Санкт-Петербургский государственный университет

Кафедра системного программирования

Artificial Intelligence Guided Symbolic Execution

Максим Владиславович Нигматулин, 22.M07-мм

Научный руководитель: д. ф.-м. Д.А. Мордвинов, доцент кафедры системного программирования
Консультант: к.ф.-м. С.В. Григорьев, доцент кафедры информатики

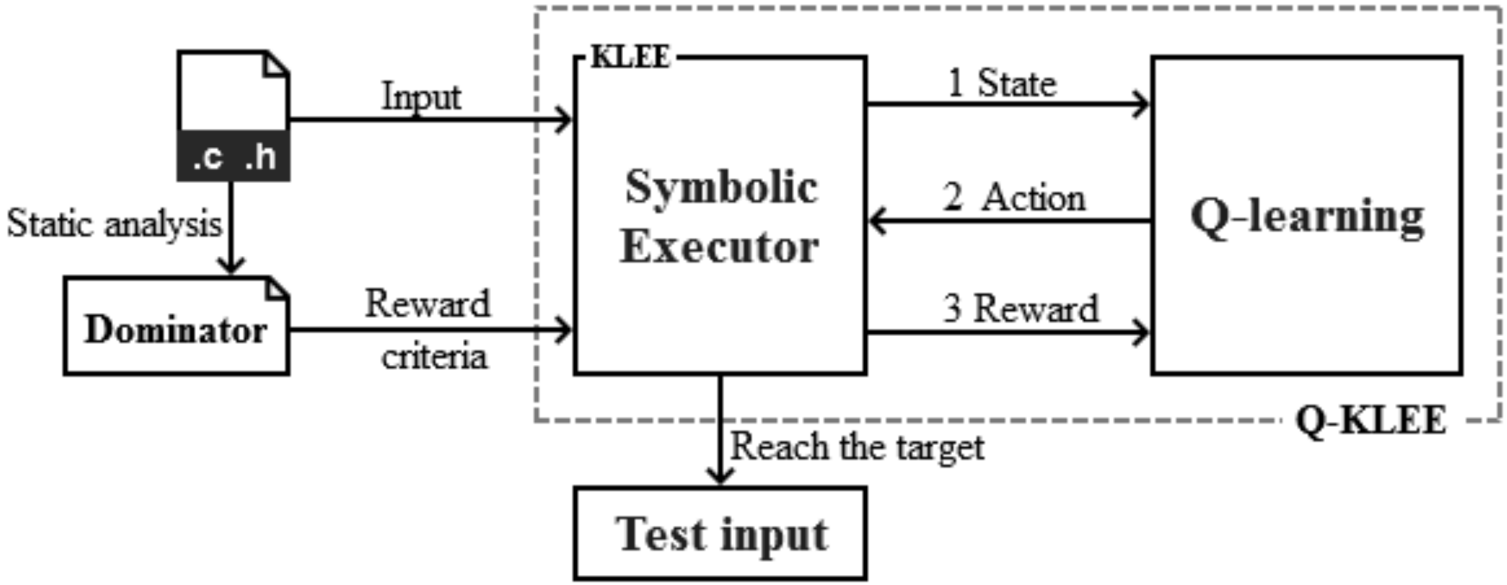
Санкт-Петербург
2023

Введение

- Символьное исполнение — техника анализа ПО, позволяющая понять, какие данные вызывают выполнение каждой части программы
- Одна из проблем — “взрыв” путей, которые нужно исследовать

Существующие решения: Q-KLEE[1]

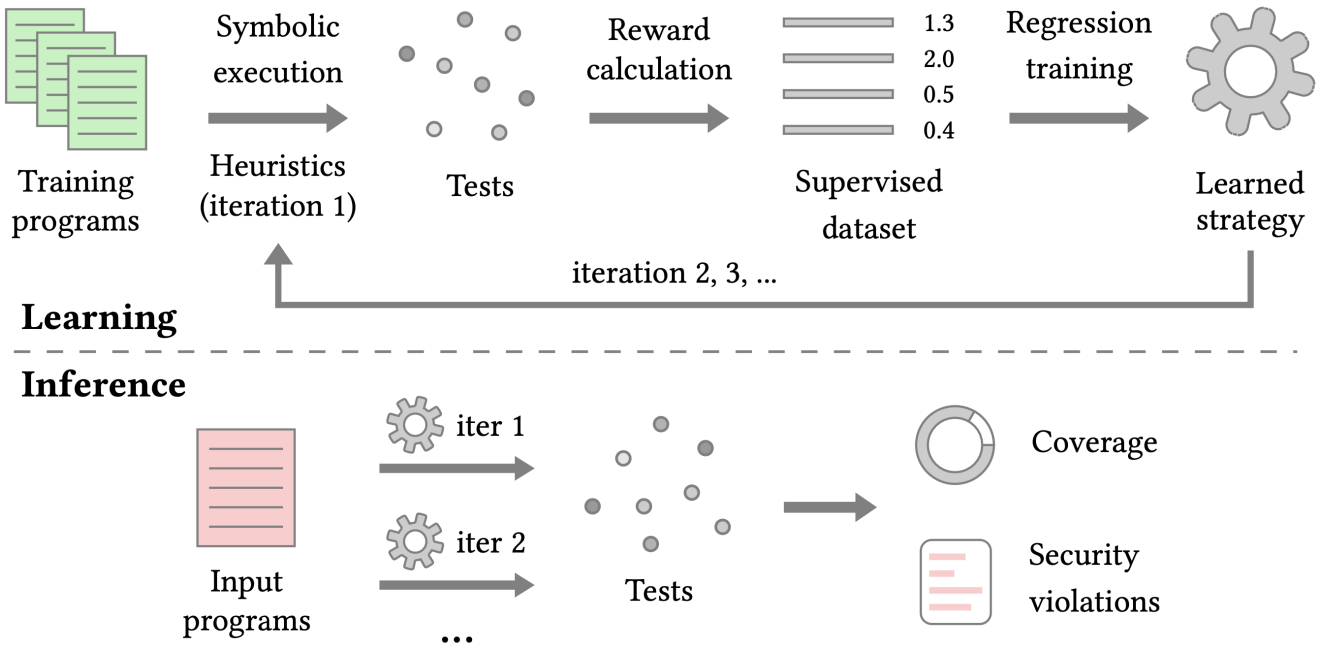
Согласно бенчмаркам, исследует в 10 раз меньше путей, исполняет в 10 раз меньше инструкций за незначительно большее время



[1] J. Wu, C. Zhang and G. Pu, "Reinforcement Learning Guided Symbolic Execution,"2020

Существующие решения: Learch[2]

- KLEE as symbolic execution engine
- Возможность обучать свои модели
- Возможность генерировать датасет на своих данных



[2] Jingxuan He, Gishor Sivanrupan, Petar Tsankov, and Martin Vechev, "Learning to Explore Paths for Symbolic Execution"

Существующие решения: PettingZoo

- Мультиагентное обучение с подкреплением
- Поддержка нескольких игровых окружений
- Запуск параллельно почти из коробки

Существующие решения

- Q-KLEE — можно улучшить
- Learch — не позвояет работать с Reinforcement Learning
- PettingZoo — сложно портировать

Постановка задачи

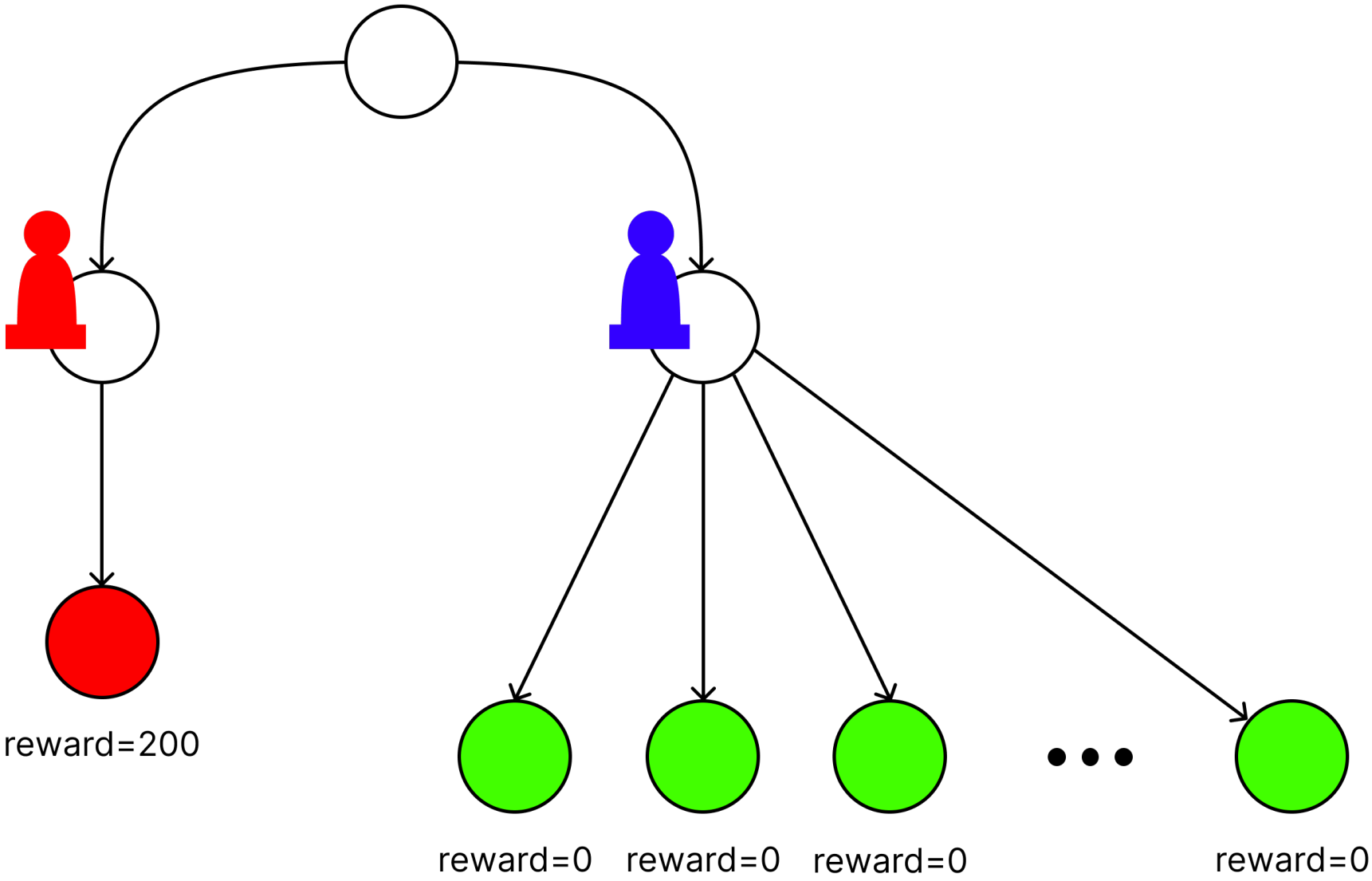
Цель работы: реализовать среду для обучения с подкреплением при взаимодействии с V# как игровой средой

Поставленные задачи:

- Поддержать протокол общения с V#
- Запустить обучение
- Распараллелить обучение

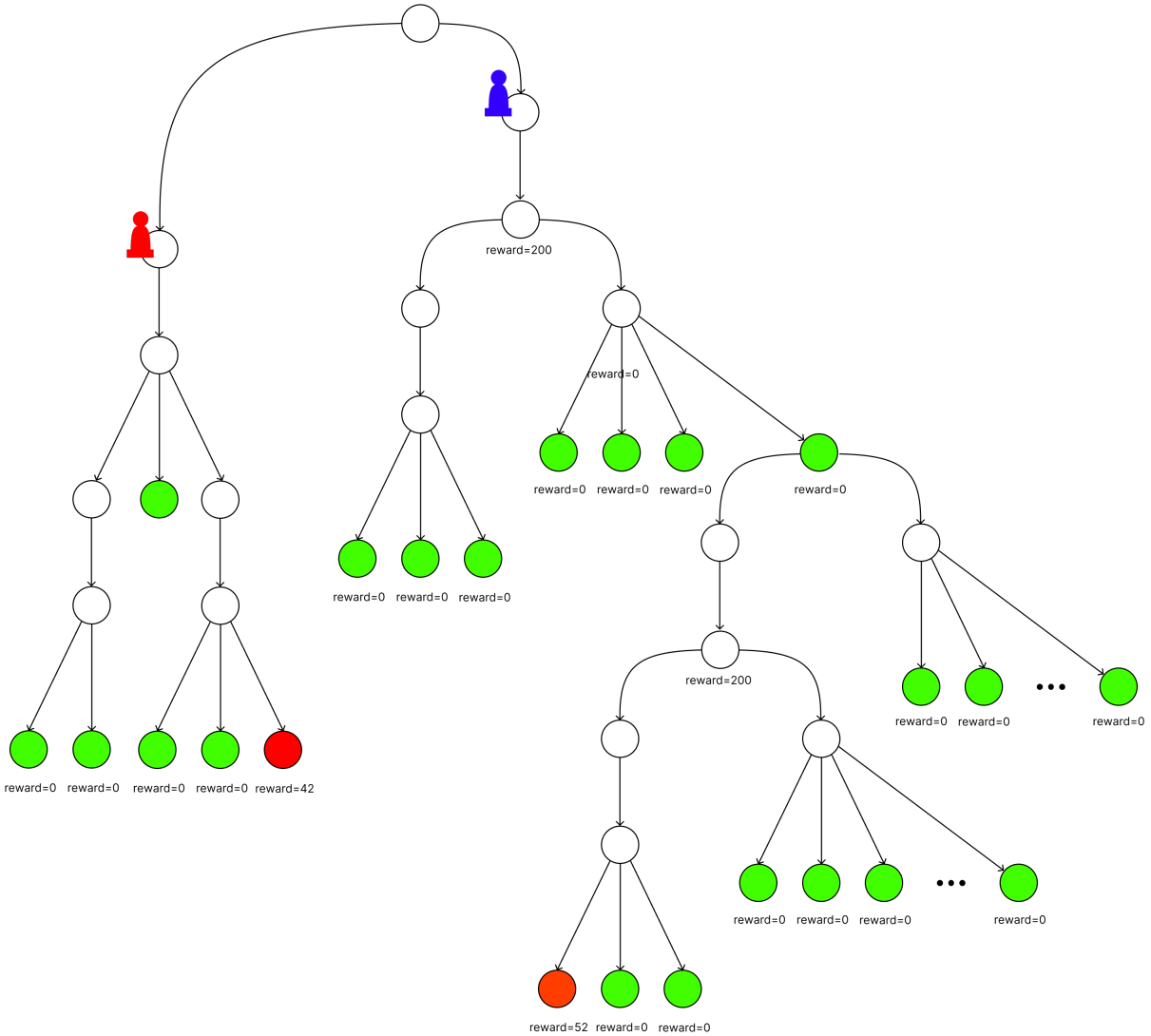
Игровая аналогия

Какую фишку подвинуть?



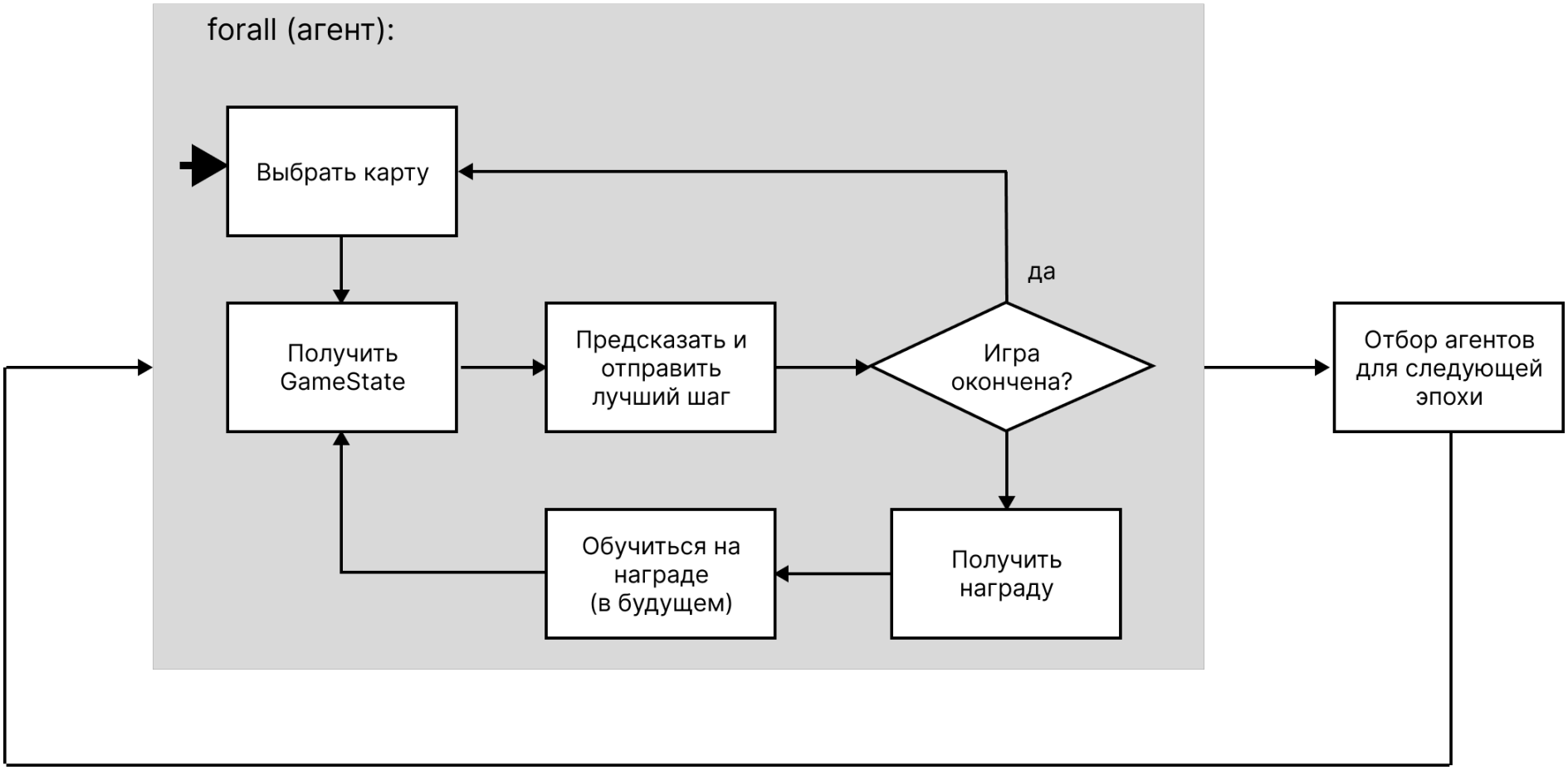
Игровая аналогия

Какую фишку подвинуть?

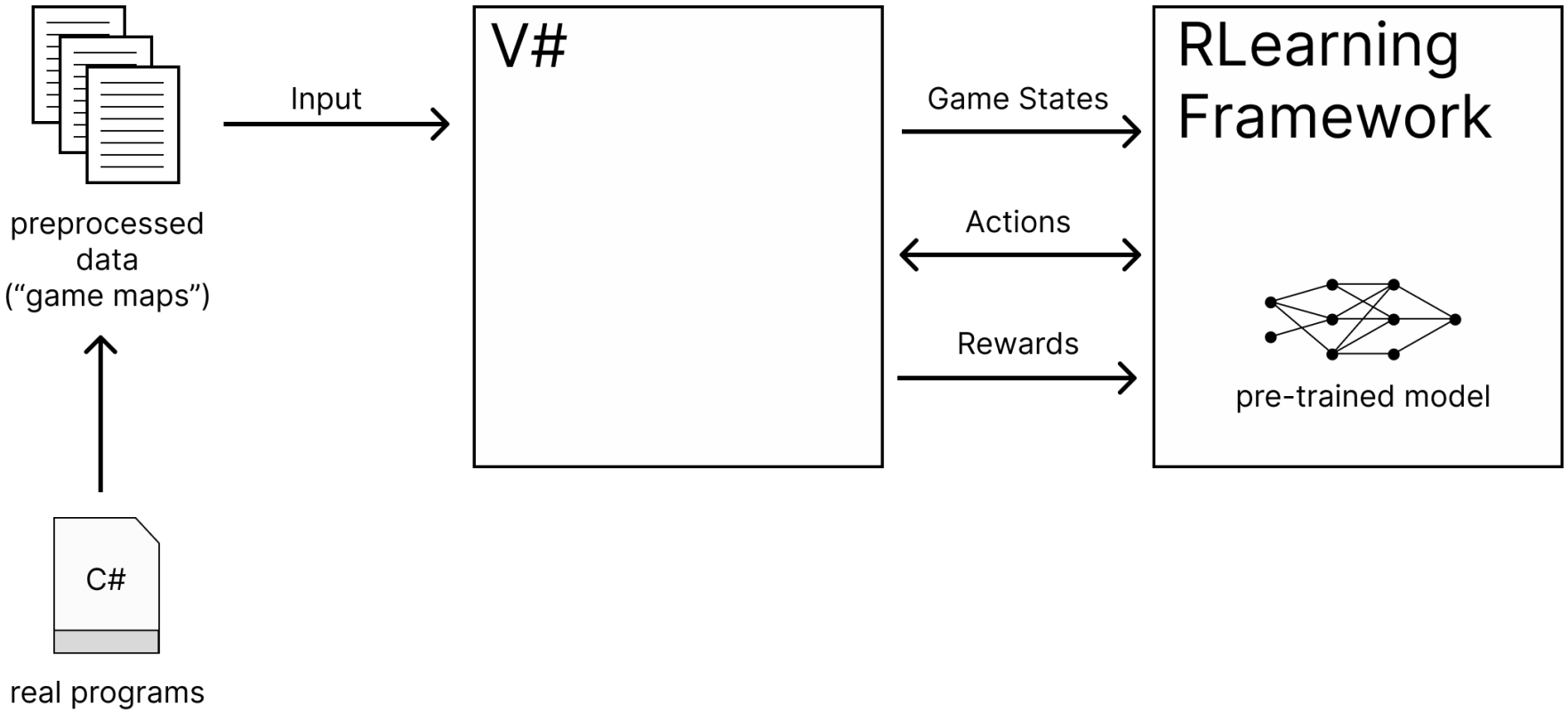


Процесс обучения

Эпоха

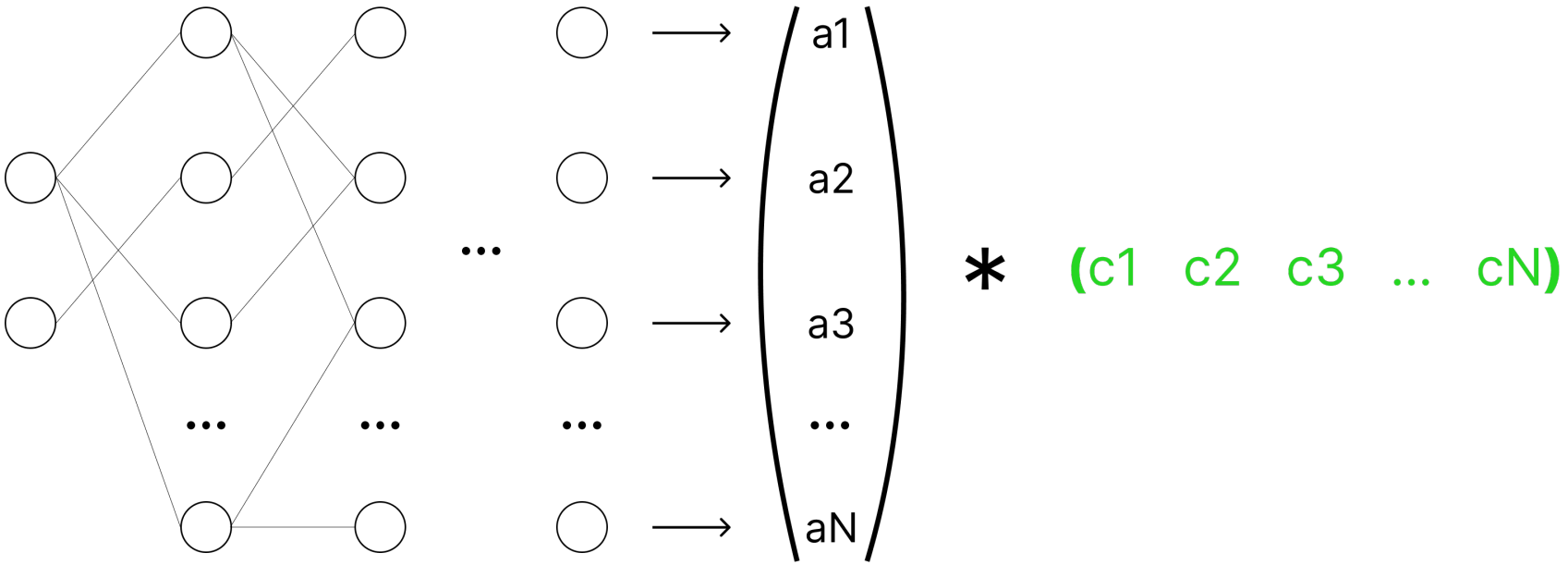


Структура решения



Обучение

- Вектор из весов, умножается на выходные данные нейронной сети
- Состояния ранжируются по сумме компонентов

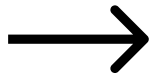


Обучение: дополнительный слой

Положительная динамика наблюдается, но об однозначных результатах говорить рано

Epoch 1/20

	BinarySearch_Method_0
agent1	coverage %: 100.00, steps: 352
agent2	coverage %: 88.24, steps: 600
agent3	coverage %: 100.00, steps: 362
agent4	coverage %: 100.00, steps: 278
agent5	coverage %: 64.71, steps: 600
agent6	coverage %: 70.59, steps: 600
agent7	coverage %: 64.71, steps: 600
...	...

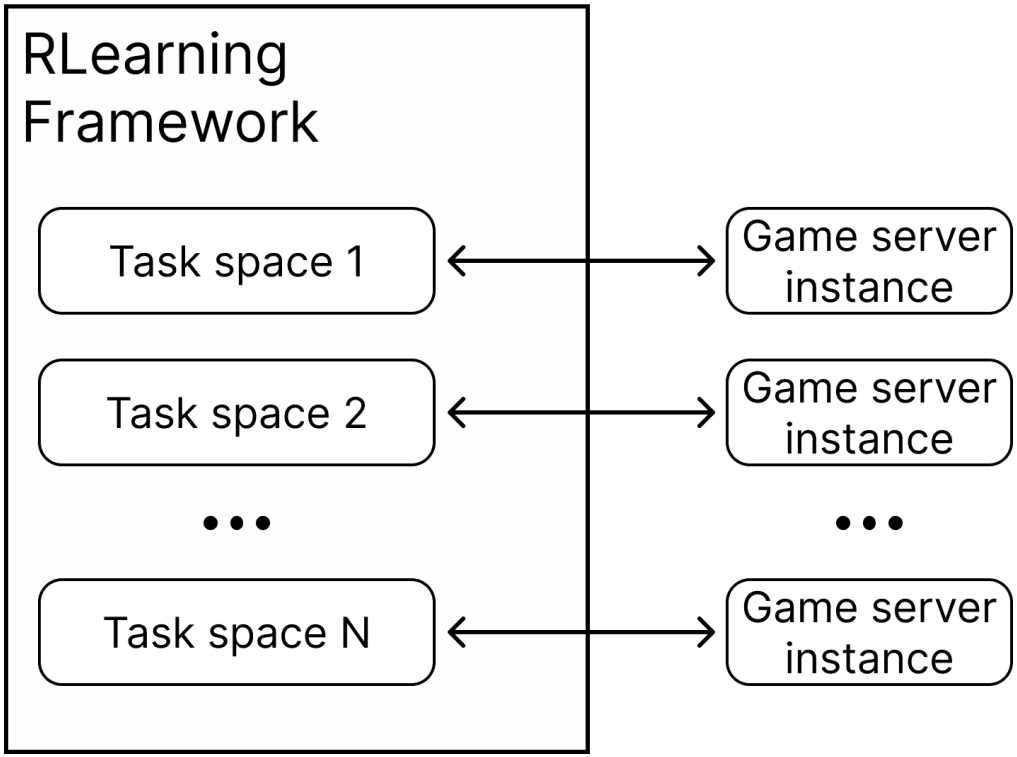


Epoch 3/20

	BinarySearch_Method_0
agent1	coverage %: 100.00, steps: 353
agent2	coverage %: 100.00, steps: 284
agent3	coverage %: 100.00, steps: 338
agent4	coverage %: 100.00, steps: 284
agent5	coverage %: 100.00, steps: 355
agent6	coverage %: 100.00, steps: 346
agent7	coverage %: 82.25, steps: 600
...	...

Параллелизм: CPU

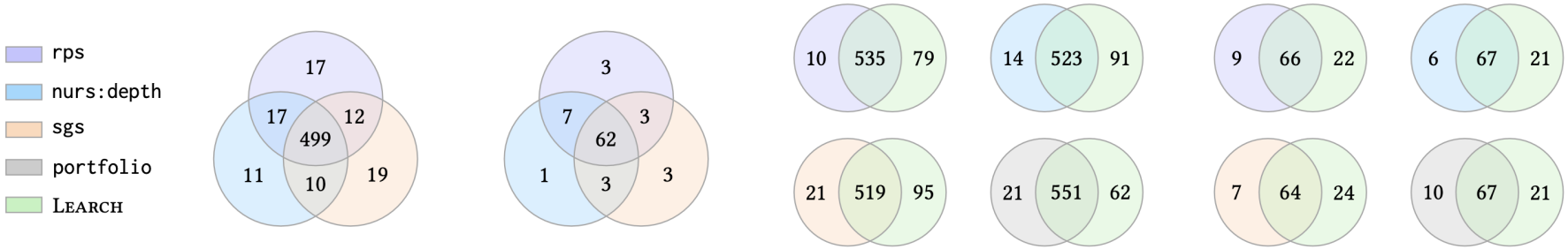
- “Игры” в одном поколении передаются в пул потоков для обработки
- Взаимодействие с несколькими игровыми серверами
- Работает быстрее



Результаты

- Реализован протокол передачи данных с V# через сокеты
- Запущено обучение
- Реализован параллелизм на CPU

ML > эвристики[3]



(a) Average line coverage by heuristics.

(b) UBSan violations found by heuristics.

(c) Average line coverage: Learch v.s. heuristics.

(d) UBSan violations: Learch v.s. heuristics.

Figure 3: Limitations of existing manually designed heuristics and how Learch outperforms them for our `coreutils` test set.

[3] Jingxuan He, Gishor Sivanrupan, Petar Tsankov, and Martin Vechev, "Learning to Explore Paths for Symbolic Execution"