

9-Month Progress Presentation

Meshal Binnasban

King's College London — Department of Informatics

November 11, 2025

Brzozowski's Derivative

$$der_a(\mathbf{0}) \Rightarrow \mathbf{0}$$

$$der_a(\mathbf{1}) \Rightarrow \mathbf{0}$$

$$der_a(c) \Rightarrow \begin{cases} \mathbf{1} & \text{if } a = c, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$der_a(r_1 + r_2) \Rightarrow der_a(r_1) + der_a(r_2)$$

$$der_a(r_1 \cdot r_2) \Rightarrow der_a(r_1) \cdot r_2 + \begin{cases} der_a(r_2) & \text{if } nullable(r_1), \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$der_a(r^*) \Rightarrow der_a(r) \cdot r^*$$

Brzozowski's derivatives

Matching Example

Regular expression $(ab + ba)$ String ba

$$\begin{aligned}der_b r &= der_b(ab + ba) \\&= der_b(ab) + der_b(ba) \\&= (der_b a) \cdot b + (der_b b) \cdot a \\&= \mathbf{0} \cdot b + \mathbf{1} \cdot a\end{aligned}$$

$$\begin{aligned}der_a(der_b r) &= der_a(\mathbf{0} \cdot b + \mathbf{1} \cdot a) \\&= der_a(\mathbf{0} \cdot b) + der_a(\mathbf{1} \cdot a) \\&= der_a(\mathbf{0}) \cdot b + (der_a(\mathbf{1}) \cdot a + der_a a) \\&= \mathbf{0} \cdot b + (\mathbf{0} \cdot a + \mathbf{1})\end{aligned}$$

Size Explosion

Regular expression $((a)^* + (aa)^* + (aaa)^* + (aaaa)^* + (aaaaa)^*)^*$ String $\underbrace{a \dots a}_n$

$$\begin{aligned}der_a r &= der_a (((a)^* + (aa)^* + (aaa)^* + (aaaa)^* + (aaaaa)^*)^*) \\&= der_a ((a)^* + (aa)^* + (aaa)^* + (aaaa)^* + (aaaaa)^*) \cdot r^* \\&= ((a)^* + (a \cdot (aa)^*) + (aa \cdot (aaa)^*) + \dots) \cdot r^*\end{aligned}$$

$$\begin{aligned}der_a (der_a r) &= der_a ((a)^* + (a \cdot (aa)^*) + (aa \cdot (aaa)^*) + \dots) \cdot r^* \\&\quad + der_a r^* \\&= ((a)^* + (aa)^* + (a \cdot (aaa)^*) + \dots) \cdot r^* \\&\quad + ((a)^* + (a \cdot (aa)^*) + (aa \cdot (aaa)^*) + \dots) \cdot r^*\end{aligned}$$

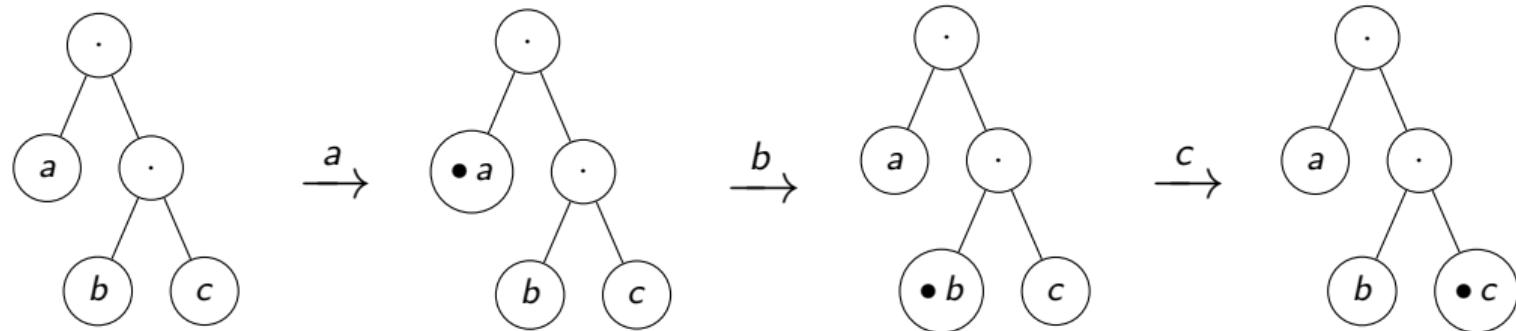
$$\begin{aligned}der_a (der_a (der_a r)) &= (der_a ((a)^* + (aa)^* + (a \cdot (aaa)^*) + \dots) \cdot r^* \\&\quad + der_a r^*) + (der_a ((a)^* + (a \cdot (aa)^*) + (aa \cdot (aaa)^*) \\&\quad + \dots) \cdot r^*) + der_a r^*\end{aligned}$$

Without simplifications: $n = 4 \Rightarrow 985$, $n = 10 \Rightarrow 65,353$.

With simplifications: $n = 4 \Rightarrow 315$, $n = 10 \Rightarrow 777$.

Marked Approach: Fischer et al. / Asperti et al.

Regular expression $a \cdot (b \cdot c)$ String abc



Our Definition of *shifts*

$$\text{shifts}(ms, 0) \Rightarrow []$$

$$\text{shifts}(ms, 1) \Rightarrow []$$

$$\text{shifts}(ms, c) \Rightarrow [\bullet s \mid \bullet c :: s \in ms]$$

$$\text{shifts}(ms, r_1 + r_2) \Rightarrow \text{shifts}(ms, r_1) @ \text{shifts}(ms, r_2)$$

$$\text{shifts}(ms, r_1 \cdot r_2) \Rightarrow \text{let } ms' = \text{shifts}(ms, r_1) \text{ in}$$

$$\begin{cases} \text{shifts}(ms' @ ms, r_2) @ ms' & \text{if } \text{nullable}(r_1) \wedge \text{nullable}(r_2), \\ \text{shifts}(ms' @ ms, r_2) & \text{if } \text{nullable}(r_1), \\ \text{shifts}(ms', r_2) @ ms' & \text{if } \text{nullable}(r_2), \\ \text{shifts}(ms', r_2) & \text{otherwise.} \end{cases}$$

$$\text{shifts}(ms, r^*) \Rightarrow \text{let } ms' = \text{shifts}(ms, r) \text{ in}$$

$$\text{if } ms' = [] \text{ then } [] \text{ else } \text{shifts}(ms', r^*) @ ms'$$

Matching Example

Regular expression $(a + (a \cdot c))$ **String** ac

1. $|_{[\bullet ac]} (a + (a \cdot c))$
2. $(|_{[\bullet ac]} a) + (|_{[\bullet ac]} (a \cdot c))$
3. $(a |_{[\bullet c]}) + ((a \cdot c) |_{[\bullet []]})$
 $= [\bullet c, \bullet []]$

Future Work

- ▶ **Extend the matcher.** Add support for bounded repetitions, intersection, lookahead, and lookbehind operators.
- ▶ **POSIX value extraction.**
- ▶ **Formalisation in Isabelle/HOL.**
- ▶ **Exploring context-free generalisation.** If time allows, we aim to extend the marked approach to context-free grammars, as done for derivatives.

References I

-  A. Asperti, C. S. Coen, and E. Tassi.
Regular Expressions, au point.
arXiv, <http://arxiv.org/abs/1010.2604>, 2010.
-  J. A. Brzozowski.
Derivatives of regular expressions.
J. ACM, 11(4):481–494, Oct. 1964.
-  S. Fischer, F. Huch, and T. Wilke.
A Play on Regular Expressions: Functional Pearl.
In *Proc. of the 15th ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 357–368, 2010.
-  M. Might, D. Darais, and D. Spiewak.
Parsing with derivatives: A functional pearl.
In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pages 189–195. ACM, 2011.

References II



S. Owens, J. Reppy, and A. Turon.

Regular-expression derivatives re-examined.

Journal of Functional Programming, 19(2):173–190, 2009.



C. Tan and C. Urban.

POSIX Lexing with Bitcoded Derivatives, pages 26:1–26:18.

Apr. 2023.