Program Boosting: Program Synthesis via Crowd-Sourcing

Cochran, Robert A., et al.

presenter name(s) removed for FERPA considerations

What is a regular expression (regex)?

- A regular expression defines a pattern that strings must match

- What would a regex for **phone numbers** look like?

Ans1: ^[0-9]{3}-[0-9]*-[0-9]{4}$?

and

Ans2: ^[0-9]{3}-[0-9]{3}-[0-9]*$?

becomes

^[0-9]{3}-[0-9]{3}-[0-9]{4}$

A regular expression for phone numbers with hyphens!

“A regex for phone numbers?”

- Was “**^[0-9]{3}-[0-9]{3}-[0-9]{4}$**” also your answer in mind?

- What about no hyphens? With spaces?
  Country code? Which country?
  Any extensions?

People think differently!

Observation: People think differently!

- It might be due to **ambiguity** in task **specification**
- Or they just consider different cases, cover different bases

- **Simple** task can be really **challenging!**

Research Questions

- Can **crowd-sourcing** improve solution **accuracy** on difficult programming tasks, by synthesizing over individual programmers' inaccurate solutions?
  - *Task such as coming up with a **regular expression** to recognize email addresses, URLs, phone numbers, and dates*

Key Idea: Crowdsourcing and Two-Crowds

- While people may get different parts wrong, **blending** these **partially** **incorrect** programs may provide better solutions

- **Experts/Developers** and **non-professional crowd** each can contribute different aspects to a task in the program boosting process

Example: Regex for URL - Bountify

BOUNTIFY POST BOUNTIES ABOUT

Regular expression to validate URLs

Please write a regular expression that *validates URLs* that may start with *http*, *https*, or *ftp*. Note that we are asking for original work. **Please do not copy your answer from other sites.**

All inputs below should be accepted by your regular expression.

- http://foo.com/blah_blah/
- http://foo.com/blah_blah_(wikipedia)
- http://foo.com/blah_blah_(wikipedia)_(again)
- http://www.example.com/wpstyle/?p=364

- http://foo.com/blah_blah_(wikipedia)_(again)
- http://www.example.com/wpstyle/?p=364
- https://www.example.com/foo/?bar=baz&inga=42&quux
- http://✪df.ws/123

Positive

- http://userid@example.com:8080
- http://userid@example.com:8080/
- http://userid:password@example.com

- http://-.z_.tr/#?+;=%40%80%2f%00@example%com[
- http://%@live.com
- http://(user%d@-example.com:8080/
- http://(userid@example.com:808%

Negaitive

- http://(userid@example.com:808%

Example: Regex for URL - From experts (Crowd #1)

Sign in Sign up

Regular expression to validate URLs

Please write a regular expression that *validates URLs* that may start with *http*, *https*, or *ftp*. Note that we are asking for original work. **Please do not copy your answer from other sites.**

All inputs below should be accepted by your regular expression.

$10

awarded to

Tags

:8080
:8080/

)080
)080/

/

cted by your regular expression:

62f%00@example%com[

:8080/
808%

submit

https://bountify.co/5f

Regex source	Regex length	True positive	True negative	Overall accuracy
@krijnhoetmer	115	.78	.41	.59
@gruber	71	.97	.36	.65
@gruber v2	218	1.00	.33	.65
@cowboy	1,241	1.00	.15	.56
@mattfarina	287	.72	.44	.57
@stephenhay	38	1.00	.64	.81
@scottgonzales	1,347	1.00	.15	.56
@rodneyrehm	109	.83	.36	.59
@imme_emosol	54	.97	.74	.85
@diegoperini*	**502**	**1.00**	**1.00**	**1.00**

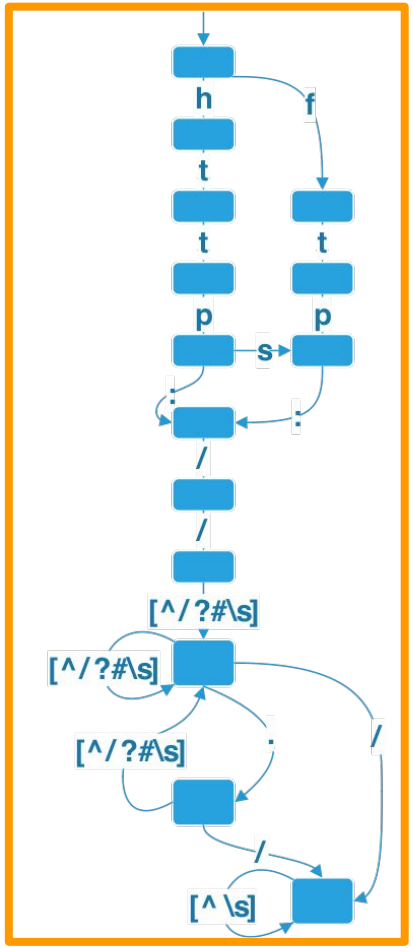
Figure 1: Representative regular expressions for URLs obtained from http://mathiasbynens.be/demo/url-regex. For every possible solution we show its length, true and false positive rates, and the overall accuracy. The last row is the winner.

http://pages.cs.wisc.edu/~loris/papers/popl15crowdboost.pptx

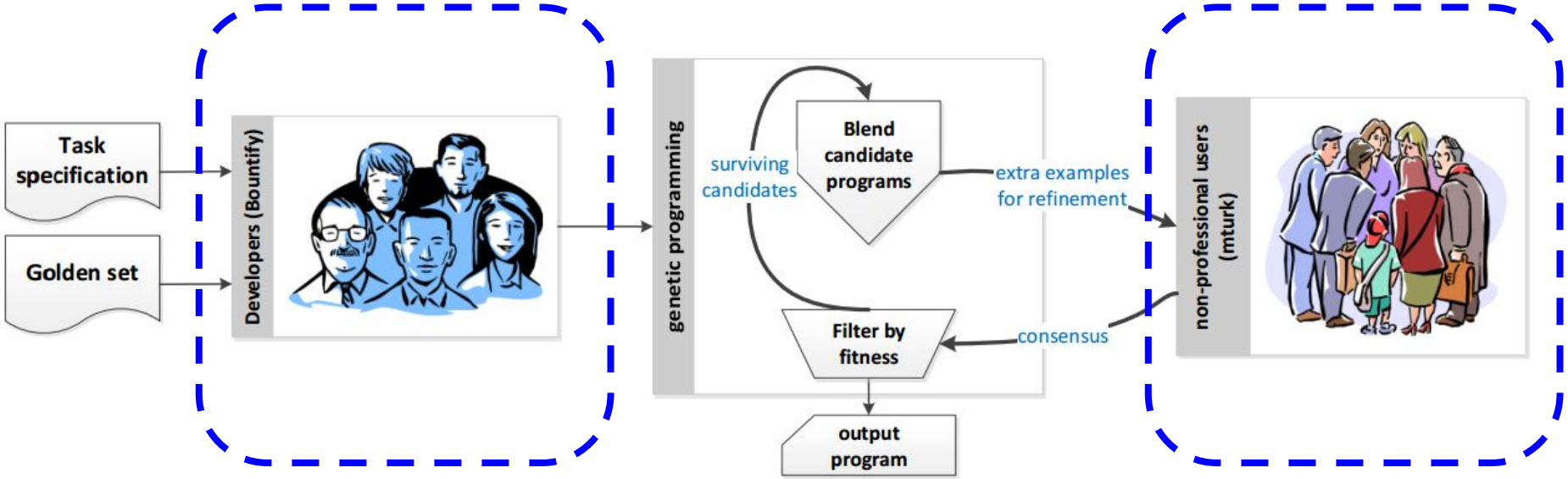
Representation: Symbolic Finite Automata (SFA)

- Extension of classical finite state automata to represent **regexes**

- Allow transitions to be labeled with predicates
  - Needed to handle UTF-16 (2^{16} characters)



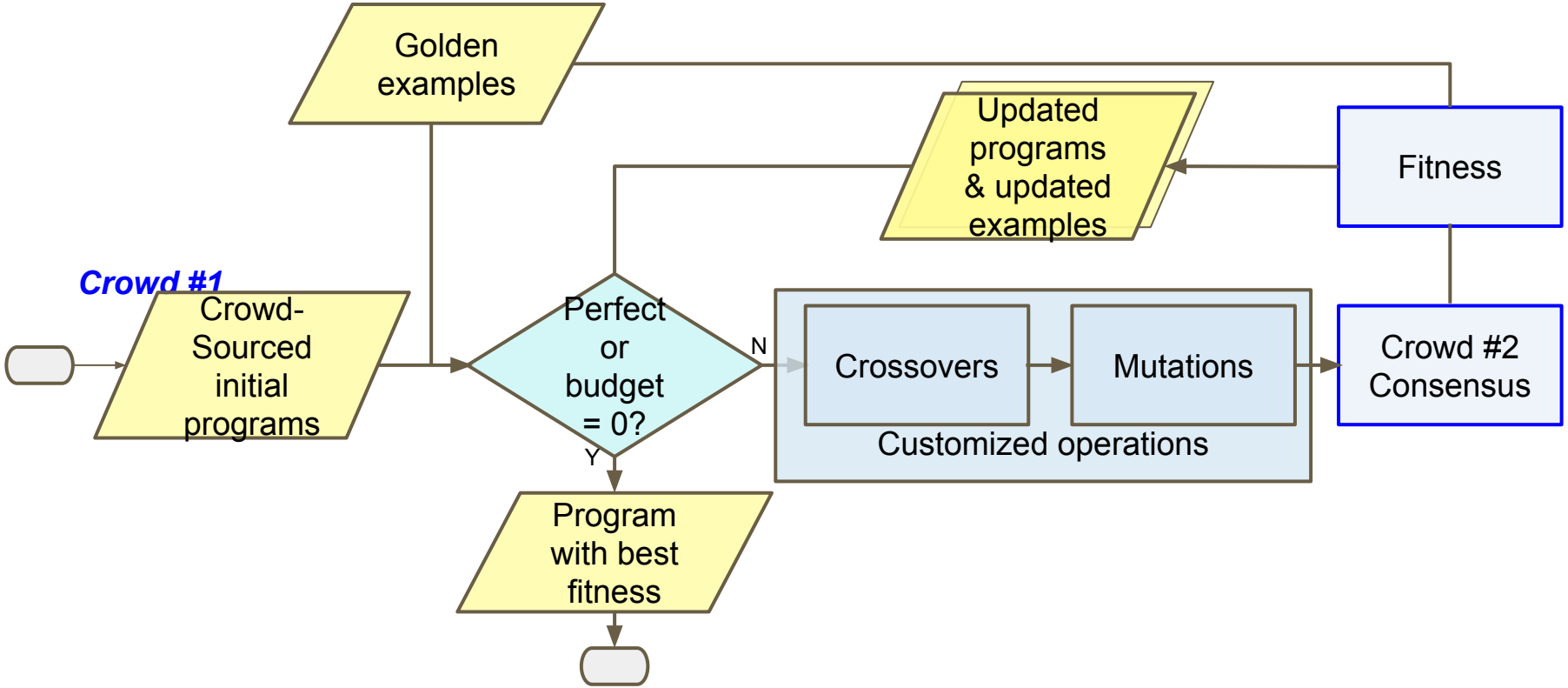
Program Boosting: System Architecture



Crowd #1
Expert/ Developers

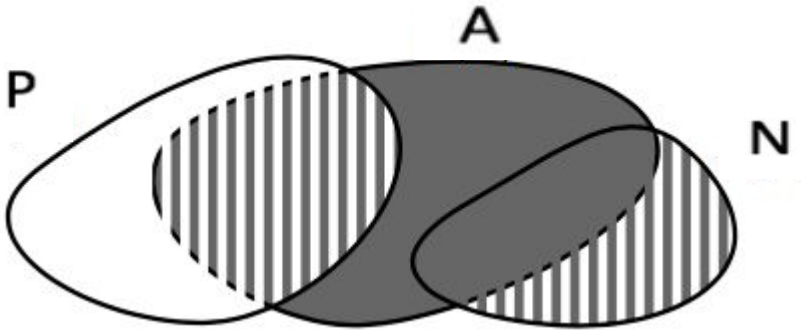
Crowd #2
Non-professional

Program Boosting: Iterative Genetic Programming



Fitness Scoring

How well does a program perform on a given data set?



A = Examples a program considered positive

P = **Positive** examples

N = **Negative** examples

*The **dashed** region is where the program is considered **correct**.*

$$\text{Accuracy} = \frac{|A \cap P| + |N \setminus A|}{|P \cup N|}$$

Experiment Evaluation

Sources

Initial Condition

Tasks	Initial Test Set +	Initial Test Set -	Candidate regexes	Candidate regex source: Bountify	Regexlib	Other
Phone numbers	20	29	8	3	0	5
Dates	31	36	6	3	1	2
Emails	7	7	10	4	3	3
URLs	36	39	9	4	0	5

Boosting Process

- Tested on **465** genetic programmed regexes
- Limited the boosting iteration to **10** for each regex
- Generated **0-207** test strings from each regex

Results

The more comprehensive test set for final evaluation

EVALUATED ON...	Initial Test Set			Crowd-Sourced Test Set		
		Boosted			Boosted	
Task	**initial**	**no crowd**	**crowd**	**initial**	**no crowd**	**crowd**
Phone numbers	0.80	0.90	0.90	0.79	0.88	0.91
Dates	0.85	0.99	0.97	0.78	0.78	0.95
Emails	0.71	0.86	0.86	0.79	0.72	0.90
URLs	0.67	0.91	0.88	0.64	0.75	0.89

Average 16.25% accuracy boost

Significant improvement on already high quality initial input.

Time and Cost

Time

- Main time cost is the classification latency from Mechanical Turk
- Between the four test categories, the averages ranged from 4 minutes to 37 minutes

Monetary Cost:

- Bountify: $5-$10 per question for total 4 questions
- Mechanical Turk: average $0.41-$3 per regex for 465 regexes

Contributions

- **Program Boosting**
  Method for **semi-automatic** program synthesis by blending a set of initial **crowd-sourced programs**
- **CROWDBOOST**
  Tool for generations of **complex regular expressions**
  - First to adopt **symbolic setting** with genetic programming using **Symbolic Finite Automata** (SFAs)
    - Supports complex alphabets such as UTF-16

Discussions Q1

Any other possible applications?
Other than Regex?

Discussions Q2

Money:
How do we compensate the programmers?
What if we can’t afford to pay them?

Discussions Q3

When to stop collecting answers?
How confident are you that the majority is correct?

Discussions Q4

How to counteract overfitting to the data?

Discussions Q5

There may be slow response times from the crowd.
How can this be fixed?

References

- Cochran, Robert A., et al. "Program boosting: Program synthesis via crowd-sourcing." 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pp. 677-688, Mumbai, India, 2015.
- Information Flocking Applied to Genetic Programming Visualization, slides
- In search of the perfect URL validation regex https://mathiasbynens.be/demo/url-regex
- Perini, Diego. “Regex-weburl.js.” *GitHub*. N.p., 5 Dec. 2010. Web. 01 Mar. 2017.
- Program Boosting: Program Synthesis via Crowd-Sourcing, POPL ‘15, slides
- Veanes, Margus. "Applications of symbolic finite automata." International Conference on Implementation and Application of Automata. Springer Berlin Heidelberg, 2013.
- Weimer, Westley, et al. "Automatically finding patches using genetic programming." Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009.

Thank You!

Regular expression to validate US phone numbers

Please write a regular expression that validates a US 10-digit phone numbers, with an optional US country code (1). Note that we are asking for original work.

Please do not copy your answer from other sites

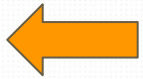
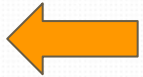
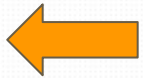
All inputs below should be accepted by your regular expression.

- 800 555 1212
- (828) 112 5555
- 404-555-1234
- 2141231243
- 1-241-123-1212

All of these inputs should be rejected by your regular expression:

- 789 789
- 44a 555 1234
- 2-555-214-7899
- 828)112-3555

Please provide the regular expression in the form **/^ YOUR ANSWER IS HERE $/** as part of your answer.
Please **test your regex on the samples provided before submitting**

$5

awarded to Researcher

Tags

perl
regular expression
PHP
javascript

Tweet submit

https://bountify.co/5b

Edits (1) Posted over 3 years ago by too4words

https://mathiasbynens.be/demo/url-regex

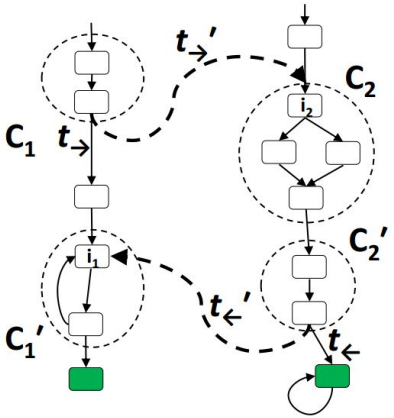
Example: Twitter Example for Regex

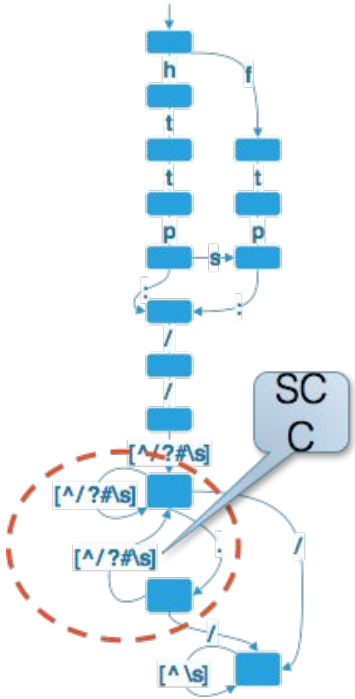
URL	Spoon Library	@krijnhoetmer	@gruber	@gruber v2	@cowboy	Jeffrey Friedl	@mattfarina	@stephenhay	@scottgonzales	@rodneyrehm
These URLs should match (1 → correct)										
http://foo.com/blah_blah	1	1	1	1	1	1	1	1	1	1
http://foo.com/blah_blah/	1	1	1	1	1	1	1	1	1	1
http://foo.com/blah_blah_(wikipedia)	1	1	1	1	1	1	1	1	1	0
http://foo.com/blah_blah_(wikipedia)_(again)	1	1	1	1	1	1	1	1	1	0
http://www.example.com/wpstyle/?p=364	1	1	1	1	1	1	1	1	1	1
https://www.example.com/foo/?bar=baz&inga=42&quux	1	1	1	1	1	1	1	1	1	1
http://✪df.ws/123	0	0	1	1	1	1	0	1	1	1
http://userid:password@example.com:8080	0	1	1	1	1	0	1	1	1	1
http://userid:password@example.com:8080/	0	1	1	1	1	0	1	1	1	1
http://userid@example.com	0	1	1	1	1	0	1	1	1	1
http://userid@example.com/	0	1	1	1	1	0	1	1	1	1
http://userid@example.com:8080	0	1	1	1	1	0	1	1	1	1
http://userid@example.com:8080/	0	1	1	1	1	0	1	1	1	1
http://userid:password@example.com	0	1	1	1	1	0	1	1	1	1
http://userid:password@example.com/	0	1	1	1	1	0	1	1	1	1
http://142.42.1.1/	0	1	1	1	1	1	1	1	1	1
http://142.42.1.1:8080/	0	1	1	1	1	1	1	1	1	1
http://➡.ws/䨻	0	0	1	1	1	0	0	1	1	0
http://⌘.ws	0	0	1	1	1	0	0	1	1	1
http://⌘.ws/	0	0	1	1	1	0	0	1	1	1

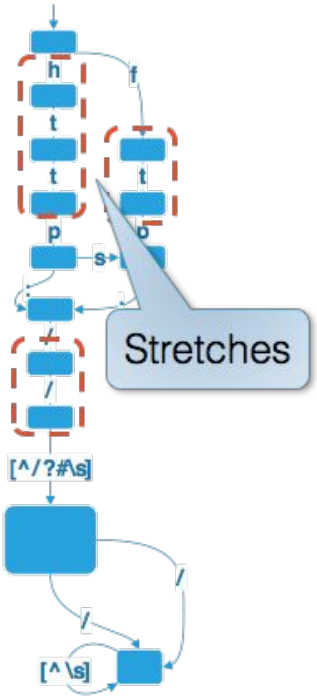
- **URL validation challenge from Mathias Bynens, Dec. 10**

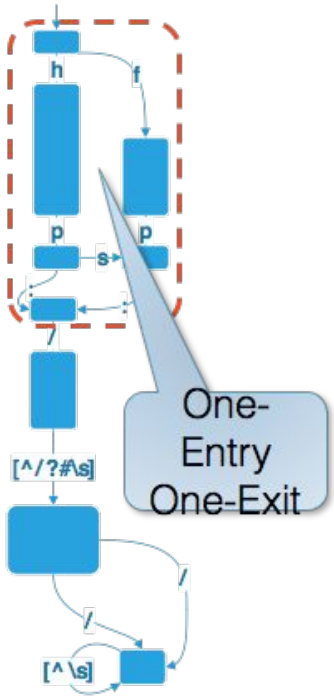
Genectic Programming Operation: Crossover

- Redirecting Edges
- Collapsing Stretches
- One-way Crossovers



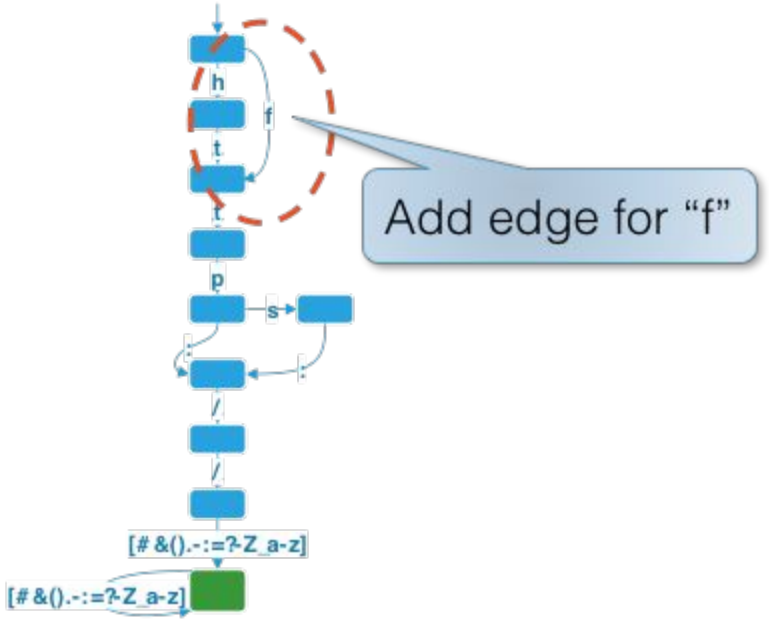






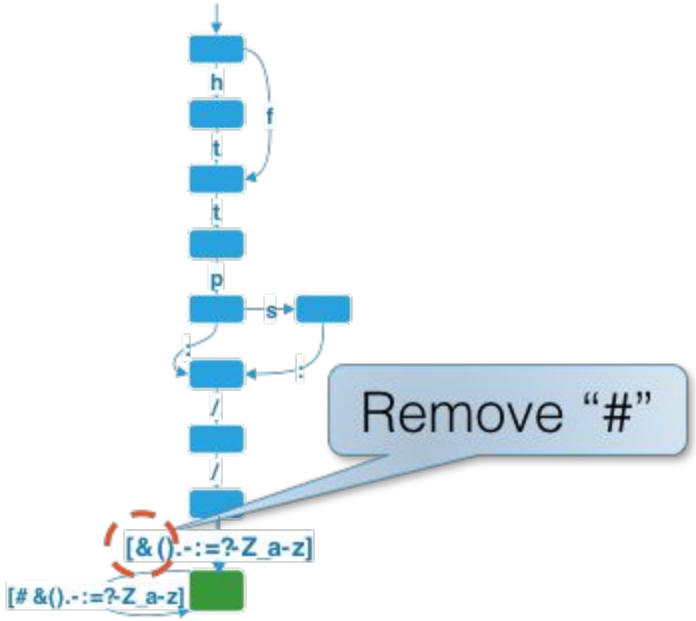
Genectic Programming Operation: Mutation

- Positive Mutation: ftp://foo.com



Augmenting mutations

- Negative Mutation: http://#



Diminishing mutations

Example: Regex for URL

```
var re_weburl = new RegExp(
 "^" +
   "(?:(?:https?|ftp)://)" +
   "(?:\\S+(?::\\S*)?@)?" +
   "(?:" +
     "(?!(?:10|127)(?:\\.\\d{1,3}){3})" +
     "(?!(?:169\\.254|192\\.168)(?:\\.\\d{1,3}){2})" +
     "(?!172\\.(?:1[6-9]|2\\d|3[0-1])(?:\\.\\d{1,3}){2})" +
     "(?:[1-9]\\d?|1\\d\\d|2[01]\\d|22[0-3])" +
     "(?:\\.(?:1?\\d{1,2}|2[0-4]\\d|25[0-5])){2}" +
     "(?:\\.(?:[1-9]\\d?|1\\d\\d|2[0-4]\\d|25[0-4]))" +
   "|" +
     "(?:(?:[a-z\\u00a1-\\uffff0-9]-*)*[a-z\\u00a1-\\uffff0-9]+)" +
     "(?:\\.(?:[a-z\\u00a1-\\uffff0-9]-*)*[a-z\\u00a1-\\uffff0-9]+)*" +
     "(?:\\.(?:[a-z\\u00a1-\\uffff]{2,}))" +
     "\\.?" +
   ")" +
   "(?::\\d{2,5})?" +
   "(?:[/?#]\\S*)?" +
 "$", "i"
);
```

Regex source	Regex length	True positive	True negative	Overall accuracy
@krijnhoetmer	115	.78	.41	.59
@gruber	71	.97	.36	.65
@gruber v2	218	1.00	.33	.65
@cowboy	1,241	1.00	.15	.56
@mattfarina	287	.72	.44	.57
@stephenhay	38	1.00	.64	.81
@scottgonzales	1,347	1.00	.15	.56
@rodneyrehm	109	.83	.36	.59
@imme_emosol	54	.97	.74	.85
@diegoperini*	**502**	**1.00**	**1.00**	**1.00**

Figure 1: Representative regular expressions for URLs obtained from http://mathiasbynens.be/demo/url-regex. For every possible solution we show its length, true and false positive rates, and the overall accuracy. The last row is the winner.