The Journal of Open Source Software

MatchPy: Pattern Matching in Python

Manuel Krebber[1] **and Henrik Barthels**[1]

1 AICES, RWTH Aachen University

DOI: 10.21105/joss.00670

Software

- Review
- Repository
- Archive

Submitted: 02 February 2018
Published: 13 June 2018

Summary

Pattern matching is a powerful tool for symbolic computations. Applications include symbolic integration, term rewriting systems, theorem proving and the manipulation of abstract syntax trees. Given a pattern and an expression, the goal of pattern matching is to find a substitution for all the variables in the pattern such that the pattern becomes the expression. As an example, consider the pattern `f(x)`, where `x` is a variable, and the expression `f(a)`. Then the substitution that replaces `x` with `a` is a match. In practice, functions can also be associative and/or commutative, which makes matching more complicated and can lead to multiple possible matches.

Among existing systems, Mathematica (Wolfram Research n.d.) arguably offers the most expressive pattern matching. Unfortunately, no lightweight implementation of pattern matching as general and flexible as Mathematica exists for Python. The purpose of MatchPy (Krebber 2018; Krebber, Barthels, and Bientinesi 2017a) is to provide this functionality in Python. While the pattern matching in SymPy (Meurer et al. 2017) can work with associative/commutative functions, it does not support finding multiple matches, which is relevant in some applications. Furthermore, SymPy does not support sequence variables and is limited to a predefined set of mathematical operations.

Many-to-One Matching

In many applications, a fixed set of patterns is matched repeatedly against different subjects. The simultaneous matching of multiple patterns is called many-to-one matching, as opposed to one-to-one matching which denotes matching with a single pattern. Many-to-one matching can achieve a significant speed increase compared to one-to-one matching by exploiting similarities between patterns. MatchPy includes efficient algorithms for many-to-one matching (Krebber, Barthels, and Bientinesi 2017b), as opposed to Mathematica and SymPy.

The basic algorithms implemented in MatchPy have been described in a Master thesis (Krebber 2017).

Use in Ongoing Research

MatchPy is a central part of (Barthels 2018), an experimental tool for the automatic generation of optimized code for linear algebra problems.

References

Barthels, Henrik. 2018. “Linnea.” 2018. https://github.com/HPAC/linnea.

Krebber, Manuel. 2017. “Non-Linear Associative-Commutative Many-to-One Pattern Matching with Sequence Variables.” *CoRR* abs/1705.00907. http://arxiv.org/abs/1705.00907.

———. 2018. “MatchPy.” 2018. https://github.com/HPAC/matchpy.

Krebber, Manuel, Henrik Barthels, and Paolo Bientinesi. 2017a. “MatchPy: A Pattern Matching Library.” In *Proceedings of the 15th Python in Science Conference*. https://doi.org/10.25080/shinma-7f4c6e7-00b.

———. 2017b. “Efficient Pattern Matching in Python.” In *Proceedings of the 7th Workshop on Python for High-Performance and Scientific Computing*. https://doi.org/10.1145/3149869.3149871.

Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. “SymPy: Symbolic Computing in Python.” *PeerJ Computer Science* 3 (January):e103. https://doi.org/10.7717/peerj-cs.103.

Wolfram Research, Inc. n.d. *Mathematica*. Accessed January 19, 2018. http://www.wolfram.com/mathematica/.