Rust in Python: Fixing regular expressions

Andrew Roden: andrew@roden.me
David Blewett: david@dawninglight.net

Structure

- What is the problem? -- Demo
- Why is this slow? -- What is automata?
- What to do? -- Building an intuition
- Using a better algorithm -- Intro to Rust
- How python and rust work together providing a solution

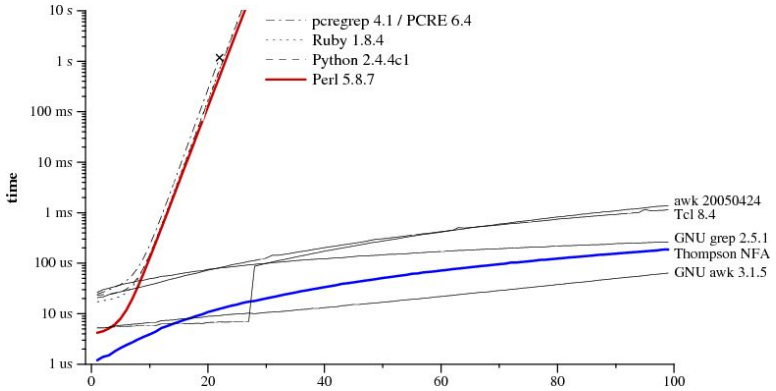
Part 1 - Demo

- Regexp matching URL or email
  - r'([a-zA-Z][a-zA-Z0-9]*)://([^ /]+)(/[^]*)? | ([^ @]+)@([^ @]+)'
- Worst case is not matching string
  - Using zen of python string
  - import this; a_string = this.s.decode('rot13')
- Timing of Python's regular expression vs Rure
  - {'python': 6.703774929046631, 'rure': 0.46964287757873535}

CROWDSTRIKE

How bad?

Think bubble sort vs quick sort



regular expression and text size n; $a?^n a^n$ matching a^n

https://swtch.com/~rsc/regexp/regexp1.html

Part 2 - Why?

- Core concept of Regular expressions introduced in the 1950's
- The first major implementation was done by Ken Thompson in 1960's
- Deep roots as
  - text editor tool (grep/vim)
  - lexer for programming languages

What in the 1960s?

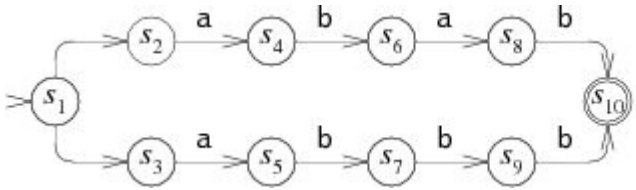
- Deterministic finite automaton
  - All states are determined by precise input
- Non-deterministic automaton is similar but includes “guesses” about the next state. This is the backtracking

CROWDSTRIKE

Non-deterministic Finite Automata

Given input “abbc” see the “backtracking”

https://swtch.com/~rsc/regexp/regexp1.html



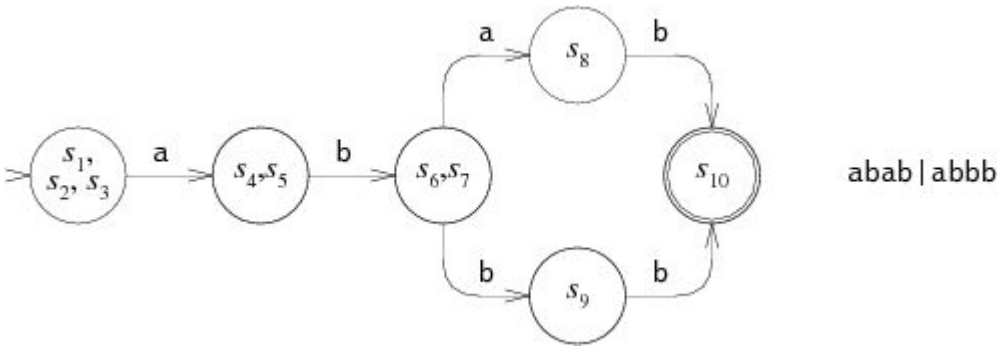
abab | abbb

CROWDSTRIKE

Deterministic Finite Automata

All states are determined by precise input. Consider “abbc” again

https://swtch.com/~rsc/regexp/regexp1.html#ritchie



Why have non-DFA at all?

- Backtracking buys some really cool regexp features
  - Look ahead/behind
  - Backreference matching groups
  - Conditional branching
- Python - Batteries included!

Part 3 - Building intuition

- Many programming languages use backtracking
  - Java/C#/Ruby/Python/Perl
- Microsoft put it well
  - In general, a Nondeterministic Finite Automaton (NFA) engine like .NET regular expression engine places the responsibility for crafting efficient, fast regular expressions **on the developer**.

Worst cases

- “evil” matches have greedy qualifier
  - .compile a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?a?aaaaaaaaaaaaaaaaa
  - .search aaaaaaaaaaaaaaaaa
- “evil” non-matches
  - minimize separates

Example 1

- NFA bad; abab|abbb
- Better: ab(a|b)b

- The naive NFA diagram now more closely resembles the DFA

Example 2

- Consider matching list of directories

Bad:

/data/foo | /data/bar/ | /data/whatever

Better:

/data/ (foo | bar | whatever)

Example 3

- Reduce overlap of selectors

Bad CSV match:

```
.*, .*, .*
```

Better:

```
[^,]* , [^.]* , [^.]*
```

What Is Rust?

- New systems language
- Replacement for C
- Focus on memory safety and race-free operation
- Not garbage collected (but can use a garbage collector)
- Uses a borrow-checker
- Packages are called crates, installed with cargo

What Is Rust?

- Install with rustup
- Playground
- Guessing Game Tutorial

What is rure?

- rure is a C API to Rust's regex library
- Guarantees linear time searching using finite automata
- Must give up backreferences and arbitrary lookaround
- Includes capturing groups, lazy matching, Unicode support and word boundary assertions
- Matching semantics correspond to Perl's, or "leftmost first"

What is rure-python?

- CFFI interface to rure
  - Python <-> C <-> Rust
  - Supports Python 2 and Python 3
- No support for compiling rure automatically
  - Expects pre-built shared object/DLL to bundle into the wheel

Design Decisions

- Functionality split between high-level (Pythonic) and low-level (Rusty)
- High-level
  - import rure as re
  - accepts only Unicode strings
  - API compatible with stdlib
  - Raises exceptions on expressions not supported by rure
  - Transparent translation of stdlib options to rure

Design Decisions

- Functionality split between high-level (Pythonic) and low-level (Rusty)
- Low-level
  - rure.lib.Rure
  - accepts only UTF8-encoded byte strings
  - Pay for what you use

Links

- https://github.com/davidblewett/rure-python
- https://www.rustup.rs/
- https://www.rust-lang.org/en-US/documentation.html
- https://play.rust-lang.org/
- https://github.com/rust-lang/regex/tree/master/regex-capi
- https://cffi.readthedocs.io/en/latest/
- https://swtch.com/~rsc/regexp/regexp1.html