# Replacing Heuristic Rule Ordering in Symbolic Integration with Learned Policies*

Rohit John[1], Rashid Barket[2], and Matthew England[2]

[1] University of Bath, Bath, U.K.
rpj31@bath.ac.uk
[2] Coventry University, Coventry, U.K.
{barketr, Matthew.England}@coventry.ac.uk

**Abstract**

Rule-based symbolic integrators such as SymPy's `manualintegrate` apply transformation rules in a fixed order, often producing inefficient search trees. We cast rule selection as a reinforcement learning problem in which a neural policy ranks integration rules while the Computer Algebra System executes transformations and verifies correctness. Trained on only 50,000 unlabelled expressions, our agent reduces rule attempts by 77% and runtime by 80% on the RUBI benchmark set while improving accuracy over both `manualintegrate` and a supervised learning baseline trained on 2,000,000 labelled examples.

## 1 Introduction

Symbolic integration remains a challenging problem in Computer Algebra, lacking a single complete algorithmic solution. Rule-based solvers such as SymPy's `manualintegrate` rely on hand-crafted heuristics to select transformation rules, often resulting in long, inefficient derivations. By learning to prioritise promising rules, a neural model can reduce wasted rule attempts and guide the solver toward a solution more efficiently, without replacing the underlying symbolic reasoning engine. Prior work approaches rule selection via supervised behaviour cloning [5], or trains transformers to predict antiderivatives directly [2]. We cast rule selection as a reinforcement learning problem for the first time, using trajectory-level rewards to produce multiple learning signals from a single expression, enabling training on substantially fewer examples without sacrificing correctness.

## 2 Method

We model rule selection as a finite-horizon Markov Decision Process $(S, A, T, R, \gamma)$. A state $s_t$ is the current integrand encoded as a prefix-notation token sequence, with a prepended `[CLS]` token [1] as a global expression summary. The action $a_t$ is one of 24 discrete integration rules. Applying $a_t$ to $s_t$ either produces a terminal node, one or more sub-problems each becoming a new state $s_{t+1}$, or leaves the integrand unchanged on failure, advancing to the next rule in the ranked list. Rewards are assigned per attempted rule: $r_t = -0.05$ for a successful application and $r_t = -0.2$ for a failed attempt. At episode termination, successful-action rewards are post-scaled by an outcome-dependent factor $\lambda(o_T) \in \{-15, +15, +10\}$ for correct, incorrect, and `DontKnowRule` outcomes respectively; since successful-step rewards are negative, $\lambda = -15$ makes them positive for correct episodes while the remaining values amplify the penalty, prioritising correctness over short but wrong derivations. The policy maximises expected discounted return $J(\theta) = \mathbb{E}\big[\sum_n \gamma^n \tilde{r}_n\big]$.

*Code and models are available at https://github.com/rohitpj/RL_for_Integration.

**RL architecture.** Both actor and critic are built on an encoder-only Transformer: the actor produces logits over the 24 rules from the CLS representation while the critic produces a scalar value estimate. Parameters are optimised using the PPO algorithm [4].

**Supervised baseline.** Following the approach of [5], we train an encoder-only Transformer of identical architecture to our RL model to classify expression–rule pairs. A linear head over the CLS representation produces a softmax distribution over rules, optimised via cross-entropy loss to imitate SymPy's rule selection.

**Data.** The RL model was trained on 50,000 unlabelled expressions generated by the FWD method of [2], and the supervised baseline was trained on the SIRD dataset, a collection of 2,000,000 labelled rule decisions extracted from SymPy integration traces.

# 3 Results

We evaluate all three methods on four datasets: held-out FWD (in-domain), IBP and BWD (out-of-domain), and the RUBI independent test suite [3] as an external benchmark. We compare the RL model against SymPy's `manualintegrate` (*Manual*) and our supervised behaviour-cloning baseline across accuracy (correct integrations), total rule attempts (*steps*), number of branches (successful rule applications), and average runtime.

Table 1: Integration results across datasets.

| Dataset | Method | Accuracy (%) | # Steps | # Branches | Avg. Time (s) |
|---|---|---|---|---|---|
| FWD | Manual | 95.7 | 101.5 | 22.5 | 0.44 |
| | Supervised | 94.8 | 70.2 | 10.1 | 0.22 |
| | RL | **96.3** | **53.9** | **8.3** | **0.19** |
| IBP | Manual | 91.7 | 142.9 | 28.4 | 0.60 |
| | Supervised | 92.1 | 54.8 | 10.9 | 0.24 |
| | RL | **94.5** | **51.2** | **7.5** | **0.22** |
| BWD | Manual | 41.3 | 271.3 | 41.1 | 1.05 |
| | Supervised | 60.5 | 177.0 | 23.4 | 0.77 |
| | RL | **69.7** | **75.9** | **9.6** | **0.33** |
| RUBI | Manual | 39.8 | 209.6 | 37.7 | 1.49 |
| | Supervised | 43.9 | 81.5 | 11.5 | 0.41 |
| | RL | **44.4** | **62.1** | **8.2** | **0.30** |

# 4 Conclusion

We introduce an RL-based rule selection policy for symbolic integration within SymPy's `manualintegrate` framework, formulating rule selection as a finite-horizon MDP with trajectory-level rewards encouraging correctness and efficiency. Our method requires no labelled intermediate rule supervision, achieves higher accuracy while substantially reducing attempted rules and runtime across all datasets and demonstrates strong out-of-domain generalisation.

# References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186. Association for Computational Linguistics, June 2019.

[2] G. Lample and F. Charton. Deep learning for symbolic mathematics. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[3] A. Rich, P. Scheibe, and N. Abbasi. Rule-based integration: An extensive system of symbolic integration rules. *Journal of Open Source Software*, 3(32):1073, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[5] V. Sharma, A. Nagpal, and M. F. Balin. SIRD: Symbolic integration rules dataset. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023.