

Advanced Approaches for Data Languages

Denial-of-service-resistant matching of regexes with backreferences

Ing. Jan Vašák • supervisor: doc. Ing. Ondřej Lengál, Ph.D.

Motivation

- **ReDoS** (regular expression denial of service): an **attack** that exploits the algorithmic complexity of regex matching.
- **Backreferences** let a regex refer back to an already matched part of the input, for example, `/(.).*\1$/` matches any text whose last character occurs earlier in the text.
- All state-of-the-art tools (see Table 1) match regexes with backreferences (**REBR**) using **backtracking** with **exponential** worst-case complexity.
- Hence no existing matcher guarantees **ReDoS-free** REBR matching.

Register Set Automata

- **Register Automata (RA)** have a finite number of registers, each holding a single symbol. They capture a subclass of REBRs, but are **not determinizable** in general.
- **Register Set Automata (RSA)** store sets of values in their registers to deterministically represent a large class of RAs. For a constructed RSA, matching has **linear worst-case complexity** w.r.t. the input.

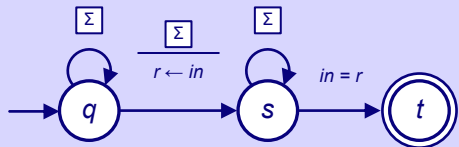


Figure 1. Non-deterministic RA for `/(.).*\1$/` over the (potentially infinite) alphabet Σ

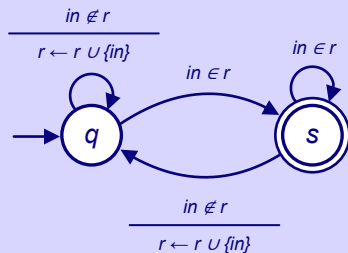


Figure 2. Deterministic RSA for the same regex; registers hold sets of symbols

Efficient REBR Matching

- RA construction and **determinization** run in Python; the RSA is then compiled to **VM code** executed by a **C++** virtual machine (**rsacpp**).
- The tool **RENGAR** generated adversarial inputs for **1,305 REBRs** from practice, run against industrial matchers with backreference support.
- In Table 1 the **> 0.1 s** and **> 1 s** columns count ReDoS vulnerabilities of two severities.
- **Zero ReDoS vulnerabilities:** rsacpp finishes all 1,305 attacks under 0.1 s, while **every industrial matcher times out on 2–48 inputs and runs over a second on 25–115 inputs.**

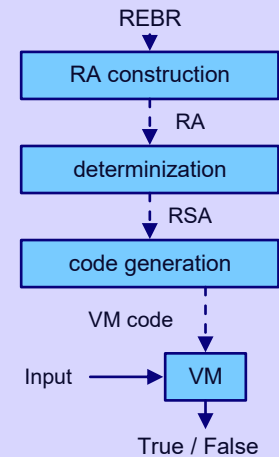


Table 1. Experimental results, TO = 100 s; errors = no result returned (crash or internal timeout); – = rsacpp defines the accepted regex set.

Figure 3. Pipeline of the VM matcher

tool	< 0.1 s	0.1–1 s	1–5 s	5–10 s	10–50 s	50–100 s	TOs	> 0.1 s	> 1 s	errors	mean	std.dev
rsacpp	1305	0	0	0	0	0	0	0	0	–	0.002	0.002
grep	1153	19	6	1	16	7	17	66	47	86	0.846	7.102
re	1084	106	40	14	10	3	48	221	115	0	0.511	4.293
pcr2	1031	27	15	4	4	0	2	52	25	222	0.14	1.179
js	645	59	8	1	12	1	37	118	59	542	0.539	3.711
java	554	76	18	5	7	5	23	134	58	617	0.975	6.491
.net	653	41	30	1	11	3	28	114	73	538	0.852	6.009

Published at PLDI '26 (CORE A*)

HAVLENA, V.; HOLÍK, L.; LENGÁL, O.; VAŠÁK, J.; GULČÍKOVÁ, S. *Towards Efficient Matching of Regexes with Backreferences using Register Set Automata*. PLDI'26.