
0089-v.png
0090-v.png
0091-v.png
0092-v.png
0093-v.png
0094-v.png
0095-v.png
0096-v.png
0097-v.png
0098-v.png
0099-v.png
0100-v.png
0101-v.png
0102-v.png
0103-v.png
0104-v.png
0105-v.png
0106-v.png
0107-v.png
0108-v.png
0109-v.png
0110-v.png
0111-v.png
0112-v.png
0113-v.png
0114-v.png
0115-v.png
0116-v.png
0117-v.png
0118-v.png
0119-v.png
0120-v.png
0121-v.png
0122-v.png
0123-v.png
0124-v.png
0125-v.png
0126-v.png
0127-h.png
0128-h.png
0129-h.png
0130-h.png
0131-h.png
0132-h.png
0133-h.png
0134-h.png
0135-h.png
0136-h.png
0137-h.png
0138-h.png
0139-v.png
0140-v.png
0141-v.png
0142-v.png
0143-v.png
0144-v.png
0145-v.png
0146-v.png
0147-v.png
0148-v.png
0149-v.png
0150-v.png
0151-q.png
0152-q.png
0153-q.png
0154-q.png
0155-q.png
0156-q.png
0157-q.png
0158-q.png
0159-h.png
0160-h.png
0161-h.png
0162-h.png
0163-h.png
0164-h.png
0165-h.png
0166-h.png
0167-h.png
0168-h.png
0169-h.png
0170-h.png
0171-h.png
0172-h.png
0173-h.png
0174-h.png
0175-h.png
0176-h.png
0177-h.png
0178-h.png
0179-h.png
0180-h.png
0181-h.png
0182-h.png
0183-h.png
0184-h.png
0185-h.png
0186-h.png
0187-h.png
0188-h.png
0189-h.png
0190-h.png
0191-h.png
0192-h.png
0193-h.png
0194-h.png
0195-h.png
0196-h.png
0197-h.png
0198-h.png
0199-h.png
0200-h.png
0201-v.png
0202-v.png
0203-v.png
0204-v.png
0205-v.png
0206-v.png
0207-v.png
0208-v.png
0209-v.png
0210-v.png
0211-v.png
0212-v.png
0213-v.png
0214-v.png
0215-v.png
0216-v.png
0217-v.png
0218-v.png
0219-v.png
0220-v.png
0221-v.png
0222-v.png
0223-v.png
0224-v.png
0225-v.png
0226-v.png
0227-v.png
0228-v.png
0229-v.png
0230-v.png
0231-v.png
0232-v.png
0233-v.png
0234-v.png
0235-q.png
0236-q.png
0237-q.png
0238-q.png
0239-q.png
0240-q.png
0241-q.png
0242-q.png
0243-q.png
0244-q.png
0245-q.png
0246-q.png
0247-q.png
0248-q.png
0249-q.png
0250-q.png
0251-q.png
0252-q.png
0253-q.png
0254-q.png
0255-q.png
0256-q.png
0257-q.png
0258-q.png
0259-h.png
0260-h.png
0261-h.png
0262-h.png
0263-h.png
0264-h.png
0265-h.png
0266-h.png
0267-h.png
0268-h.png
0269-h.png
0270-h.png
0271-h.png
0272-h.png
0273-h.png
0274-h.png
0275-h.png
0276-h.png
0277-h.png
0278-h.png
0279-h.png
0280-h.png
0281-h.png
0282-h.png
0283-h.png
0284-h.png
0285-h.png
0286-h.png
0287-h.png
0288-h.png
0289-h.png
0290-h.png
0291-h.png
0292-h.png
0293-h.png
0294-h.png
0295-h.png
0296-h.png
0297-h.png
0298-h.png
0299-h.png
0300-h.png
0301-h.png
0302-h.png
0303-h.png
0304-h.png
0305-h.png
0306-h.png
0307-h.png
0308-h.png
0309-h.png
0310-h.png
0311-h.png
0312-h.png
0313-h.png
0314-h.png
0315-h.png
0316-h.png
0317-h.png
0318-h.png
0319-h.png
0320-h.png
0321-h.png
0322-h.png
0323-h.png
0324-h.png
0325-h.png
0326-h.png
0327-h.png
0328-h.png
0329-h.png
0330-h.png
0331-h.png
0332-h.png
0333-h.png
0334-h.png
0335-h.png
0336-h.png
0337-h.png
0338-h.png


0339-h.png
0340-h.png
0341-h.png
0342-h.png
0343-h.png
0344-h.png
0345-h.png
0346-h.png
0347-h.png
0348-h.png
0349-h.png
0350-h.png
0351-h.png
0352-h.png
0353-h.png
0354-h.png
0355-h.png
0356-h.png
0357-h.png
0358-h.png
0359-h.png
0360-h.png
0361-h.png
0362-h.png
0363-h.png
0364-h.png
0365-h.png
0366-h.png
0367-h.png
0368-h.png
0369-h.png
0370-h.png
0371-h.png
0372-h.png
0373-h.png
0374-h.png
0375-h.png
0376-h.png
0377-h.png
0378-h.png
0379-h.png
0380-h.png
0381-h.png
0382-h.png
0383-h.png
0384-h.png
0385-h.png
0386-h.png
0387-h.png
0388-h.png
0389-h.png
0390-h.png
0391-h.png
0392-h.png
0393-h.png
0394-h.png
0395-h.png
0396-h.png
0397-h.png
0398-h.png
0399-h.png
0400-h.png
0401-h.png
0402-h.png
0403-h.png
0404-h.png
0405-h.png
0406-h.png
0407-h.png
0408-h.png
0409-h.png
0410-h.png
0411-h.png
0412-h.png
0413-h.png
0414-h.png
0415-h.png
0416-h.png
0417-h.png
0418-h.png
0419-h.png
0420-h.png
0421-h.png
0422-h.png
0423-h.png
0424-h.png
0425-h.png
0426-h.png
0427-h.png
0428-h.png
0429-h.png
0430-h.png
0431-h.png
0432-h.png
0433-h.png
0434-h.png
0435-h.png
0436-h.png
0437-h.png
0438-h.png
0439-h.png
0440-h.png
0441-h.png
0442-h.png
0443-h.png
0444-h.png
0445-h.png
0446-h.png
0447-h.png
0448-h.png
0449-h.png
0450-h.png
0451-h.png
0452-h.png
0453-h.png
0454-h.png
0455-h.png
0456-h.png
0457-h.png
0458-h.png
0459-h.png
0460-h.png
0461-h.png
0462-h.png
0463-h.png
0464-h.png
0465-h.png
0466-h.png
0467-h.png
0468-h.png
0469-h.png
0470-h.png
0471-h.png
0472-h.png
0473-h.png
0474-h.png
0475-h.png
0476-h.png
0477-h.png
0478-h.png
0479-h.png
0480-h.png
0481-h.png
0482-h.png
0483-h.png
0484-h.png
0485-h.png
0486-h.png
0487-h.png
0488-h.png
0489-h.png
0490-h.png
0491-h.png
0492-h.png
0493-h.png
0494-h.png
0495-h.png
0496-h.png
0497-h.png
0498-h.png
0499-h.png
0500-h.png
0501-h.png
0502-h.png
0503-h.png
0504-h.png
0505-h.png
0506-h.png
0507-h.png
0508-h.png
0509-h.png
0510-h.png
0511-h.png
0512-h.png
0513-h.png
0514-h.png
0515-h.png
0516-h.png
0517-h.png
0518-h.png
0519-h.png
0520-h.png
0521-h.png
0522-h.png
0523-h.png
0524-h.png
0525-h.png
0526-h.png
0527-h.png
0528-h.png
0529-h.png
0530-h.png
0531-h.png
0532-h.png
0533-h.png
0534-h.png
0535-h.png
0536-h.png
0537-h.png
0538-h.png
0539-h.png
0540-h.png
0541-h.png
0542-h.png
0543-h.png
0544-h.png
0545-h.png
0546-h.png
0547-h.png
0548-h.png
0549-h.png
0550-h.png
0551-h.png
0552-h.png
0553-h.png
0554-h.png
0555-h.png
0556-h.png
0557-h.png
0558-h.png
0559-h.png
0560-h.png
0561-h.png
0562-h.png
0563-h.png
0564-h.png
0565-h.png
0566-h.png
0567-h.png
0568-h.png
0569-h.png
0570-h.png
0571-h.png
0572-h.png
0573-h.png
0574-h.png
0575-h.png
0576-h.png
0577-h.png
0578-h.png
0579-h.png
0580-h.png
0581-h.png
0582-h.png
0583-h.png
0584-h.png
0585-h.png
0586-h.png
0587-h.png
0588-h.png
0589-h.png
0590-h.png
0591-h.png
0592-h.png
0593-h.png
0594-h.png
0595-h.png
0596-h.png
0597-h.png
0598-h.png
0599-h.png
0600-h.png
0601-h.png
0602-h.png
0603-h.png
0604-h.png
0605-h.png
0606-h.png
0607-h.png
0608-h.png
0609-h.png
0610-h.png
0611-h.png
0612-h.png
0613-h.png
0614-h.png
0615-h.png
0616-h.png
0617-h.png
0618-h.png
0619-h.png
0620-h.png
0621-h.png
0622-h.png
0623-h.png
0624-h.png
0625-h.png
0626-h.png
0627-h.png
0628-h.png
0629-h.png
0630-h.png
0631-h.png
0632-h.png
0633-h.png
0634-h.png
0635-h.png
0636-h.png
0637-h.png
0638-h.png
0639-h.png
0640-h.png
0641-h.png
0642-h.png
0643-h.png
0644-h.png
0645-h.png
0646-h.png
0647-h.png
0648-h.png
0649-h.png
0650-h.png
0651-h.png
0652-h.png
0653-h.png
0654-h.png
0655-h.png
0656-h.png
0657-h.png
0658-h.png
0659-h.png
0660-h.png
0661-h.png
0662-h.png
0663-h.png
0664-h.png
0665-v.png
0666-v.png
0667-v.png
0668-v.png
0669-v.png
0670-v.png
0671-v.png
0672-v.png
0673-v.png
0674-v.png
0675-v.png
0676-v.png

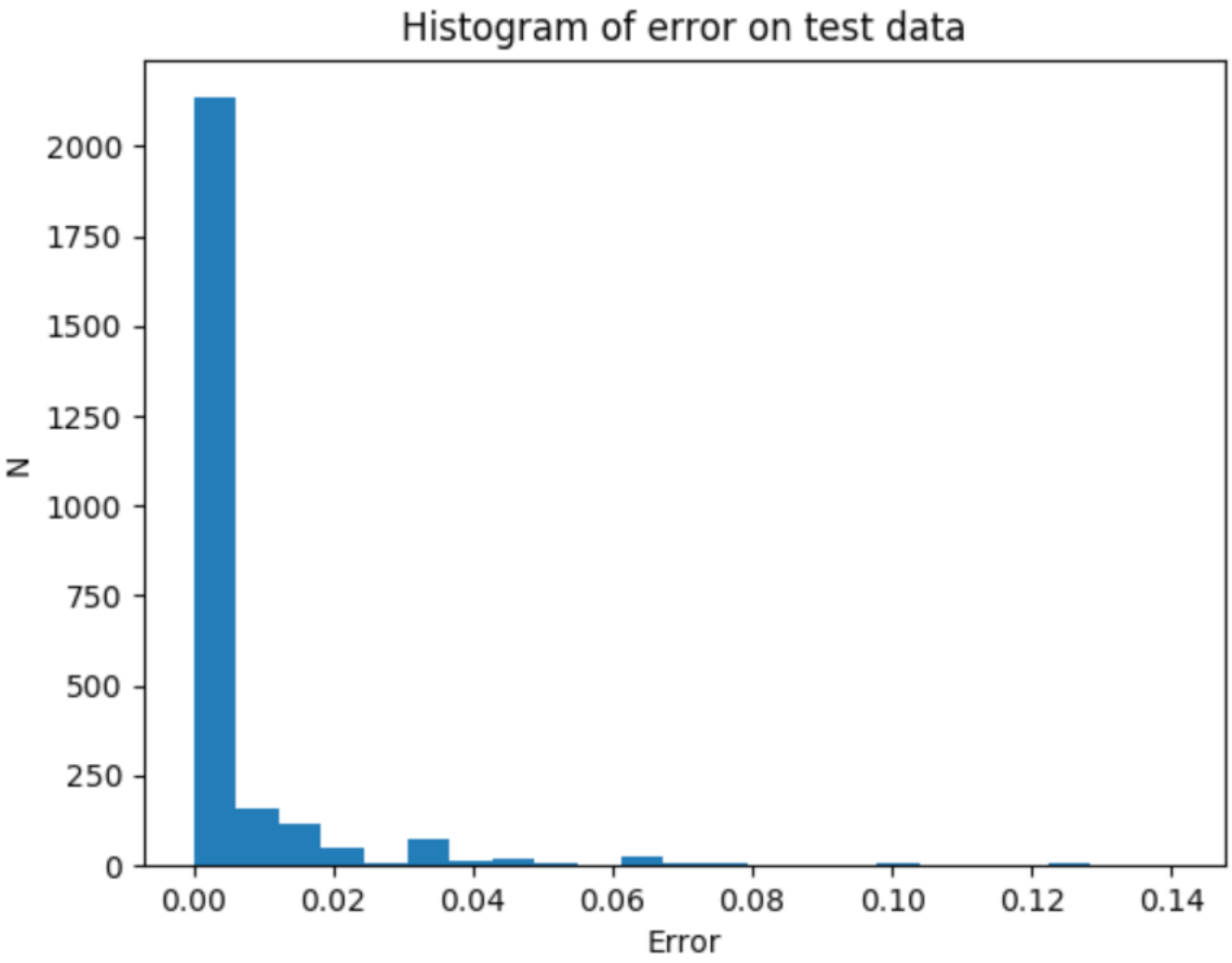
i2c : Model

- Rekurentní síť
  - Konvoluční enkodér
  - Rekurentní dekodér
    - Gated Recurrent Unit (GRU)
    - Výstup: k´d v prefixové notaci
- Trénovací metrika: BLEU-4
  - Tzn trénujeme jednoduše na „textu“ kódu
- Používám toolkit NeuralMonkey kluků z UFALu
  - https://github.com/ufal/neuralmonkey

i2c: Výsledky (32×32)

Stats for test data

Number of test instances: **2642**
Number of absolute matches: **2010**
Percent absolute matches: **76.079** %
Average error: **0.005**
Worst error: **0.141**
Number of output codes in incorrect format: **0**

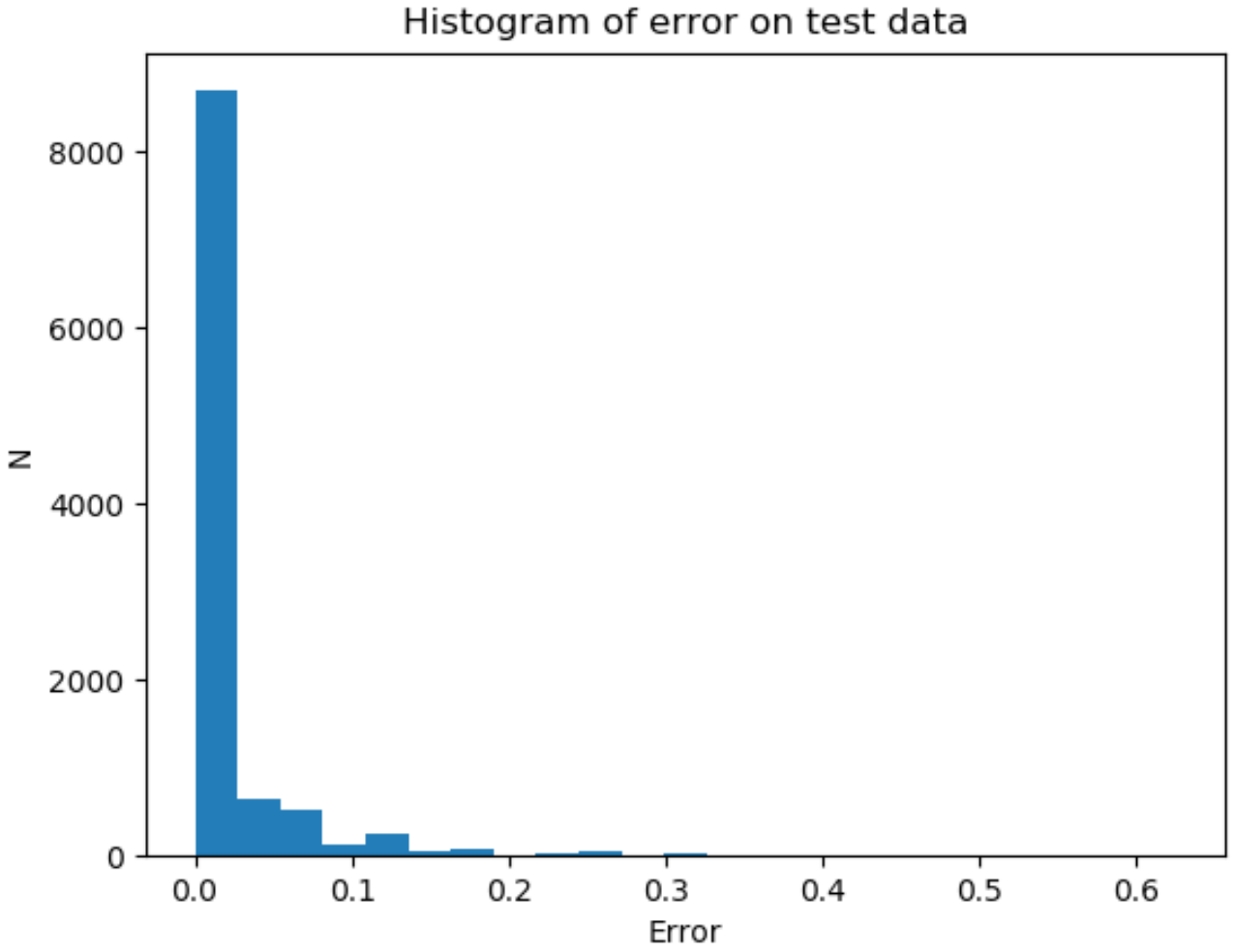


i2c: Výsledky (64×64)

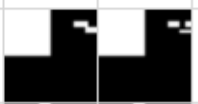
Results on test data

Stats for test data

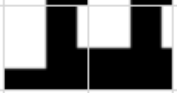
Number of test instances: **10505**
Number of absolute matches: **8478**
Percent absolute matches: **80.704** %
Average error: **0.015**
Worst error: **0.625**
Number of output codes in incorrect format: **29**



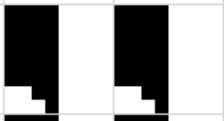
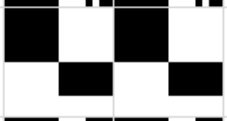
32×32
Náhodné
příklady

file	in	out	raw output / input prefix	error
00065313.png			v v W h W v v W h v W B B B W h v W W v v W v q W B h h h h B W W W B B B W	0.015625
00051089.png			v B h q v B h W B B B W B q B q q B W h B B v B h B B B B W B B	0
00010961.png			v B h q W h B W B W W v B h q W h B W B W W	0
00011386.png			v B q B W v W v W B B v B q B W v W v W B B	0
00006861.png			h B v h h B W h W B B h B v h h B W h W B B	0
00035148.png			h h B h B h B h B v W B W h h B h B h B h q B W B W B W	0.03125
00019001.png			h h v v W B W h B W W h h v v W B W h B W W	0
00038128.png			h B h W h B q v B W W W B h B h W h h B B q v B W W W h B B	0
00015228.png			h h h B W v W B h B W h h h B W v W B h B W	0
00017012.png			q v B v B W B B h B W q v B v B W B B h B W	0
00031831.png			v B v B h h v h B W W W B q B v B q h B W W W h W W B B	0
00002717.png			h W h W h v W B h W B h W h W h v W B h W B	0
00036468.png			h v q B B W v B h B W v B W W q h B q W B W v B v W W v B W W W	0
00073335.png			h v W v B h h B q W v B W B h B W B B h v W h q B B v h B B B h v W v v W B B v B W B B	0.009765625
00074295.png			q q B q h B W W W B W B B W B q q B q h B W W W h B h B v B q B B B W W B B W B	0
00018614.png			h h v W B h h B W B B h h v W B h h B W B B	0

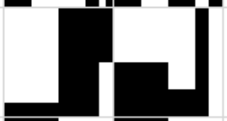
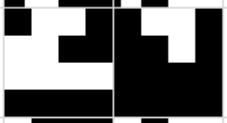
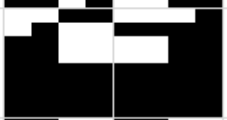
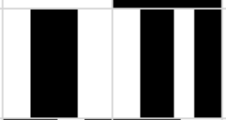
32×32
Nejhorší

file	in	out	raw output / input prefix	error
00066843.png			v B v W v v q B W B W B h W B v v B B v W v v h v q v B B B B B W B h W W B	0.140625
00045041.png			v q h W B B B W v v v W B W W v v h h W B B h B W v v h W W v B W W	0.125
00035537.png			h q h v B W B W W B W h q q B W h W W W h B W W B W	0.125
00047024.png			v q B v B W h B W q W B W W B v q B v B W h B W q W B W W h h W B B	0.125
00026414.png			q W v B v B W B B q W v B v B W h W B B	0.125
00037066.png			h W h v h W h W v h B h W B W B B q W W h h W h W q B W W W B h W B	0.125
00072860.png			v q B W q W W v B v W B B v W B W v q v B B W v h W q B v B W W W h W B v B v W B W	0.11328125
00079253.png			h q v B h W B B W h B W h B h W q B W v B W B h h v h v B W h B B B W h B q W W q W h B B B W B	0.109375
00043935.png			h h B h q B h W B v B W h W B W W h h B q v h h B B W B h h W B W B W W	0.109375
00068310.png			v B h q h q W v W q B W W B W B W B W B B v B h q h v W h h v B B W v v W B W B W W B B	0.103515625
00072402.png			q h W B v v h B h W B W v W h B B W B h v h W B q v B W v B W v v h W B W W v B B v W B	0.1015625
00074992.png			v v h h q W B W h W v B W W W v W v W B B v h v q W q h W W B h W B W W W v v B W W W h B B	0.1015625
00080455.png			h h W h q B h W B B h W B q v h W B B W B W W h h W h q W h v B W B B W h h v q B B W B W B W W	0.1015625
00074721.png			v v v W h h B W h W v h W v h W B W W W B v q W h B W v h W v W v W q h W W W B W W W v B B	0.099609375
00054394.png			v q h h B W B v B W W B W v q h h B v v W W W W v B h v B B W W B W	0.09375
00065947.png			v B v v B v W h h B W h h B W W B v B v v B h v W h B v B W W v h W h B v W B B	0.09375
00024515.png			v v v h W B h B W W B v v v h W h B W B W B	0.09375
00040370.png			h B h W h h B B h B h v B W B h B h W h h W B q B B q B W W B B	0.09375

64×64
Náhodné příklady

file	in	out	raw output / original input	error
00051705.png			output : h W v v B v B W h W v B W original : q W W v B v B W v h W B W	0
00037973.png			output : v W q h W q B W W B W W W original : v W h h W v q B W W B W W	0
00042623.png			output : v h B h B v W q B B W B W original : q B W h B v W h B v W B W	0
00019608.png			output : h h v B h W v W B W B original : h h v B h W v W B W B	0
00005215.png			output : h B h B h W v B h W B original : h B h B h W v B h W B	0
00024679.png			output : v v q h W B B B W W W original : v v q h W B B B W W W	0
00042708.png			output : v B h h W h h B W h B W B original : q B h W h h B W h B W B B	0
00005392.png			output : h B h B v B W original : h B h B v v B v B W W	0.03125
00034496.png			output : v B h B v W v h W v B W B original : q B B B q W W W v v B W B	0.03125
00013229.png			output : q B W W h B h h B W W original : q B W W h B h h B W W	0
00010821.png			output : v B h h h v W B B B W original : v B h h h v W B B B W	0
00027914.png			output : q h h B h W B W B B W original : q h h B h W B W B B W	0
00028328.png			output : v q q W W v B W W W B B B original : q v v v W h W B W W B B B	0.03125
00041319.png			output : h h W v B h h B W B v B W original : h h W v B h h B W B v B W	0

64×64
Nejhorší

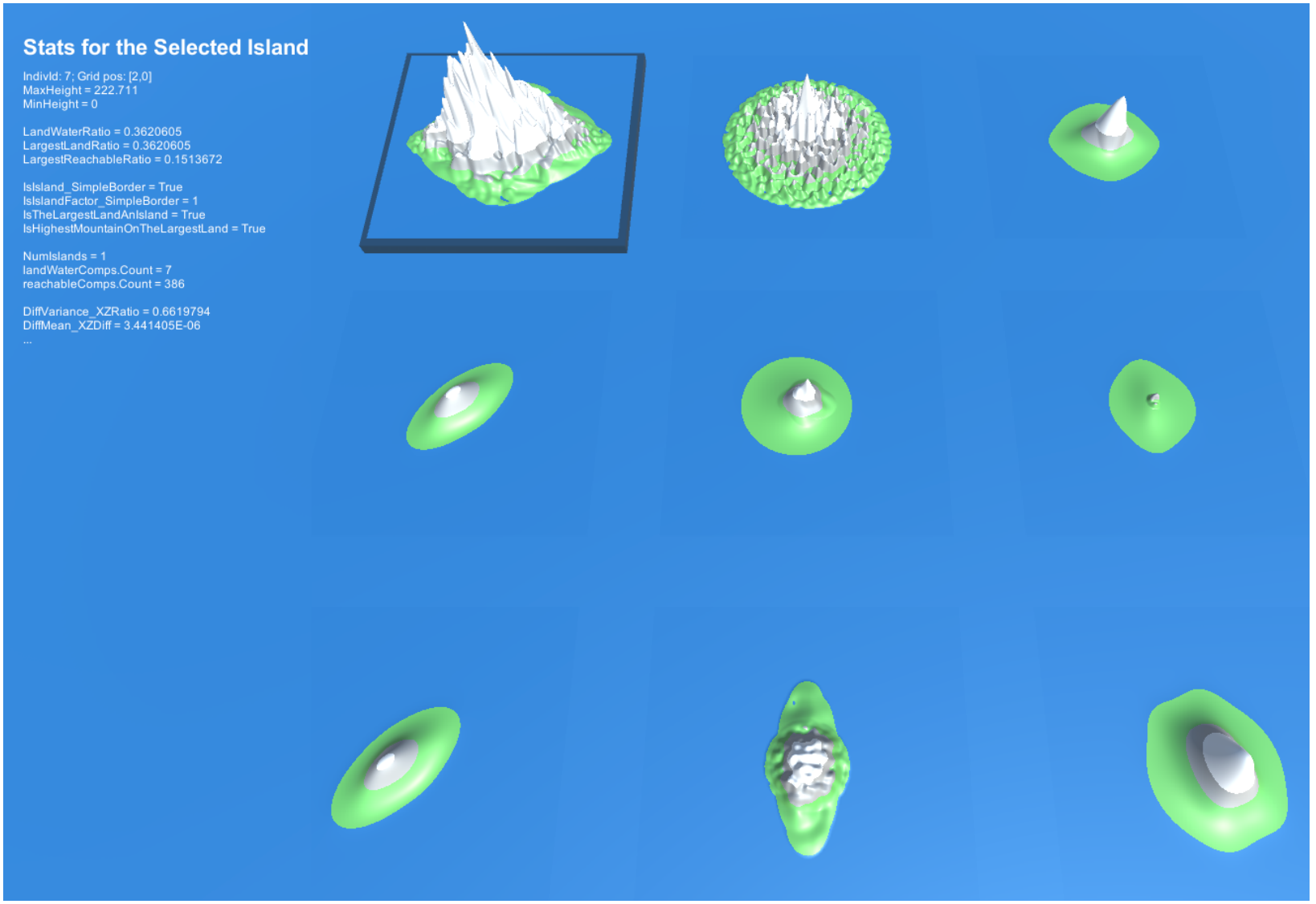
file	in	out	raw output / original input	error
00034118.png			output : v v h h v W B B B W v B original : v h v q W B B B W B v W B	0.625
00035879.png			output : q q q B B W B W W B W B original : q q q B B W B W W B W W B	0.5
00048766.png			output : v W v v v v B v W B W B original : v W v v v v B v B W B W B	0.46875
00044452.png			output : v h W v B W v B v W B original : v h W v B W v W v B v W B	0.4375
00039738.png			output : v h W B v h W h W B v B W original : q W B h W h W B v B v B W	0.4375
00034790.png			output : q B W v v h B v B W W B original : q B W h W v v B v B W W B	0.4375
00043187.png			output : h q v B W v W B B v W B B original : h q v B W v W B W B h W B	0.4375
00047423.png			output : q h q W B h W B W B W W B original : v h h q W B h W B W W W B	0.375
00038183.png			output : v W v B v W v v B W B original : v W v v W B v B v v W B W	0.375
00047075.png			output : v v B h W B q W B B v W B original : v v B h W h W B q B B W B	0.375
00052093.png			output : h h W v B v B W v W B original : h h W v B v B W v B v W B	0.375
00035360.png			output : h q h W B q W B B B W B B original : h q v W h W B h B W B W B	0.375
00050400.png			output : h v B W h h W h B W h W B original : h v B W h h h B W W h B W	0.375
00012691.png			output : v v W B v v v B W W B original : v v W B v v B v B W W	0.375

i2c : Future work

- Podstatně rozšířit sadu symbolů
  - Symetrie
  - Fraktály
  - Shadery
- Rekurzivní použití modelu pro malé obrázky na velké obrázky

- Ostrůvky...

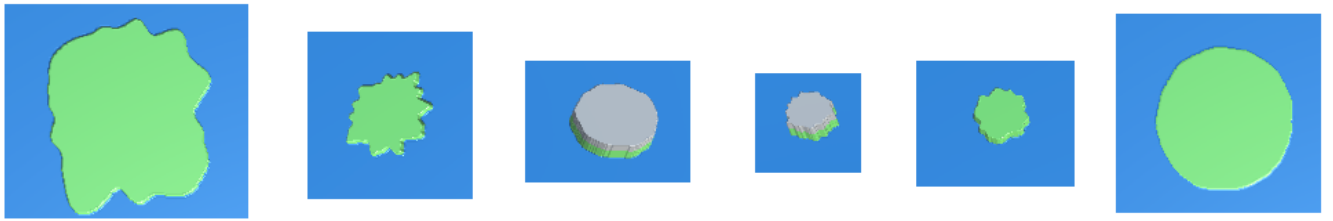
Work in Progress:
Interaktivní evoluce terénů pro ostrovy



((+ ((+ ((((((gauss 8) 2) 4) 32) 32) 0)) ((((((gauss 8) -1) -2) 16) 16) 2.356194))) ((+ ((+ ((+
((((((gauss 32) -1) -1) 32) 16) 3.141593)) ((+ ((((((gauss 8) -1) -2) 16) 32) 3.141593))
(((((((((pcone 1.25) 52) 82) 2.5) 50) -4) 555.9) 784.8) 0.03)))) ((((((gauss 8) -4) -1) 32) 32) 5.497787))) ((((((gauss 4) 4) 1) 16) 32) 3.926991)))

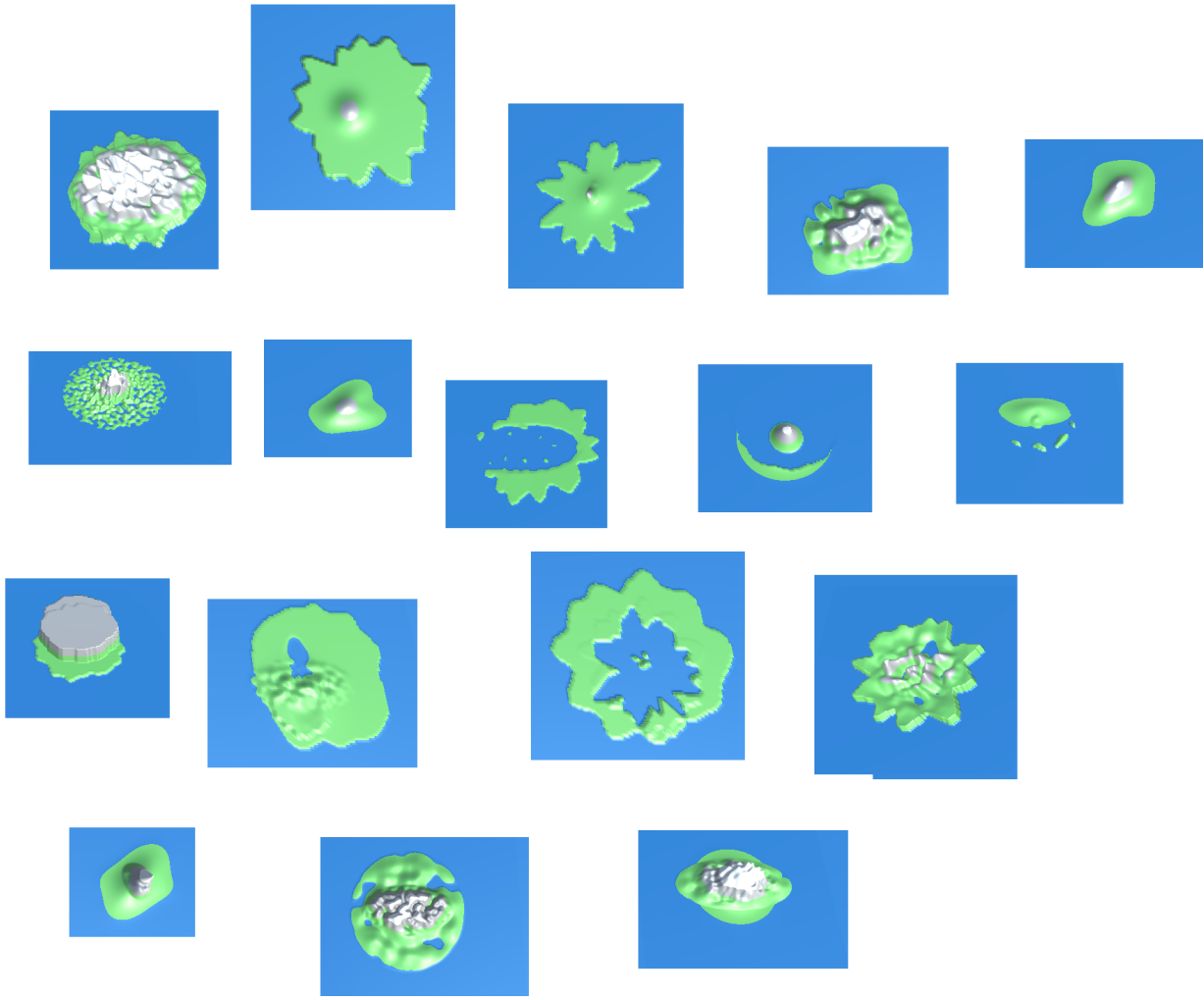
Strategie: Krok 1a

- Identifikovat zajimavy **mustry na ostrovy** (**listy**)
  - GaussHill
  - PerlinCone
  - PerlinCylinder



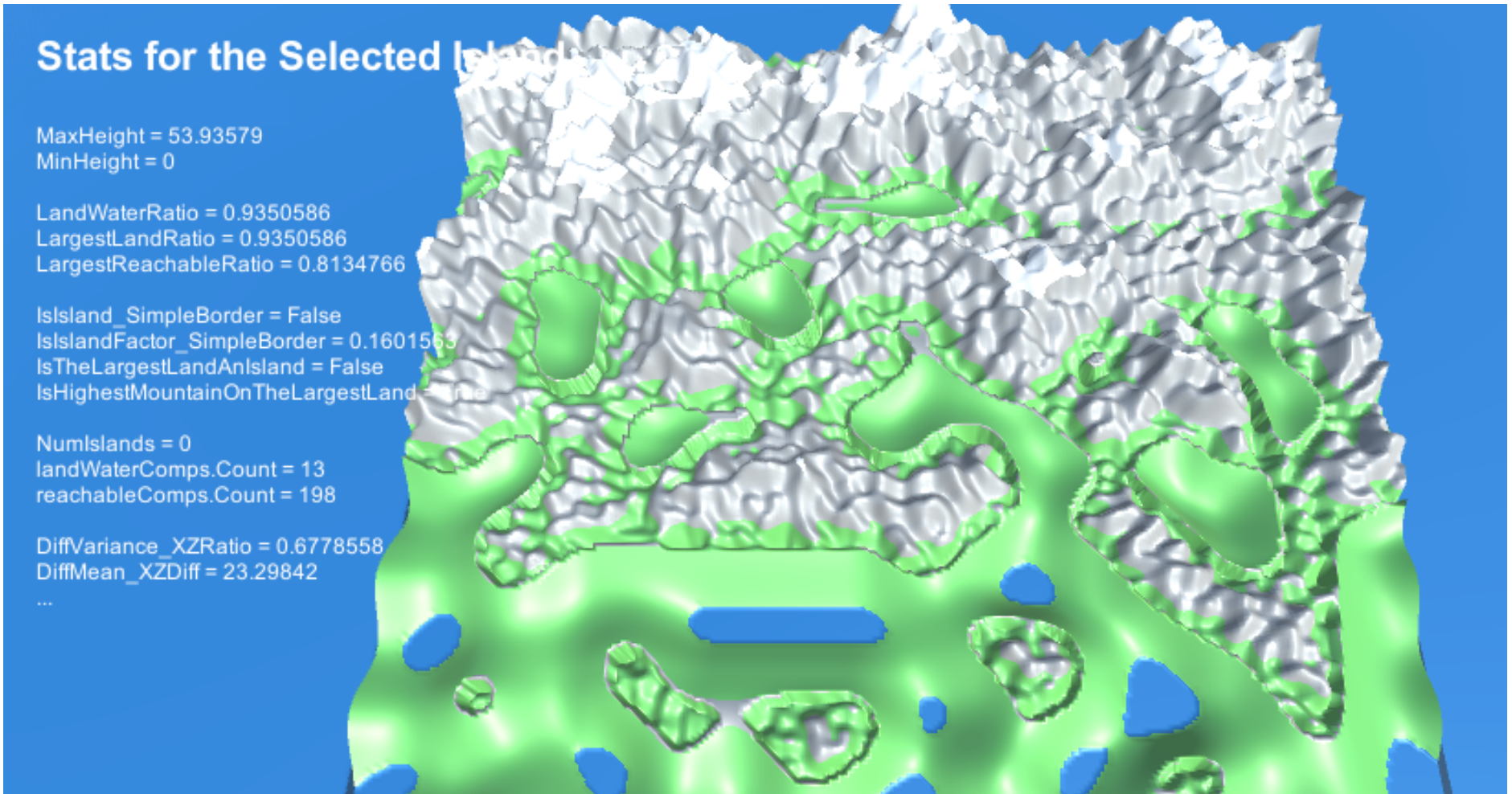
Strategie: Krok 1b

- Najít vhodné „kombinátory“ ostrovů (**binární operace**)
  - add
  - sub
  - mean
  - sigmoid step
  - lin interpolate
  - …
- Filtry (**unární operace**)
  - Eroze
  - ...



Strategie: Krok 2a

- Najít měřitelné veličiny, které by mohly šlechtění zautomatizovat



Strategie: Krok 2b

- Z veličin udělat „jazyk“ a zkusit 2-level evoluci

- Formy evoluce (evaluace):
  - seamless (měření chování uživatelů)
  - přiznaná (hvězdičky, porovnání)

Kdyby to někoho zajímalo podrobněji...

Preprinty několika vybraných článků: https://gitlab.com/tomkren/preprints

Díky za pozornost :)