Эвристические стратегии редукции термов в задачах формальной верификации

Трилис Алексей Андреевич
научный руководитель: А.М.Ляшин

НИУ ВШЭ — Санкт-Петербург

9 июня 2025 г.

Введение

- **Формальная верификация** — доказательство корректности программ
- **Смарт-контракты** — подходящая область для верификации
- **Coq**[1] — популярный инструмент для формальной верификации
- **Ursus**[2] — фреймворк для верификации смарт-контрактов, построен на Coq

[1] The Coq Development Team, *The Coq Proof Assistant*, URL: https://rocq-prover.org/doc/v8.16/refman/.

[2] Pruvendo, *Ursus language documentation*, URL: https://ursus-lang.dev/.

Фреймворк Ursus

Входные данные

```
uint64 result;
function f(uint64 a, uint64 b) {
    uint64 x = result;
    x = x + a;
    x = x - b;
    result = x;
}
```

```
y1 = стартовое состояние системы
y2 = указатель на глобальную переменную result
y3 = y1, в котором создана новая переменная x со значением y2
y4 = указатель на состояние переменной x в y3
y5 = значение указателя y4
y6 = y5 + a
y7 = y3, в котором значение переменной x заменено на y6
y8 = указатель на состояние переменной x в y7
y9 = значение указателя y8
y10 = y9 - b
y11 = y7, в котором значение переменной x заменено на y10
y12 = указатель на состояние переменной x в y11
y13 = значение указателя y12
y14 = y11, в котором значение глобальной переменной result заменено на y13
y15 = y14 (итоговое состояние системы)
```

- Система уравнений Y вида $y_i = T_i(y_1, y_2, \ldots, y_{i-1})$
- Спецификация $P(y_1, y_2, \ldots, y_n)$

Нужно упростить P, подставив все y_i и вычислив результат

Редукции в Coq

Редукция	Порядок	Достоинства	Недостатки
`cbv`	call by value	Нет дубликации вычислений	Мёртвый код Неэффективность реализации[3]
`lazy`	call by need	Мёртвый код не вычисляется Мемоизация вычислений	Дубликация вычислений[4] Неэффективность реализации[3]
`native_compute`[5]	call by value	Эффективность	Вычисляет до конца[6] Не подходит для символьного вычисления
`vm_compute`[7]	call by value	Эффективность	Вычисляет до конца[6] Не подходит для символьного вычисления

[3] Gross, «Performance Engineering of Proof-Based Software Systems at Scale».

[4] `https://github.com/rocq-prover/rocq/issues/18520`

[5] Boespflug, Dénès и Grégoire, «Full Reduction at Full Throttle».

[6] `https://github.com/rocq-prover/rocq/issues/4476`

[7] Grégoire и Leroy, «A compiled implementation of strong reduction».

Цель и задачи

Цель: Оптимизация символьного вычисления результата функции в контексте системы Ursus.
Задачи:

- Разработать несколько стратегий вычисления, используя классические порядки редукций и эвристики, продиктованные структурой специфических для Ursus данных.
- Подготовить набор программ на Ursus для использования в качестве тестовых данных.
- Сравнить разработанные стратегии на тестовом наборе и выявить среди них наиболее производительные.

Базовые стратегии

native

$(Y, P) \leftarrow$ input
$F(x) \leftarrow x$
for $i \leftarrow 1$ to $|Y|$ **do**
  $F(x) \leftarrow F(\texttt{let } x_i = T_i \texttt{ in } x)$
  for $j \leftarrow i + 1$ to $|Y|$ **do**
    $T_j \leftarrow T_j[y_i := x_i]$
return $reduce(F(P))$

topdown

$(Y, P) \leftarrow$ input
for $i \leftarrow 1$ to $|Y|$ **do**
  $T_i \leftarrow reduce(T_i)$
  for $j \leftarrow i + 1$ to $|Y|$ **do**
    $T_j \leftarrow T_j[y_i := T_i]$
  $P \leftarrow P[y_i := T_i]$
return $reduce(P)$

bottomup

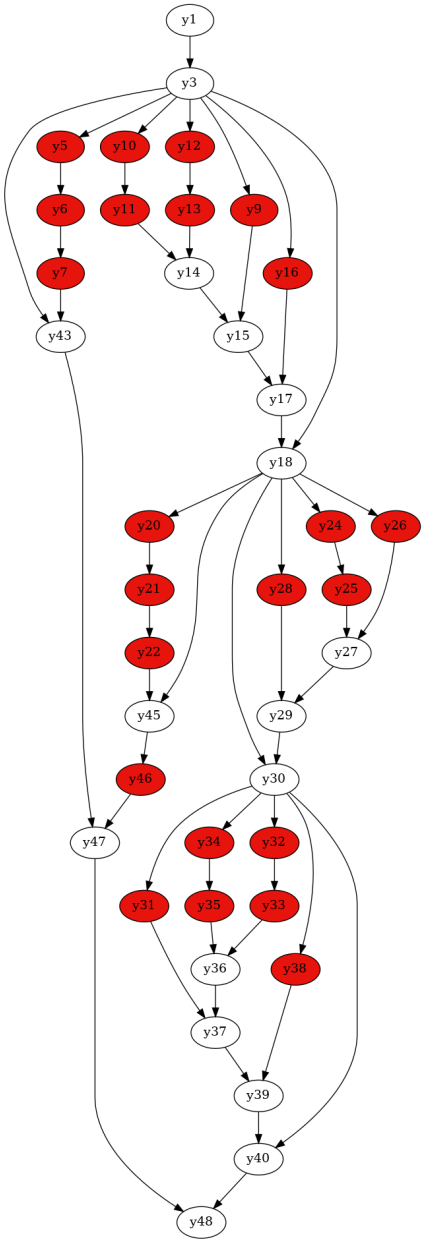
$(Y, P) \leftarrow$ input
for $i \leftarrow |Y|$ to 1 **do**
  $P \leftarrow P[y_i := T_i]$
return $reduce(P)$

bottomup-reductions

$(Y, P) \leftarrow$ input
for $i \leftarrow |Y|$ to 1 **do**
  $T_i \leftarrow reduce(T_i)$
  $P \leftarrow P[y_i := T_i]$
return $reduce(P)$

- Две версии каждой стратегии, с `cbv` и `lazy`
- Перед каждой стратегией удаляем y_i, не использующиеся в P

Графовые эвристики 1



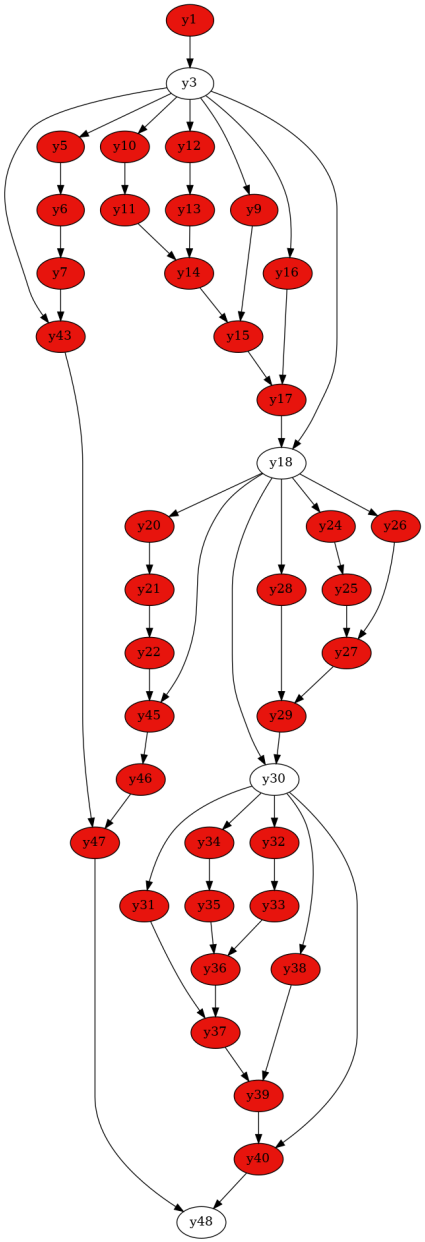
- Стягивание вершин = подстановка термов
- Много вершин с outdeg = indeg = 1
- Можно их стянуть

---

contractions

$(Y, P) \leftarrow$ input
$T \leftarrow P(y_0, y_n)$
for $i \leftarrow 1$ to $|Y|$ **do**
  if $indeg(y_i) = 1$ and $outdeg(y_i) = 1$ **then**
    for $j \leftarrow i + 1$ to n **do**
      $T_j \leftarrow T_j[y_i := T_i]$
    $P \leftarrow P[y_i := T_i]$
    $Y \leftarrow Y \setminus (y_i, T_i)$
return $basic\_strategy(Y, P)$

Графовые эвристики 2



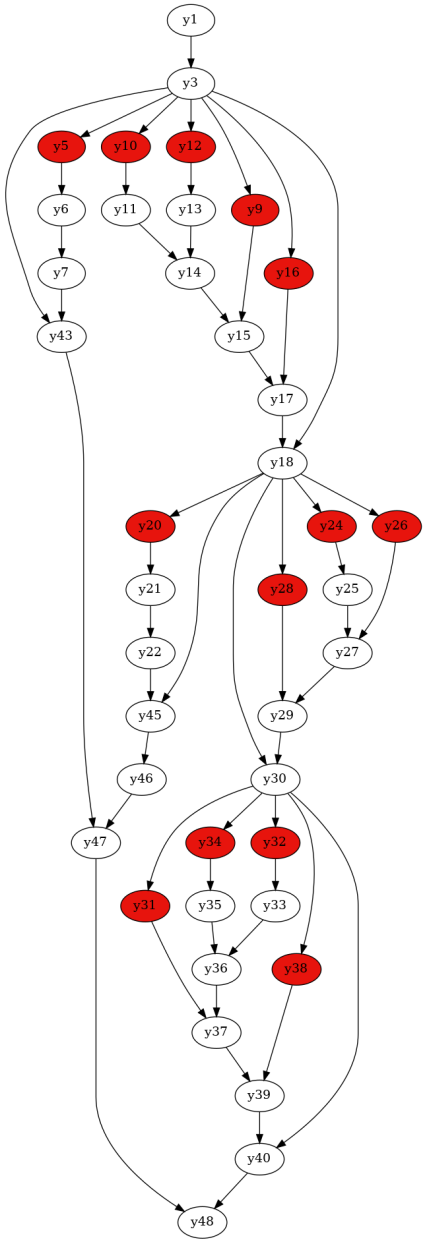
- Почти все вершины с outdeg = 1
- Можно их стянуть
- От этого возникнет дубликация, но не очень большая

---

contractions-strong

$(Y, P) \leftarrow$ input
for $i \leftarrow 1$ to $|Y|$ **do**
  if $outdeg(y_i) = 1$ **then**
    for $j \leftarrow i + 1$ to n **do**
      $T_j \leftarrow T_j[y_i := T_i]$
    $P \leftarrow P[y_i := T_i]$
    $Y \leftarrow Y \setminus (y_i, T_i)$
return $basic\_strategy(Y, P)$

Эвристики на основе типов данных 1



- Определять indeg и outdeg сложно и долго
- Воспользуемся информацией о типах
- Все вершины типов LocalStateMapping[a] и FieldType[b] имеют indeg = outdeg = 1

---

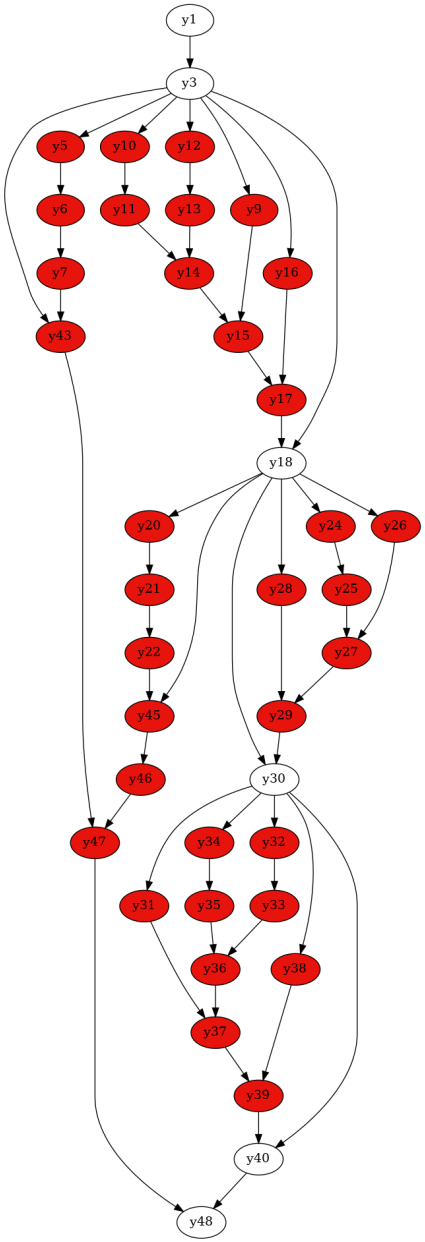
contractions-typebased

$(Y, P) \leftarrow$ input
for $i \leftarrow 1$ to $|Y|$ **do**
  if $typeof(y_i)$ is *LocalStateMapping* or *FieldType* **then**
    for $j \leftarrow i + 1$ to n **do**
      $T_j \leftarrow T_j[y_i := T_i]$
    $P \leftarrow P[y_i := T_i]$
    $Y \leftarrow Y \setminus (y_i, T_i)$
return $basic\_strategy(Y, P)$

---

[a] состояние локальных переменных
[b] проекции структур

Эвристики на основе типов данных 2



- Вершины типа Ledger[a], как правило, имеют большой outdeg
- Почти все вершины других типов имеют outdeg = 1

---

contractions-strong-typebased

$(Y, P) \leftarrow$ input
for $i \leftarrow 1$ to $|Y|$ **do**
  if $typeof(y_i)$ is not $Ledger$ **then**
    for $j \leftarrow i + 1$ to n **do**
      $T_j \leftarrow T_j[y_i := T_i]$
    $P \leftarrow P[y_i := T_i]$
    $Y \leftarrow Y \setminus (y_i, T_i)$
return $basic\_strategy(Y, P)$

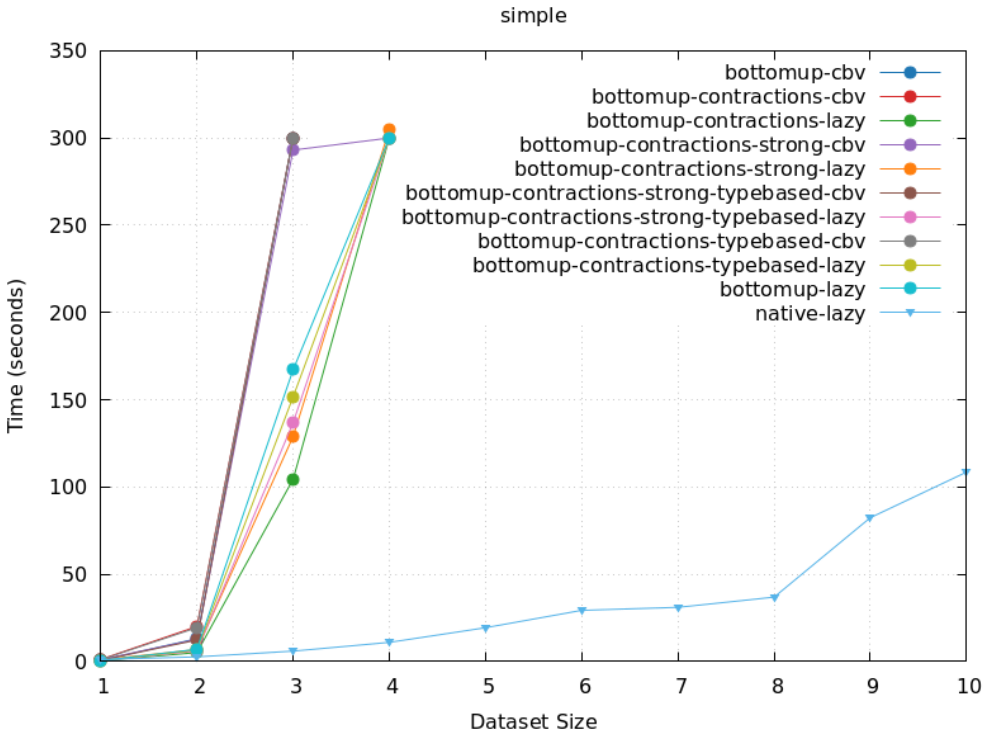
---

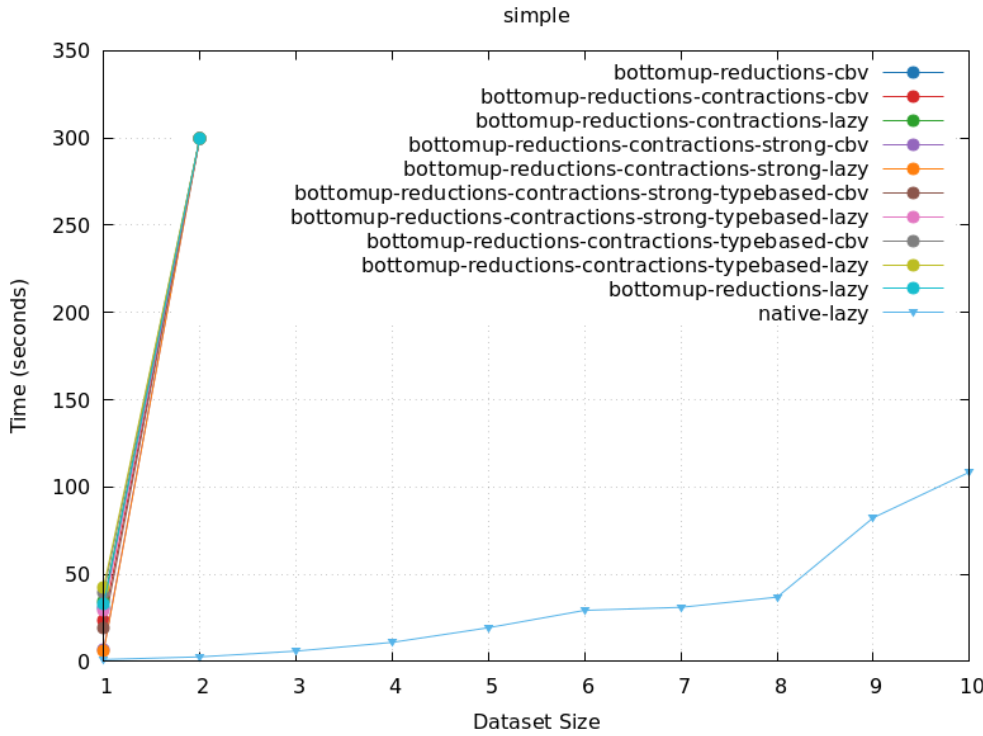
[a] глобальное состояние

Сравнение. Бенчмарки

- Синтетическая часть. Реализации алгоритма хеширования на языке Ursus, линейно увеличивающиеся по размеру кода
  - Simple, линейный код
  - Recursive, рекурсивный код
  - If, линейный код с условными операторами
  - IfAndRecursion, рекурсивный код с условными операторами
- Реальная часть. Верификация смарт-контракта мультисиг-кошелька из практики компании Pruvendo

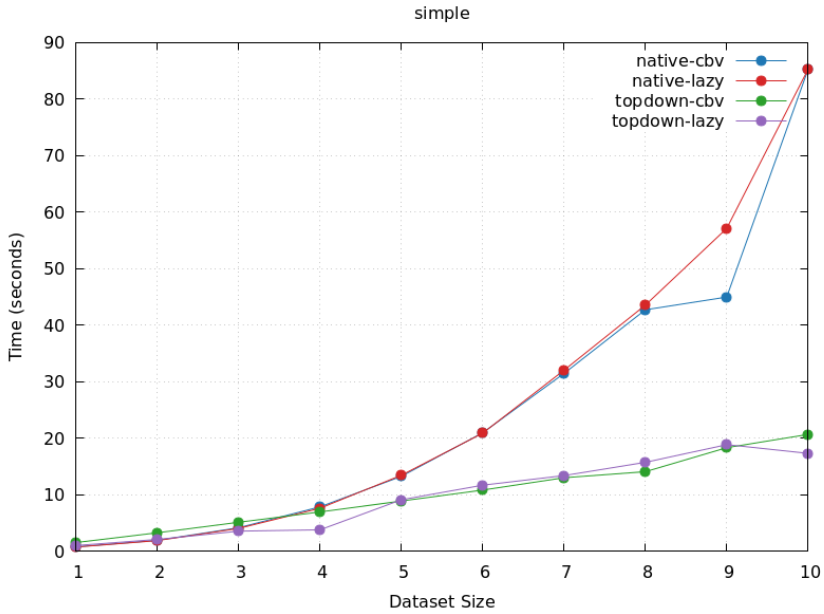
Сравнение. bottomup

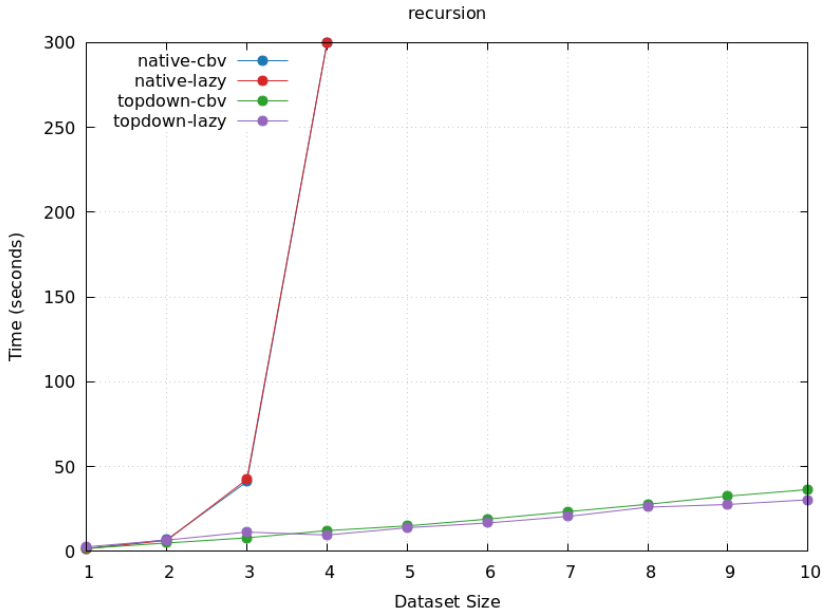


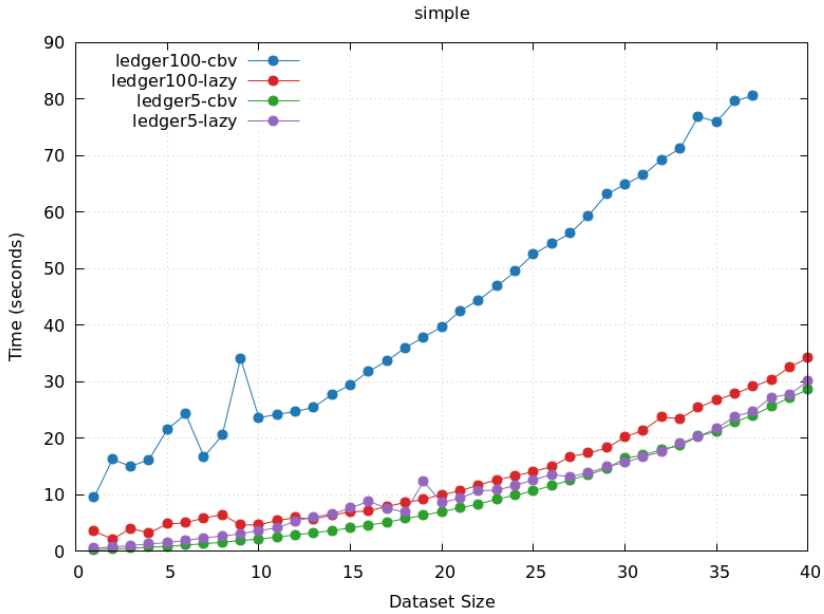


`bottomup` — все экспоненциальные

Сравнение. Базовые стратегии



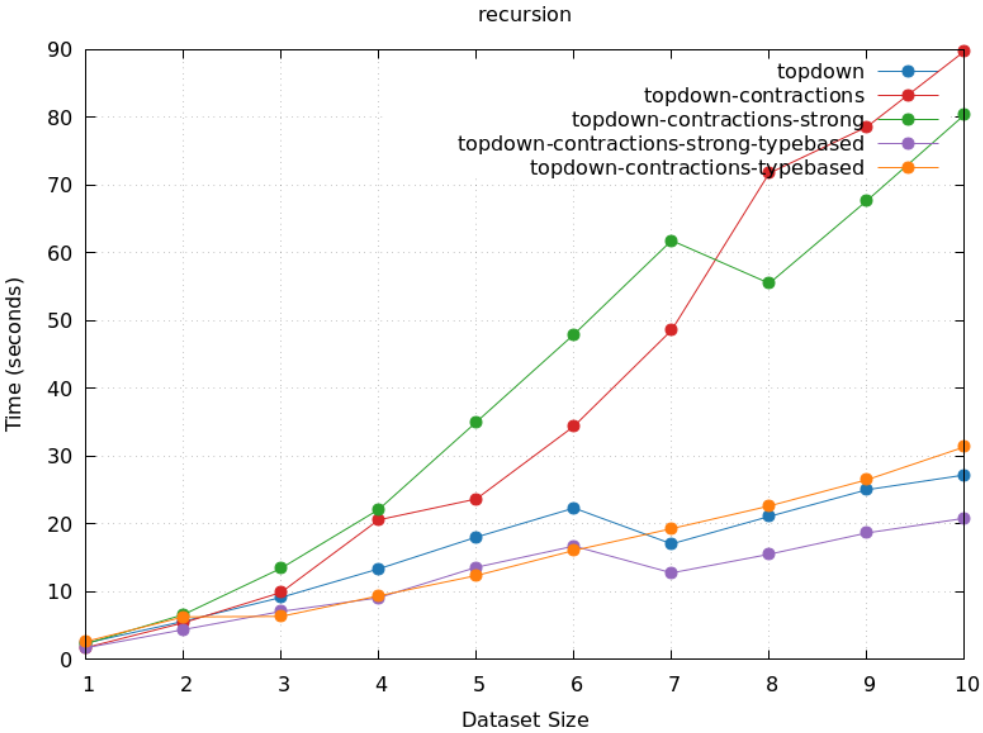


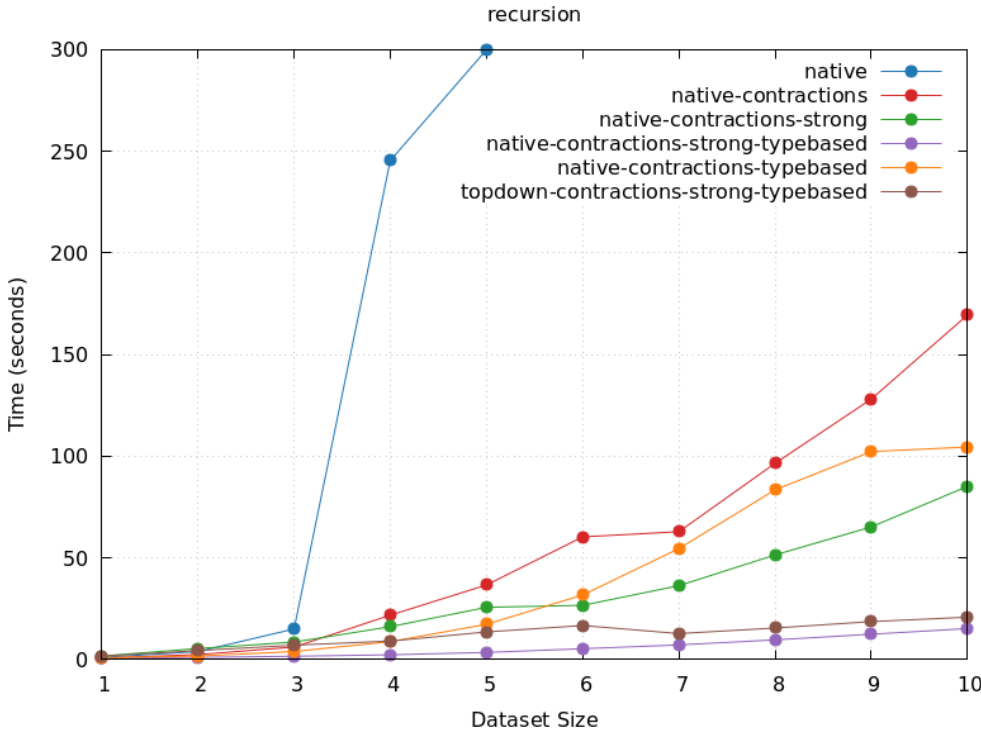


`native` на некоторых примерах экспоненциален

`lazy` значительно эффективнее при большой размерности состояния

Сравнение. Эвристические стратегии

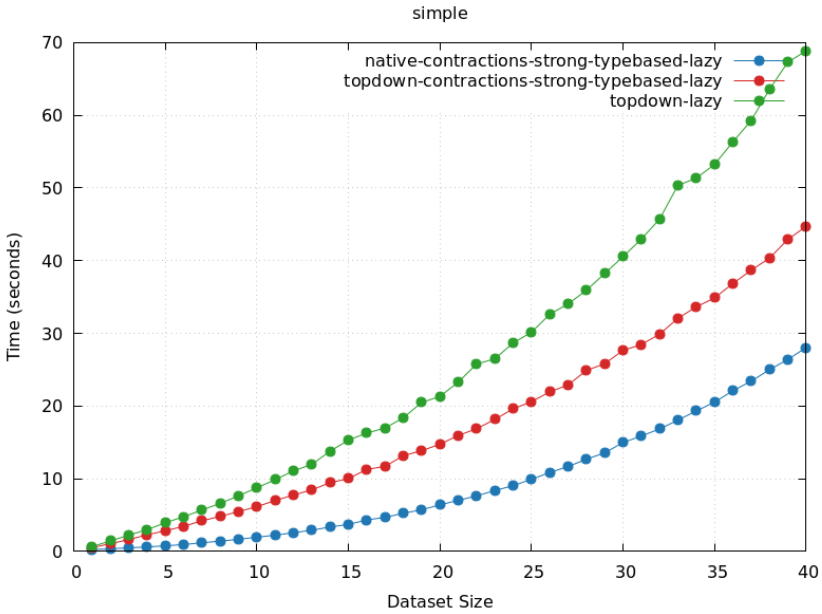


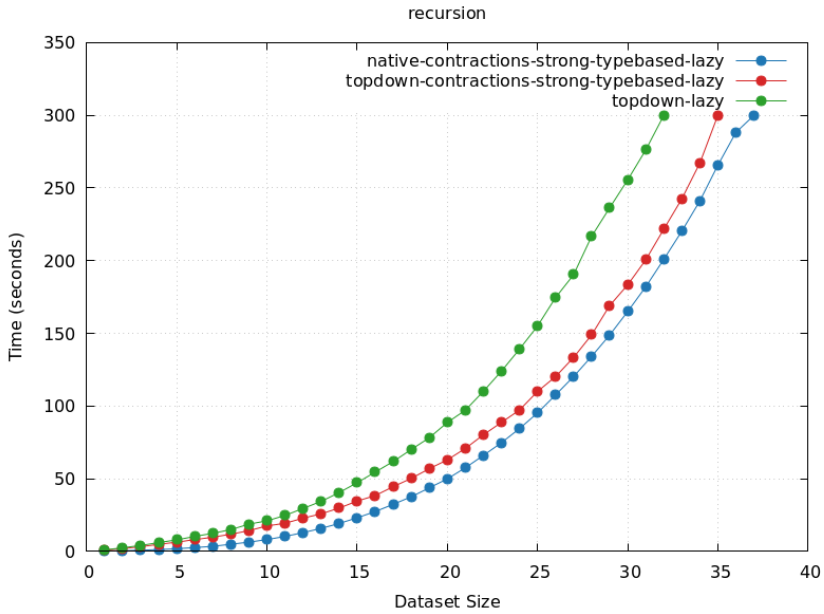


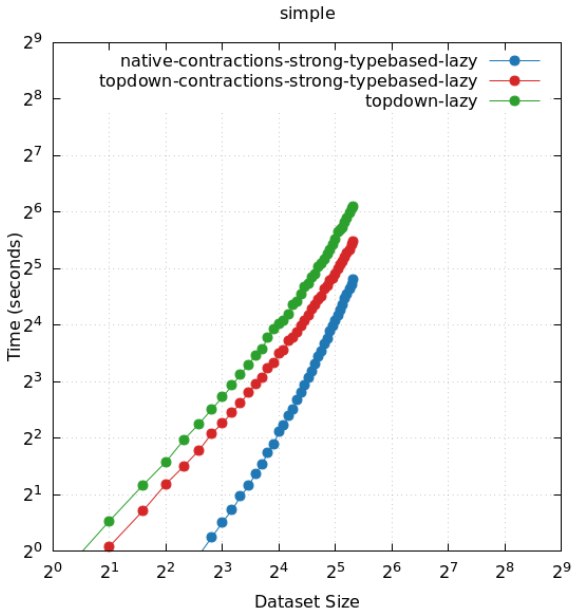
Графовые эвристики хуже, чем эвристики на типах

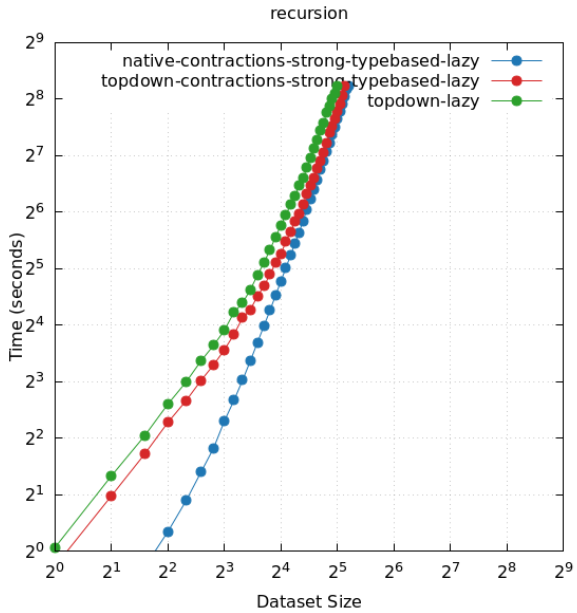
Эвристики на типах показывают лучший результат

Сравнение. Лучшие стратегии





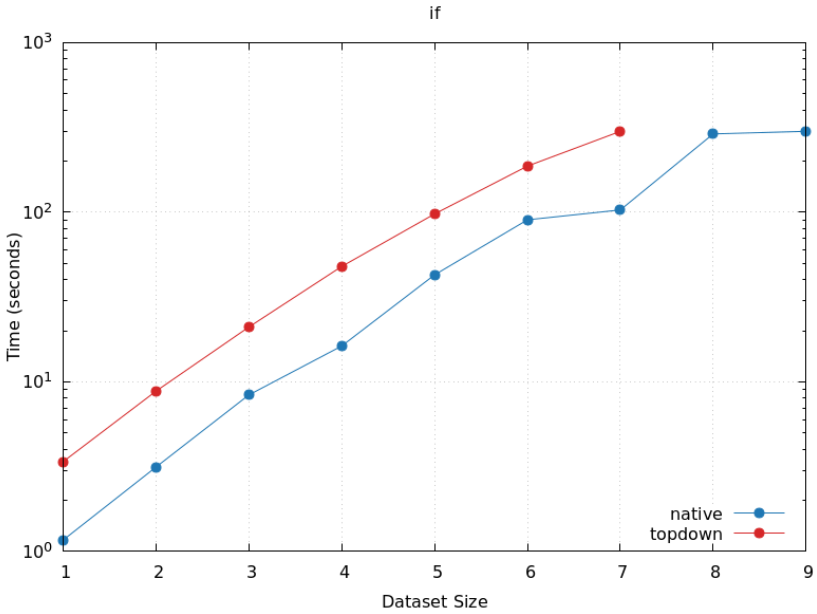


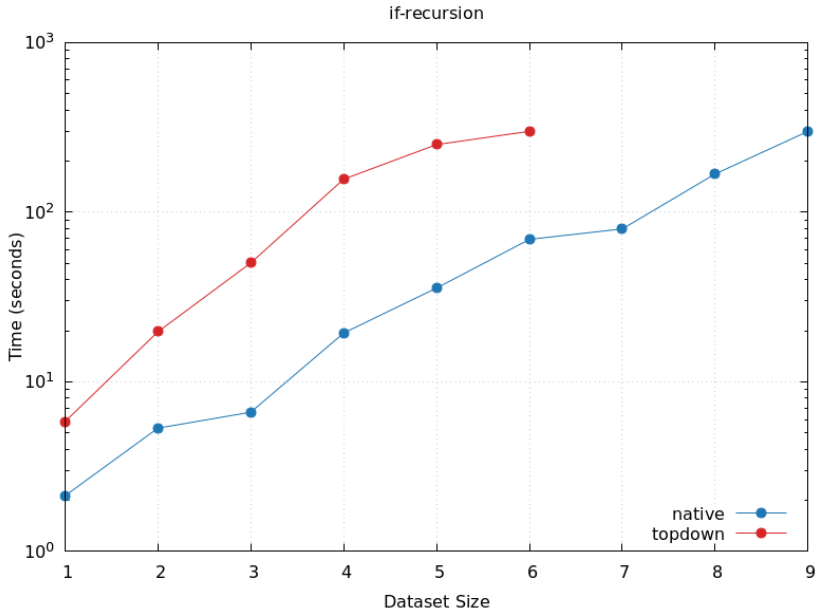


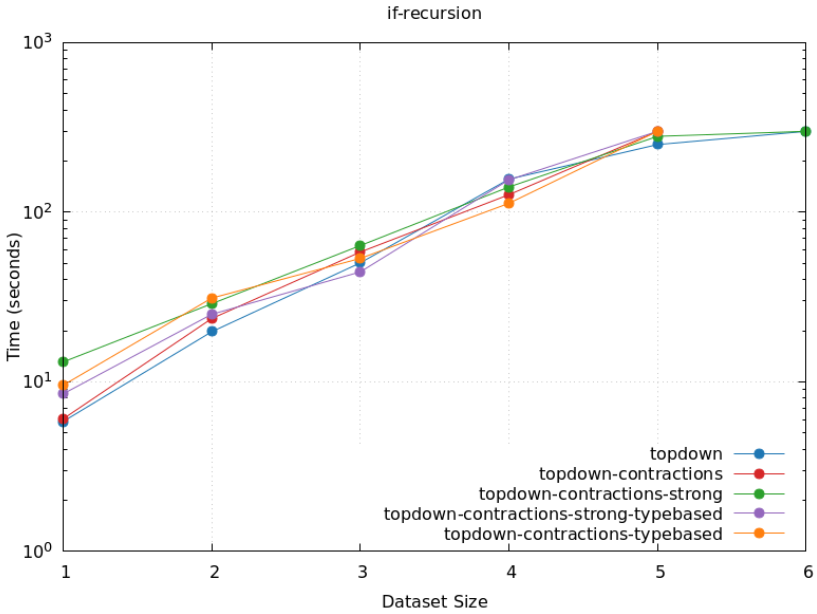
Результаты

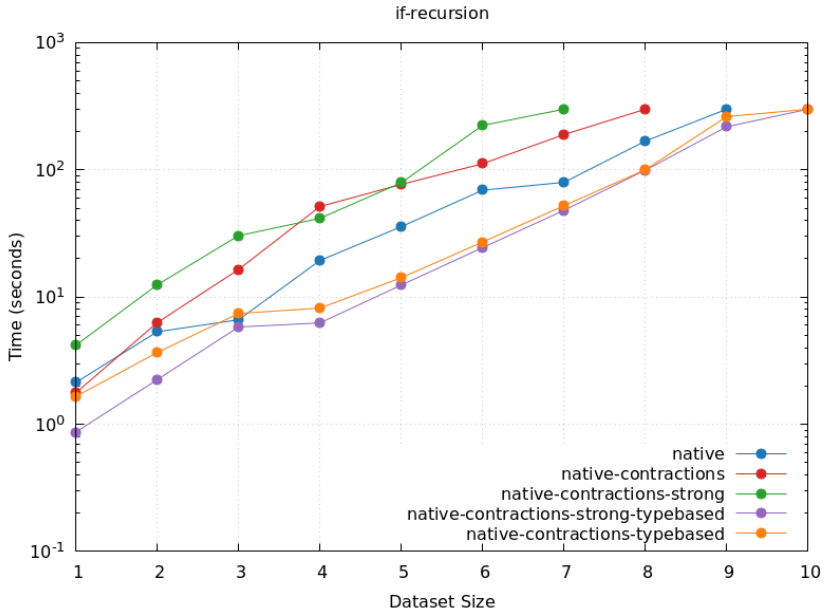
- Разработано 40 стратегий, все они реализованы на языке Ltac
- Создан набор программ на Ursus для бенчмаркинга
- Лучшая стратегии демонстрирует асимптотическое улучшение в производительности относительно базовой
- Самая эффективная стратегия на всех рассмотренных программах не показывает результат значимо хуже любой другой стратегии
- Стратегии будут использоваться в дальнейших проектах по верификации

Сравнение. Код с условными операторами









Сравнение. Реальные данные

Характеристики функций				
Функция	Строк	Вызовов функций	Глубина рекурсии	Условных операторов
submit	45	15	2	7
confirm	22	7	2	4
send	4	2	1	2

submitTransaction				
	NL	**NCSTL**	**TL**	**TCSTL**
1	0.90	**0.90**	1.46	1.24
2	0.95	**0.84**	1.05	1.04
3	0.99	**0.94**	1.44	1.43
4	1.86	**1.51**	3.58	2.93
5	1.44	**1.22**	12.26	7.99
6	23.40	**7.46**	46.16	58.54
7	23.63	**7.75**	41.19	58.16
QED	**11.51**	12.63	78.09	87.26
Σ	64.68	**33.25**	185.23	218.59

confirmTransaction				
	NL	**NCSTL**	**TL**	**TCSTL**
1	0.69	0.65	**0.53**	0.49
2	0.78	**0.70**	1.03	0.70
3	0.10	**0.09**	0.11	0.10
4	1.73	**1.44**	1.62	2.08
5	1.79	**1.25**	3.03	2.20
6	**0.20**	0.22	**0.20**	**0.20**
7	2.08	**1.35**	3.47	2.53
8	3.64	**2.03**	5.68	4.65
9	16.80	12.33	12.86	**11.79**
QED	**10.25**	10.34	31.74	26.73
Σ	38.06	**30.4**	60.27	51.47

sendTransaction				
	NL	**NCSTL**	**TL**	**TCSTL**
1	3.42	2.87	**2.30**	5.69
QED	2.5	**2.39**	3.8	4.53
Σ	5.92	**5.26**	6.1	10.22