CS 6115 Final Project Report

ERNEST NG and LAURA ZIELINSKI

Vision: Our goal for this project was to prove equivalence (and other properties) of Antimirov and Brzozowski derivatives, along with the zipper representation of the latter.

The Antimirov derivative of a regular expression, in complement to the Brzozowski derivative, returns a set of "partial derivatives," one of which matches the rest of the string [Krishnaswami 2013]. Intuitively, summing the elements of an Antimirov derivative is equivalent to the Brzozowski derivative.

We implemented the Antimirov derivative in Coq and proved results about its properties. Importantly, a regex matcher based on the Antimirov derivative is equivalent to that based on the Brzozowski derivative. We used this connection to prove results about Antimirov derivatives that are well-known for Brzozowski derivatives. For example, given a regular expression, there are finitely many Brzozowski derivatives for it and thus finitely many Antimirov partial derivatives. Moreover, the height (when represented as an AST) and number of Antimirov partial derivatives are linear in the height and size (number of AST nodes) of the original regex. These results were first explored for the Brzozowski derivative by [Greenberg 2020]. As far as we know, we are the first to prove them for Antimirov derivatives.

Finally, we proved a connection between Antimirov derivatives and the zipper representation of Brzozowski derivatives, the latter of which was first presented in Romain Edelmann's recent PhD dissertation [Edelmann 2021]. Edelmann discusses how a variant of the zipper data structure [Huet 1997] can be used to encode Brzozowski derivatives, and he observes in passing that zippers are "reminiscent" of Antimirov derivatives. In this final stage of our project, we proved that Edelmann's zippers and Antimirov derivatives represent the same set of regexes.

Coq Work:

- (`Regex.v`) We defined regular expressions and an inductive relation for regex matching based on code presented in class. In order to use `gsets`, or finite sets, from the `stdpp` library, we proved our regexes are countable and have decidable equality.
- (`Brzozowski.v`) We defined a function, `b_der`, which returns the Brzozowski derivative of a regex with respect to a character. We defined a derivative-based matcher, `b_matches`, and proved that it accepts the same strings as the original matching relation (`b_matches_matches`). This file also contains lemmas about the Brzozowski matcher.
- (`Antimirov.v`) Similarly, we defined a function, `a_der`, which returns the Antimirov derivative of a regex with respect to a character. We defined two matchers (`a_matches`, `a_matches'`) based on this derivative and proved their equivalence to each other (`a_matches_matches'`). We also wrote many lemmas about the Antimirov derivative, the Antimirov matcher, and properties of sets. Finally, we defined matching one regex in a set of regexes and proved that matching a regex is equivalent to matching one partial derivative (`a_der_matches_1`, `a_der_matches_2`).
- (`Equivalent.v`) We proved that the Antimirov-based matcher accepts a string if and only if the Brzozowski-based matcher accepts it as well (`a_b_matches`).
- (`Finite.v`) We defined a `gset`, `A_der`, which generates all the possible Antimirov derivatives of a regex. We then proved that the Antimirov derivative of a regex `r` with respect to any string is a subset of `A_der r` (`a_finite`). Since `gsets` are finite, this means that there are finitely many Antimirov partial derivatives of a regex.

- (`Height.v`) We defined the size (`re_size`), the number of AST nodes, and height (`re_height`), the height of the binary tree formed by the AST, of a regex. We proved that the maximum height of an Antimirov partial derivative is bounded by twice the height of the original regex (`a_deriv_height`). Finally, we proved that the size of the set of Antimirov partial derivatives of a regex with respect to a string is linear in the AST size of the original regex (`num_antimirov_derivs_linear_in_re_size`).
- (`Edelmann.v`) We adapted Edelmann's implementation of the zipper representation of Brzozowski derivatives (link) to use `gsets` (instead of lists) to represent finite sets, and extracted Edelmann's code to OCaml.
- (`ZipperAntimirov.v`) We proved that if we concatenate the elements within each *context* (list of regexes) contained in Edelmann's zipper, we get the same set of regexes as the Antimirov derivative (`zipper_antimirov_equivalent`). To establish this result, we needed to prove a lemma which describes how the zipper accumulates the result of previous derivation steps with each successive recursive call to the derivation function (`zipper_map_post_compose_concat`).

OCaml Work: We implemented executable regex matchers in OCaml based on Brzozowski, Antimirov, and zippers. (The zipper implementation is based on code extracted from Coq.) Specifically, we have two executable demos (implemented in OCaml) which demonstrate the following results using random examples generated by QuickCheck:

(1) Three regex matchers based on Brzozowski, Antimirov and zippers behave equivalently (they return the same result for the same (regex, string) pair).
(2) Antimirov derivatives and zippers represent the same set of regexes.

For efficiency purposes, our demo executables only print out the result for 15 random (regex, string) pairs. However, we also have a QuickCheck test suite which demonstrates that the aforementioned results hold for large numbers of random examples.

We also implemented smart constructors in OCaml that simplify expressions based on Kleene Algebra rules (e.g. $r \cdot \epsilon = r$) and rewrites terms so that they are in a normal form (all chains of +s are re-associated to the left, with their operands sorted in lexicographic order). These smart constructors are used in our QuickCheck generator for random regexes and our regex pretty-printer.

Activity breakdown (since checkpoint):

Laura:

- Prove that Antimirov and Brzozowski-based matchers are equivalent.
- Complete proofs that height and size of Antimirov partial derivatives are bounded.

Ernest:

- Prove that zippers and Antimirov derivatives are equivalent, and began proofs regarding height of Antimirov derivatives.
- Code for OCaml demo.

Productivity analysis: This phase of our project was extremely productive. We finished all the proofs we had started previously, eliminated ideas which proved to be too difficult, and formalized zippers using `gsets`. We accomplished all the non-stretch goals we had set during the checkpoint.

Grade: Excellent We accomplished all our goals and have created robust Coq and OCaml libraries

for derivative-based matchers. We believe many of our theorems are their first Coq formalizations, such as proofs that Antimirov and Brozozowski matchers are equivalent, Antimirov derivatives are bounded, and the relationship between zippers and Antimirov derivatives.

Future work:

- [Darragh and Adams 2020] demonstrate another way of representing Brzozowski derivatives using zippers, and they show how this alternate presentation can be used for parsing CFGs. Darragh and Adams's zippers use a representation that is more amenable to memoization. It would be interesting to prove in Coq whether the two zipper representations are equivalent. However, this extension would require modifying Darragh and Adams's zippers substantially: they do not handle Kleene star, and their AST constructors for + and · expect a list of operands as their argument (instead of just two subterms).
- We could extend our result that there are finitely many Antimirov derivatives for a regex to Brzozowski derivatives. This would require implementing a notion of regex equivalence.
- We could use Brzozowski or Antimirov derivatives to generate DFAs for regex matching. This would rely on the results that there are finitely many derivatives, up to equivalence.
- We previously attempted to define smart constructors for regexes which simplify regexes to a normal form. However, it was difficult to prove termination in Coq. If we got these proofs working, we could use this normal form to define regex equivalence.

REFERENCES

Pierce Darragh and Michael D. Adams. 2020. Parsing with zippers (functional pearl). *Proc. ACM Program. Lang.* 4, ICFP, Article 108 (Aug. 2020), 28 pages. https://doi.org/10.1145/3408990

Romain Edelmann. 2021. *Efficient Parsing with Derivatives and Zippers.* Ph. D. Dissertation. École Polytechnique Fédérale de Lausanne (EPFL). https://infoscience.epfl.ch/handle/20.500.14299/179767

Michael Greenberg. 2020. The Big Brzozowski. Blog post. https://www.weaselhat.com/post-819.html

Gérard Huet. 1997. The Zipper. *Journal of Functional Programming* 7, 5 (1997), 549–554. https://doi.org/10.1017/S0956796897002864

Neel Krishnaswami. 2013. Antimirov Derivatives for Regular Expressions. Blog post. https://semantic-domain.blogspot.com/2013/11/antimirov-derivatives-for-regular.html