Algorithmic Expression of Tensor Operations in Iversonian Languages

Lóránt Hadnagy

Department of Mathematics-Informatics, Sapientia Hungarian University of Transylvania

hadnagy.lorant@student.ms.sapientia.ro

Implementing complex tensor operations in code can result in verbose constructs that reduce readability and maintainability. Traditional programming approaches often require substantial boilerplate and manual indexing, which can make high-dimensional algorithms more challenging to express clearly and efficiently.

In the 1960s, Kenneth E. Iverson proposed a solution to this problem by introducing a new kind of mathematical notation aimed at making algorithms more readable and expressive [1]. This notation eventually evolved into the programming language APL [2], which emphasized concise syntax, consistent rules for array manipulation, and a tight coupling between code and mathematical ideas. Iverson's philosophy was that notation is a tool of thought, and he laid the groundwork for an entire family of array-oriented languages.

This talk explores how modern Iversonian languages, particularly BQN, continue this tradition and offer powerful tools for expressing tensor algorithms. BQN supports multidimensional data manipulation through features like rank polymorphism [3], scalar extension, leading axis theory, and prefix agreement [4]. It also incorporates rich functional elements, function trains [5], and combinators [6], allowing higher-order operations to be expressed succinctly and composed fluently. These principles allow programmers to operate on entire arrays with minimal syntactic noise, eliminating explicit loops and reducing cognitive overhead.

We demonstrate this with implementations of key tensor operations, including the n-mode product, n-mode unfolding, and higher-order singular value decomposition (HOSVD), showing how BQN enables concise, readable, and performant code. To quantify this, we analyze syntactic complexity using Halstead metrics, comparing BQN against MATLAB, a conventional matrix-oriented language, which offers similar high-level abstractions but does not follow Iverson's philosophy. The results suggest that Iversonian approaches offer clear advantages in both code compactness and expressive power.

We conclude by arguing that such languages are not only pedagogically valuable, but also highly effective for domains where mathematical clarity, development speed, and algorithmic reasoning are essential.

References

[1] K. E. Iverson, Notation as a tool of thought, *Commun. ACM* **23,** 8 (1980) 444–465.

[2] K. E. Iverson, *A Programming Language*, John Wiley & Sons, Inc., 1962.

[3] J. Slepak, O. Shivers, and P. Manolios, The Semantics of Rank Polymorphism, *arXiv preprint* arXiv:1907.00509 [cs.PL], 2019.

[4] R. K. W. Hui, Rank and uniformity, *SIGAPL APL Quote Quad* **25,** 4 (1995) 83–90.

[5] E. E. McDonnell and K. E. Iverson, Phrasal forms, *SIGAPL APL Quote Quad* **19,** 4 (1989) 197–199.

[6] C. Hoekstra, Combinatory logic and combinators in array languages, *Proc. 8th ACM SIGPLAN Int. Workshop on Libraries, Languages and Compilers for Array Programming (ARRAY 2022)*, San Diego, CA, USA, ACM, 2022, 46–57.