Software Engineering, Theory and Experimental Programming

Bauman Moscow State T.U. & PSI RAS

Expressibility of Languages with Capture Operations: an Overview

Antonina Nepeivoda, a_nevod@mail.ru

A Precursor: Pattern Languages

Dana Angluin, 1980: Finding Patterns Common to a Set of Strings.

- Letters in an alphabet Σ;
- Variables: matching against strings in Σ^*.

Non-trivial language inclusion for finite alphabets (undecidable for erasing case): $XabY$ and $XaZbY$ match against the same set of strings in $\{a, b\}^*$, i.e. $\mathscr{L}(XabY) = \mathscr{L}(XaZbY)$ if $\Sigma = \{a, b\}$.

Angluin's infamous theorem

Matching a string against a pattern is NP-complete.

In practice:

- Efficient exact matching techniques for restricted classes of patterns (regular, bounded-width).
- Techiques for approximate matching.

Extended Regular Expressions

Evolved from 1980s and Perl regular engine:

- lookaheads, lookbehinds — positive, negative. Positive lookahead syntax:
  $(? = r_1)r_2$ (matching a regex r_2 whose prefix satisfies r_1. For readability, sometimes we assume that lookahead matches against the whole string)

- capture groups and backreferences (recursive and not). Syntax:
  $(aa^+)\backslash 1^+$ — capture groups are numbered by parentheses ordering. Lookaheads are not capture groups.

- recursive definitions. Recursive definition syntax:
  $(a(?1)b|c)$ — the syntax (not the matched string) of 1-st capture group is reused.

Regex Classes Currently Investigated

- Non-recursive backref-regexes ($\mathcal{RE}w\mathcal{BR}^-$): Campeanu–Salomaa–Yu formalism (2003);
- Recursive backref-regexes ($\mathcal{RE}w\mathcal{BR}$): Shmid formalism, memory finite automata (2016);
- Recursive backref-regexes with lookahead $\mathcal{RE}w\mathcal{BR} + \mathcal{LA}$: Chida–Terauchi extended MFA formalism (2023).

Backreferences Formalisations

- Appeared much later than implementations of backref-regexes.
- Some almost repeated PCRE (Perl) backref-regexes.

Campeanu–Salomaa-Yu (CSY) formalisation:

$$\begin{cases} (\tau) & \text{(anonymous capturing)} \\ \backslash k & \text{(reading memory cell from k-th group)} \end{cases}$$

Example: $(\texttt{aa}^+)(\backslash\texttt{1})^+$ defines $\{a^n \mid \text{n is not prime}\}$

- Any capture group is initialized exactly once.
- Any reference must be preceded by the capture group textually.

Backreferences Formalisations

- Currently most used in the theoretical scope: only named capture groups.

Backref-regex (ref-words, by Shmid) operations:

$$\begin{cases} [_k \tau]_k & \text{(named capturing)} \\ \&k & \text{(reading memory cell)} \end{cases}$$

Example: $[_1\texttt{a}^*]_1\texttt{a}^+\texttt{b}\&\texttt{1}$ defines $\{a^m b a^n \mid m > n\}$

- ε-semantics (Schmid) — uninitialized reference recognizes $\{\varepsilon\}$;
- $\varnothing$-semantics (regex engines) — uninitialized reference recognizes $\varnothing$.

No impact on language properties.

Backreferences Formalisations

- Currently most used in the theoretical scope: only named capture groups.

Backref-regex (ref-words, by Shmid) operations:

$$\begin{cases} [_k \tau]_k & \text{(named capturing)} \\ \&k & \text{(reading memory cell)} \end{cases}$$

Example: $[_1\mathtt{a}^*]_1\mathtt{a}^+\mathtt{b}\&\mathtt{1}$ defines $\{a^m b a^n \mid m > n\}$

- Possibly unbalanced and nested (but not self-nested) capturing.
- References on k-th memory cell cannot occur inside a capturing group for k.

Word Equations

- Given a pair of patterns Φ, Ψ sharing common variables $X_1, \ldots, X_n$, we say a tuple $\langle \omega_1, \ldots, \omega_n \rangle$ is a solution set to equation $\Psi \doteq \Phi$ if a morphism defined by $X_i \mapsto \omega_i, \gamma \mapsto \gamma$ turns both patterns into equal strings.

- Given equation $E : aX_1X_1bX_2 \doteq X_1aX_2bX_1$, the set $\{\langle a^n, (a^nb)^m a^n \rangle \mid m, n \in \mathbb{N}\}$ is the solution set to E.

  Note: X_2-projection of the set is not context-free. Still, the X_2-projection belongs to the class of Okhotin's conjunctive languages (discussed later).

Word Equations

- Given a pair of patterns Φ, Ψ sharing common variables $X_1, \ldots, X_n$, we say a tuple $\langle \omega_1, \ldots, \omega_n \rangle$ is a solution set to equation $\Psi \doteq \Phi$ if a morphism defined by $X_i \mapsto \omega_i$, $\gamma \mapsto \gamma$ turns both patterns into equal strings.
- A k-projection of a solution set $\big\{ \langle \omega_1, \ldots, \omega_n \rangle \mid \Phi[\omega_1, \ldots, \omega_n] = \Psi[\omega_1, \ldots, \omega_n] \big\}$ is the set of k-components of the tuples.
- If a string set $\mathscr{L}$ is a projection of the solution set of a word equation $\Phi \doteq \Psi$, then $\mathscr{L}$ is said to be a language of $\Phi \doteq \Psi$.

Captured Values and Patterns

A word equation over k variables defines k languages.
As a pattern (i.e. a pair of patterns defining the same string), a word equation $\Phi \doteq \Psi$ defines a single language, as usual, $\{\Phi[\omega_1, \ldots, \omega_n] \mid \Phi[\omega_1, \ldots, \omega_n] = \Psi[\omega_1, \ldots, \omega_n]\}$.

We distinguish the first languages class $\mathcal{WE}_\pi$ and the second $\mathcal{WE}$ and note that $\mathcal{WE} \subset \mathcal{WE}_\pi$, since $a^* \in \mathcal{WE}_\pi$, but every equation defining such a variable value must have at least one explicit a-occurrence, hence its side after any substitution cannot be valued ε.

$\mathcal{REwBR}$ and $\mathcal{REwBR}_\pi$

$\mathcal{REwBR} \subset \mathcal{REwBR}_\pi$

Given $\mathscr{L} = \{a^n b^n \mid n \in \mathbb{N}\}$, it can be constructed as a captured value $\&3$ in the ref-word

$$\big([_2 a\&1 b]_2 [_1 a\&2 b]_1\big)^* [_3 \& \mid \&2]_3$$

$\mathscr{L}$ is not expressible by a $\mathcal{REwBR}$: both a-block and b-block have to contain an iteration $\Rightarrow$ asynchronous growth.

Known Analysis Techniques

- $\mathcal{REwBR}$ and $\mathcal{REwBR} + \mathcal{LA}$, known proofs refer to the restricted CSY formalisation $\Rightarrow$ no known formal techniques of disproving expressibility for multiple capturing case.

- For $\mathcal{WE}$, W. Plandowski in 2000s developed a combinatorial method relying on synchronising $\mathcal{F}$-codes.

- In 2023, J. Day used the method to prove that all the star subexpressions Φ^* of a thin regular language recognised by a word equation language has the following property: every two elements ω_1, ω_2 of the language recognised by Φ commute, i.e. $\omega_1\omega_2 = \omega_2\omega_1$.

$\mathcal{REwBR}$ and $\mathcal{REwBR} + \mathcal{LA}$: Growth Argument

The factorial star language $\{(a^{k!})^+ba^k\}$:

$$\left((?{=}\ [_2a\&1]_2a^*b)(?=[_1\&2]_1a^*b)(?{=}\ \&2^*b)\right)^* a^*b\&2$$

Main reason of non-expressibility in $\mathcal{REwBR}$: given a maximal length multiplier N^N (i.e. given length of the regex N), the words $(a^{N^N!})^+ba^{N^N}$ cannot be expressed, because of the abnormal growth of the first a-block.

In Chida's paper, the CSY formalism was referred with a non-expressible language $\left\{a^iba^{i+1}ba^{i\cdot(i+1)\cdot k} \mid i, k \in \mathbb{N}\right\}$.

REGEX and $\mathcal{CNJ}$

REGEX includes conjunctive languages by Okhotin expressible by the rules augmented with conjunction:

$$\{A_i \to \Phi_1 \,\&\, \cdots \,\&\, \Phi_2 \mid A_i \in N \,\&\, \Phi_j \in (N \cup \Sigma)^*\}$$

Example: a grammar for $\{(a^n b)^k \mid n, k \in \mathbb{N}\}$.

$$\begin{aligned} S &\to SA \,\&\, Cb \mid A \\ A &\to aA \mid ab \\ C &\to aCa \mid B \\ B &\to BA \mid b \end{aligned}$$

May be expressed with the use of recursion and lookaheads:

$$\Big(\overbrace{(?= \underbrace{(a(?2)a|b|b.^*b)}_{\text{capture group 2}} b) a^* b (?1) | a^* | \underbrace{(a(?3)a|b)}_{\text{group 3}}}^{\text{capture group 1}}\Big) b$$

Technique: same to μ-regexes for CFG, augmented with lookaheads.

Linear $\mathcal{CFL}$ and $\mathcal{REwBR}_\pi$

A language is linear-context-free $\Leftrightarrow$ it is expressed by a CFG having at most one non-terminal in each rule rhs.

Every linear $\mathcal{CFL}$ can be expressed by a capture group language.

- For every non-terminal N_i, introduce a pair of memory cells i, i' (for capturing in an lhs and for recapturing).
- Each recursive rule $N_i \to \omega_1 N_j \omega_2$ is rewritten into a subexpression $r_k : [_{i'}\omega_1 \& j\omega_2]_{i'} [_i \& i']_i$. Terminal rules for every nonterminal N_i are gathered into a disjunction and are rewritten into an expression r'_i in the similar way.
- A total expression is $r'_1 \dots r'_m (r_1 | \dots | r_k)^*$.

$\mathcal{CFL}$ and $\mathcal{REwBR}_\pi$

Some non-linear (and non-linear-conjunctive) languages such as $\{(ww^R)(vv^R) \mid w, v \in \Sigma^*\}$ can be generated by cells of $\mathcal{REwBR}$. Some non-linear context-free languages cause encoding problems (e.g. balanced parentheses language). Reason: unbounded treewidth of the derivation tree.

On the other hand, using the $\mathcal{CNJ} \subset$ REGEX technique, the balanced parenthesis language can be expressed by $\mathcal{REwBR}+\mathcal{LA}_\pi$. Idea: construct a Trellis automaton and transform it to a linear $\mathcal{CNJ}$.

d	a	b	d	r
a	a	d	a	a
b	r	b		r
d		b		a
r	r	b	b	r

$\mathcal{WE}_\pi$ and $\mathcal{REwBR} + \mathcal{LA}_\pi$

- Given $\Phi(X_1, \ldots, X_n) \doteq \Psi(X_1, \ldots, X_n)$ with a canonical numeration of variables, every first occurrence of a variable in $\Phi\Psi$ is replaced with a capture group $(.^*)$, while the other variable occurrences are replaced by $\backslash k$ operations. The resulting equation sides are Φ', Ψ'.
- Regex $(?{=}\ \Phi')\Psi'$ recognises the solution projections in its capture groups.

Main reason: capture-preserving lookaheads in REGEX.

For instance, in the expression $\big(?{=}\ (.^*)ab(.^*)\big)\backslash 2ba\backslash 1$ both first and second memory cells are captured by the lookahead checker.

$q\mathcal{WE}_\pi$ and $\mathcal{RE}w\mathcal{BR}_\pi$

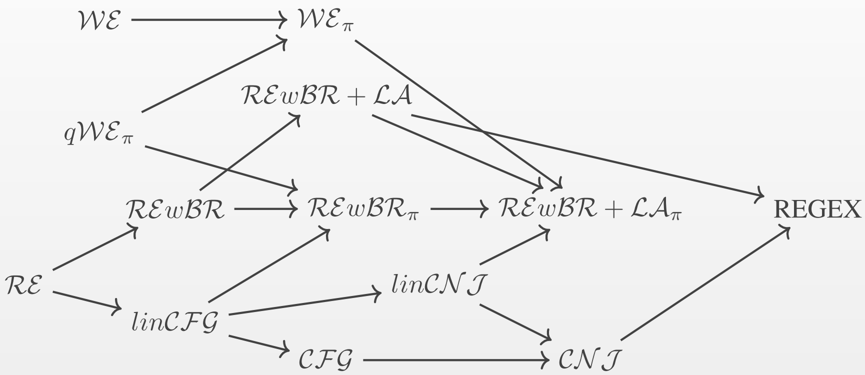
In a quadratic word equation, each variable occurs at most twice.

- Every solution to a quadratic word equation is expressed by a graph containing the assignments: $X \mapsto YX$, $X \mapsto Y$, $X \mapsto aX$, $X \mapsto \varepsilon$.
- Can be transformed to an MFA moving from graph leaves to its root with re-capturing. E.g. $X \mapsto YX$ is represented by $[_{X'} \& Y \& X]_{X'} [_X \& X']_X$ subexpression.

Variable values in $XabY \doteq YbaX$ are captured by

$$([_X a^*]_X [_Y \& X a]_Y | [_Y b^*]_Y [_X \& Y b]_X) \left([_{X'} \& Y ba \& X]_{X'} [_X \& X']_X | [_{Y'} \& X ab \& Y]_{Y'} [_Y \& Y']_Y \right)^*$$

Summary: What is Known?



Open Problems

- Techniques for disproving membership in $\mathcal{RE}w\mathcal{BR}$, $\mathcal{RE}w\mathcal{BR} + \mathcal{LA}$ classes?
- Clarifying position of word-equation-languages in the regex languages hierarchy: $\mathcal{RE}w\mathcal{BR}$, $\mathcal{RE}w\mathcal{BR}_\pi$, $\mathcal{RE}w\mathcal{BR} + \mathcal{LA}$ or $\mathcal{RE}w\mathcal{BR} + \mathcal{LA}_\pi$?
- Determining a cone over $\mathcal{WE}$ and $\mathcal{RE}w\mathcal{BR}$: a minimal language class closed under intersections with regular languages, homomorphism and inverse homomorphism operations.

References-I

1. J. Karhumäki, F. Mignosi, and W. Plandowski, The expressibility of languages and relations by word equations, J. ACM, vol. 47, no. 3, pp. 483–505, (2000).
2. Joel D. Day, Vijay Ganesh, Nathan Grewal, Matthew Konefal, Florin Manea: A Closer Look at the Expressive Power of Logics Based on Word Equations. Theory Comput. Syst. 68(3): 322-379 (2024).
3. Alexander Okhotin: Conjunctive Grammars. J. Autom. Lang. Comb. 6(4): 519-535 (2001)
4. Nariyoshi Chida, Tachio Terauchi: On Lookaheads in Regular Expressions with Backreferences. IEICE Trans. Inf. Syst. 106(5): 959-975 (2023)

References-II

5. Henning Fernau, Florin Manea, Robert Mercas, Markus L. Schmid: Pattern Matching with Variables: Efficient Algorithms and Complexity Results. ACM Trans. Comput. Theory 12(1): 6:1-6:37 (2020).
6. Markus L. Schmid: Characterising REGEX languages by regular languages equipped with factor-referencing. Inf. Comput. 249: 1-17 (2016).
7. Cezar Câmpeanu, Kai Salomaa, Sheng Yu: A Formal Study Of Practical Regular Expressions. Int. J. Found. Comput. Sci. 14(6): 1007-1018 (2003).
8. Yuya Uezato: Regular Expressions with Backreferences and Lookaheads Capture NLOG. ICALP 2024: 155:1-155:20.
9. M. Berglund and B. van der Merwe, “Re-examining regular expressions with backreferences”, Theoretical Computer Science, vol. 940, pp. 66–80, 2023.