Replacing Heuristic Ordering with Learned Policies for Symbolic Integration

Rohit John, Rashid Barket, Matthew England
University of Bath, University of Coventry

Symbolic Integration

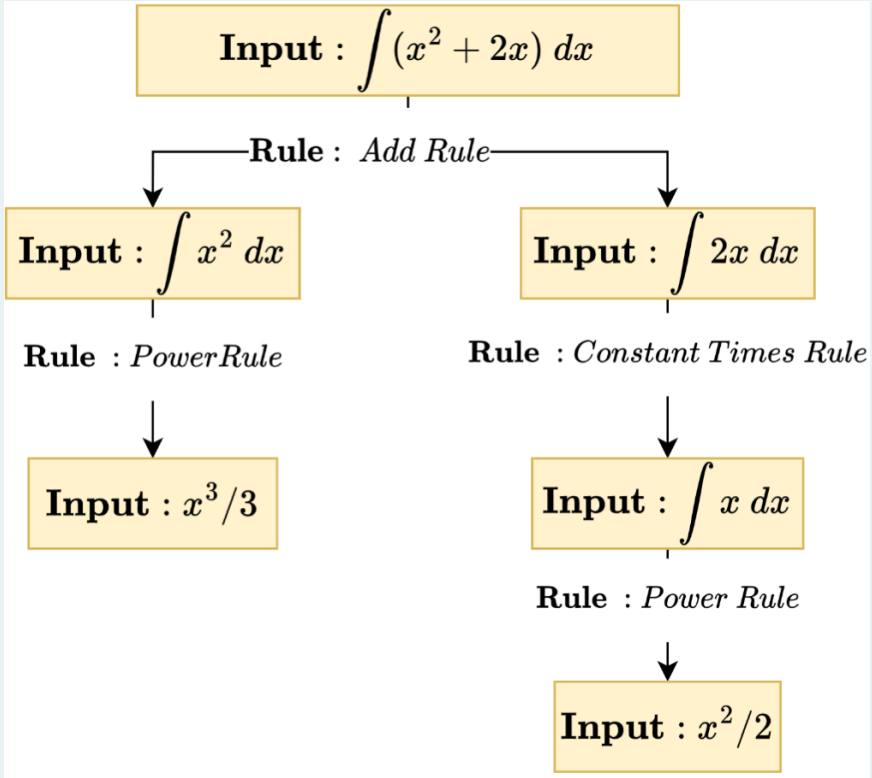
- A problem with applications across physics, engineering, and applied mathematics.
- Unlike differentiation, integration has no single algorithm that solves every case
- Generates the exact antiderivative in closed form (e.g. $\int 2x\,dx = x^2 + C$)

Challenges with Symbolic Integration

- Current solvers use either partial implementations of the Risch Algorithm or rule-based methods, both with drawbacks.
- Risch Algorithm has never been fully implemented and answers lack explainability.
- Rule-based methods imitate human integration techniques and are inefficient and struggle with more difficult integrals.

Rule based solvers & manualintegrate

- Rule-based solvers repeatedly apply transformation rules to arrive at an answer.
- SymPy's manualintegrate works recursively, splitting the integrand into sub-problems, and applying new rules to them.
- At each step it uses a fixed heuristic order, taking the first valid rule



Related work

- Lample & Charton (2020): First to use ML to improve integration, directly generated answers but generalised poorly and lacked correctness guarantees.
- Sharma et al. (2023): Trained classifier to imitate manualintegrate, requires 2M labelled pairs with minimal performance increase.
- Peifer et al. (2020): RL for S-pair selection in Buchberger's algorithm, demonstrated RL based models can beat expert heursitics.

RL for Symbolic Integration

- We can frame rule selection as a sequential decision-making problem and train an agent to select promising rule using RL.
- The agent learns a policy that ranks candidate rules; SymPy executes the chosen rule and verifies the result by differentiation
- Reward reflects whether the rule is valid and whether the integration finishes correctly and efficiently, so the agent is pushed toward correct, short derivations

Markov Decision Process formulation

- State: current integrand
- Actions: 24 discrete integration rules
- Transition: apply rule → either terminal (evaluable) or one/more sub-problems to integrate; failed rule leaves integrand unchanged, move to next ranked rule
- Reward: invalid rules attempts receive a small penalty, invalid answers return a large one.

Example, on $\int(x^2 + 2x)\,dx$:

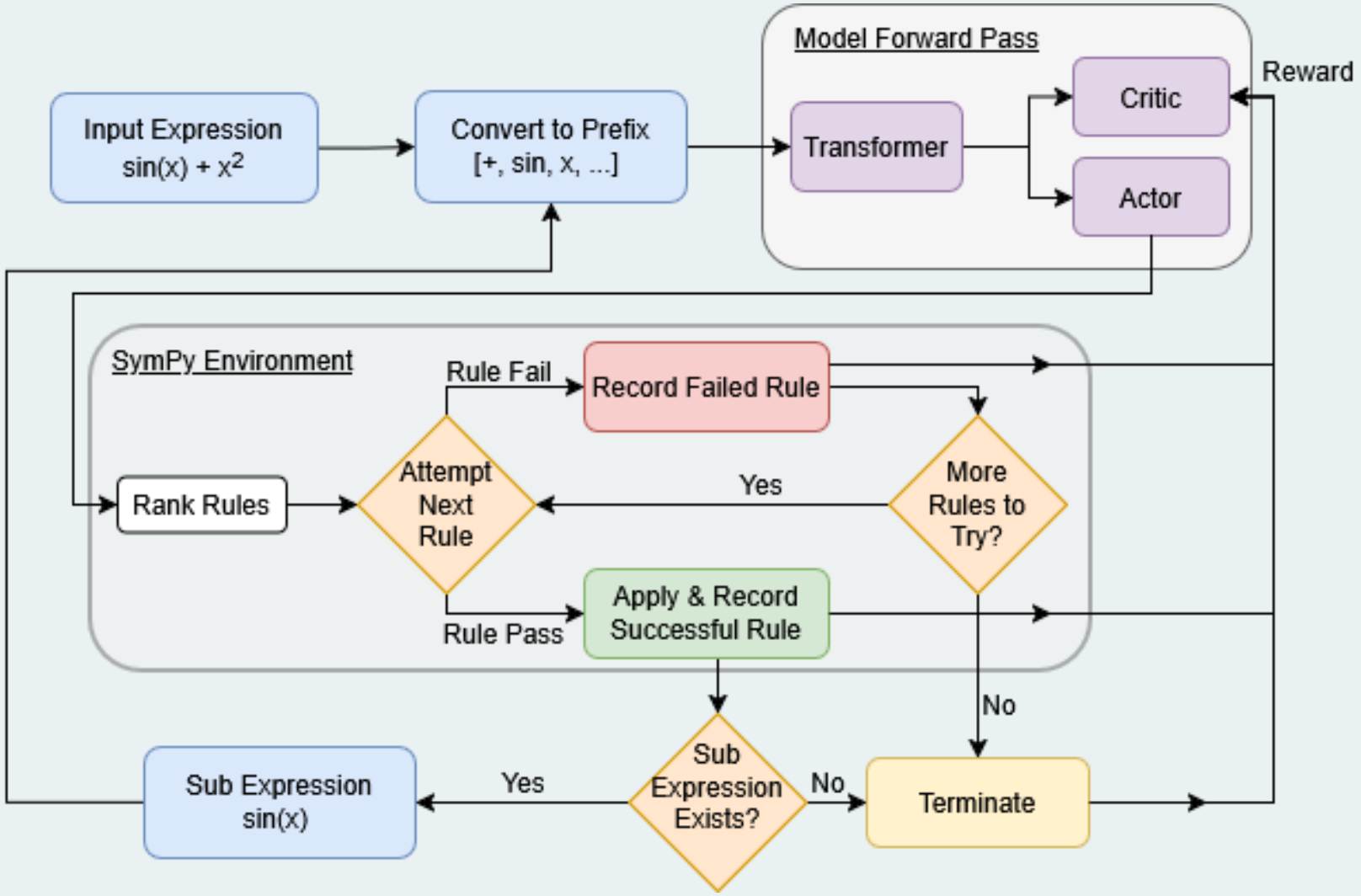
Valid: the add rule matches a sum → splits into $\int x^2\,dx$ and $\int 2x\,dx$ & receives reward.

Invalid: the trig rule finds nothing to match → returns nothing, agent receives penalty.

Model Architecture

- Encoder-only transformer over the tokenised expression
- Actor: outputs scores over the 24 rules → the ranking
- Critic: estimates the value of a state (expected return), used to stabilise learning
- PPO used to update actor/critic based on previous rewards
- Prefix notation used to improve efficiency.

Model Architecture



Supervised baseline

- Based on Sharma et al. (2023): learns to imitate SymPy's rule choice
- Trained by cross-entropy against the rule chosen by SymPy's ordering heuristic.
- At inference, ranks rules by predicted probability; SymPy attempts them in that order until one applies
- Requires 2M labelled expressions, effectively performs behaviour cloning.

Data & training

- FWD generation method (Lample & Charton 2020) produces expressions guaranteed to be elementarily integrable.
- RL agent: 50,000 generated expressions, no labels
- Supervised baseline: SIRD dataset (Sharma et al. 2023), ~2M expression–rule pairs from SymPy traces
- Training:

  Supervised - cross-entropy against SymPy's chosen rule at each state; one pass over the labelled pairs

  RL - agent integrates expressions, collects rewards over full trajectories, and updates via PPO; each expression yields many rule-selection decisions.

Results (FWD, IBP & BWD)

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**

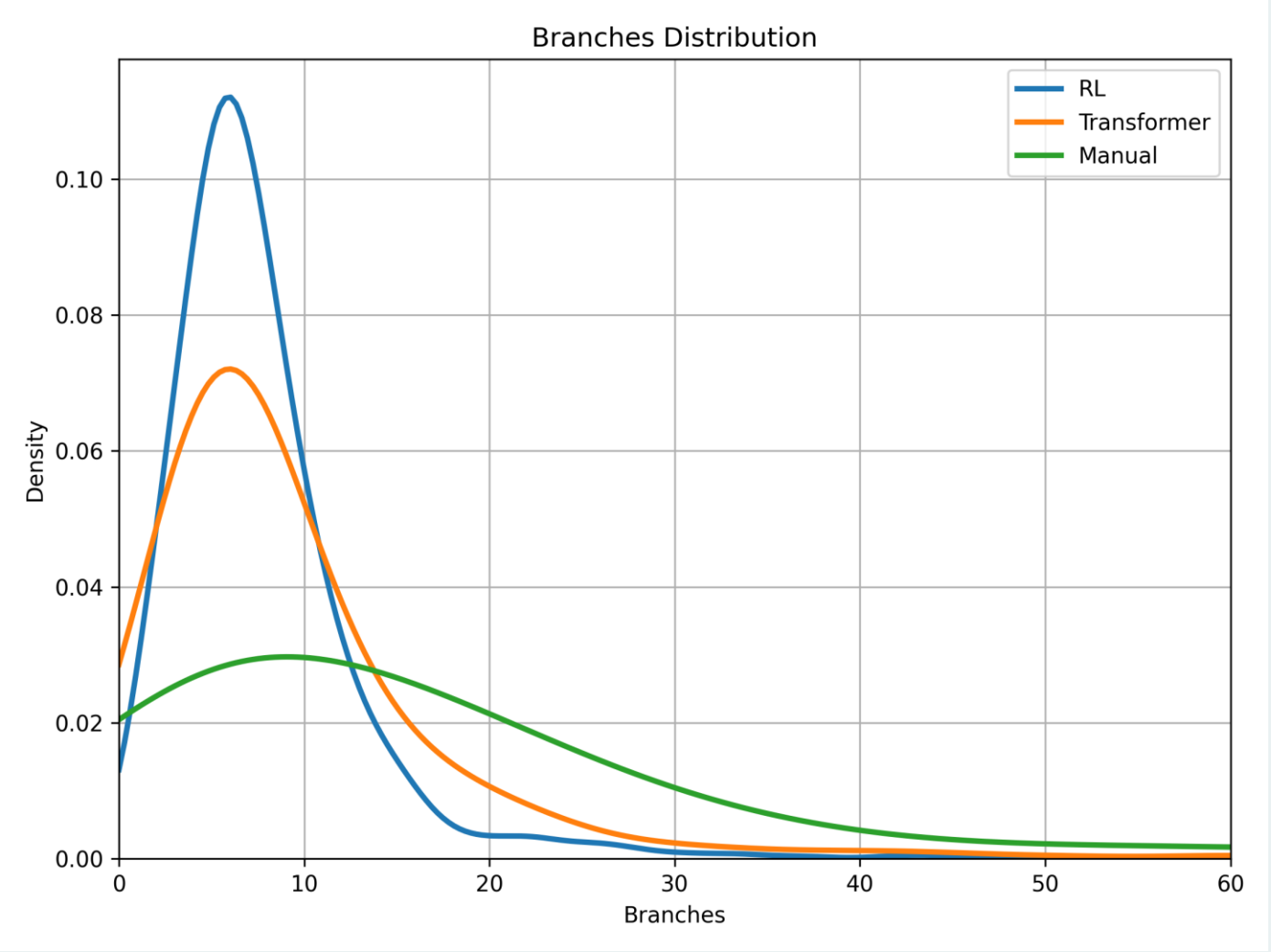
External generalisation

- RUBI: rule-based integrator with 6,700+ rules; independent test suite of 1,832 textbook/paper problems
- All three methods score under 50%, constrained by 24 rules vs RUBI's 6,700

Method	Correct (%)	Avg Steps	Avg Branches	Avg Time (s)
Manual	39.84	209.62	37.71	1.49
Supervised	43.93	81.52	11.54	0.41
RL	**44.43**	**62.06**	**8.18**	**0.30**

Branching efficiency

- Measures number of times integration function is recursively called, correlates with time taken and resources consumed.
- RL requires 76.6% less function calls than manualintegrate.



Example problems

Function	Manual Output	RL Output
$x\left(3+2x+\frac{x}{\sqrt{x^2+1}}\right)$	$\frac{2x^3}{3}+\frac{3x^2}{2}+\int\frac{x^2}{\sqrt{x^2+1}}\,dx$	$\frac{2x^3}{3}+\frac{3x^2}{2}+\frac{x\sqrt{x^2+1}}{2}-\frac{\operatorname{asinh}(x)}{2}$
$2x+e^{\sqrt{6}\sqrt{x}}$	$x^2+\int e^{\sqrt{6}\sqrt{x}}\,dx$	$\frac{\sqrt{6}\sqrt{x}e^{\sqrt{6}\sqrt{x}}}{3}+x^2-\frac{e^{\sqrt{6}\sqrt{x}}}{3}$
$\frac{1}{x^{1/2}}\left(\tan(x^{1/2})^3+2x^{1/2}+\tan(x^{1/2})\right)$	$2x+\int\frac{\tan(\sqrt{x})}{\sqrt{x}}\,dx+\int\frac{\tan(\sqrt{x})^3}{\sqrt{x}}\,dx$	$2x+\log\left(\csc(\sqrt{x})^2-1\right)-2\log\left(\cos(\sqrt{x})\right)-\log\left(\csc(\sqrt{x})^2\right)+\frac{1}{\csc(\sqrt{x})^2-1}$

Limitations

- 24-rule action space caps accuracy on RUBI-level problems, where the solver has 6,700+ rules available
- Scaling to 6,700+ actions is an open problem for RL: sparse coverage of the action space and slow convergence
- Training and evaluation so far use FWD-generated data; richer generation methods address known weaknesses in how expressions are generated

Conclusion & Future Work

- We formulate rule selection in Symbolic Integration as a sequential decision-making process and optimised it using RL.
- Our agent achieves superior performance to previous models, while requiring significantly fewer data points and zero labelling.
- More broadly we show how Computer Algebra Systems can be optimised with RL without sacrificing correctness.
- Future work may extend the rule set (currently 24), train on larger datasets or apply the demonstrated framework to a new problem.

Thank you