Автоматический вывод индуктивных инвариантов программ с алгебраическими типами данных

Костюков Юрий Олегович

Научный руководитель:
д. т. н., доцент Кознов Дмитрий Владимирович

2024

Санкт-Петербургский
государственный
университет

Содержание

Верификация программ путём вывода индуктивных инвариантов

$x, y := 0, 0$

`while` $*$ `do`

$\quad y := y + x$

$\quad x := x + 1$

`assert`$(y \geq 0)$

Верификация программ путём вывода индуктивных инвариантов

$\{x = 0 \wedge y = 0\}$

`while` $*$ `do`

$\quad y := y + x$

$\quad x := x + 1$

$\{y \geq 0\}$

Верификация программ путём вывода индуктивных инвариантов

$\{x = 0 \wedge y = 0\}$

```
while * do
```

$y := y + x$

$x := x + 1$

$\{y \geq 0\}$

Как доказать корректность этой тройки Хоара?

Верификация программ путём вывода индуктивных инвариантов

$$\{x = 0 \wedge y = 0\}$$

```
while * do
```

$$y := y + x$$

$$x := x + 1$$

$$\{y \geq 0\}$$

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

$$\begin{aligned} x = 0 \wedge y = 0 &\rightarrow \varphi(x, y) \\ \varphi(x, y) \wedge x' = x + 1 \wedge y' = y + x &\rightarrow \varphi(x', y') \\ \varphi(x, y) &\rightarrow y \geq 0 \end{aligned}$$

Верификация программ путём вывода индуктивных инвариантов

$$\{x = 0 \wedge y = 0\}$$
$$\texttt{while } * \texttt{ do}$$
$$\quad y := y + x$$
$$\quad x := x + 1$$
$$\{y \geq 0\}$$

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?

$$\begin{aligned} x = 0 \wedge y = 0 &\rightarrow \varphi(x, y) \\ \varphi(x, y) \wedge x' = x + 1 \wedge y' = y + x &\rightarrow \varphi(x', y') \\ \varphi(x, y) &\rightarrow y \geq 0 \end{aligned}$$

Верификация программ путём вывода индуктивных инвариантов

$$\{x = 0 \land y = 0\}$$
$$\texttt{while } * \texttt{ do}$$
$$\quad y := y + x$$
$$\quad x := x + 1$$
$$\{y \geq 0\}$$

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?

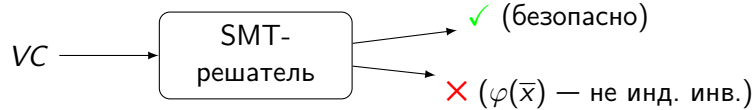
$$VC := \begin{cases} \forall x, y. \Big(x = 0 \land y = 0 \quad \to y \geq 0 \Big) \land \\ \forall x, y, x', y'. \Big(y \geq 0 \quad \land x' = x + 1 \land y' = y + x \quad \to y' \geq 0 \Big) \land \\ \forall x, y. \Big(y \geq 0 \quad \to y \geq 0 \Big) \end{cases}$$

Верификация программ путём вывода индуктивных инвариантов

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?



$$VC := \begin{cases} \forall x, y. \Big(x = 0 \wedge y = 0 \quad \rightarrow y \geq 0 \Big) \wedge \\ \forall x, y, x', y'. \Big(y \geq 0 \quad \wedge x' = x + 1 \wedge y' = y + x \quad \rightarrow y' \geq 0 \Big) \wedge \\ \forall x, y. \Big(y \geq 0 \quad \rightarrow y \geq 0 \Big) \end{cases}$$

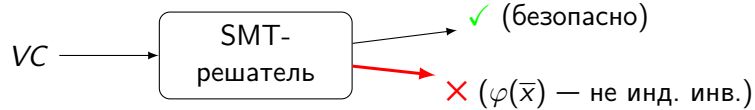
Верификация программ путём вывода индуктивных инвариантов

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?

SMT-решатель: Нет, индуктивность нарушается при $x \mapsto -1$



$$VC := \begin{cases} \forall x, y. \Big(x = 0 \wedge y = 0 \quad \rightarrow y \geq 0 \quad \Big) \wedge \\ \forall x, y, x', y'. \Big(y \geq 0 \quad \wedge x' = x + 1 \wedge y' = y + x \quad \rightarrow y' \geq 0 \quad \Big) \wedge \\ \forall x, y. \Big(y \geq 0 \quad \rightarrow y \geq 0 \Big) \end{cases}$$

Верификация программ путём вывода индуктивных инвариантов

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?

SMT-решатель: Нет, индуктивность нарушается при $x \mapsto -1$

Пользователь: А усиленная формула: $x \geq 0 \wedge y \geq 0$?

$$
\begin{aligned}
x = 0 \wedge y = 0 &\rightarrow \varphi(x, y) \\
\varphi(x, y) \wedge x' = x + 1 \wedge y' = y + x &\rightarrow \varphi(x', y') \\
\varphi(x, y) &\rightarrow y \geq 0
\end{aligned}
$$

Верификация программ путём вывода индуктивных инвариантов

Как доказать корректность этой тройки Хоара?

При помощи *пользовательского* **индуктивного инварианта** φ

Пользователь: $y \geq 0$ — индуктивный инвариант?

SMT-решатель: Нет, индуктивность нарушается при $x \mapsto -1$

Пользователь: А усиленная формула: $x \geq 0 \wedge y \geq 0$?

SMT-решатель: Да, эта формула является индуктивным инвариантом

$$
\begin{aligned}
x = 0 \wedge y = 0 &\rightarrow \varphi(x, y) \\
\varphi(x, y) \wedge x' = x + 1 \wedge y' = y + x &\rightarrow \varphi(x', y') \\
\varphi(x, y) &\rightarrow y \geq 0
\end{aligned}
$$

Дизъюнкты Хорна с ограничениями

Как автоматизировать вывод индуктивных инвариантов?

$$x = 0 \wedge y = 0 \rightarrow I(x, y)$$
$$I(x, y) \wedge x' = x + 1 \wedge y' = y + x \rightarrow I(x', y')$$
$$I(x, y) \rightarrow y \geq 0$$

Дизъюнкты Хорна с ограничениями

Как автоматизировать вывод индуктивных инвариантов?

Заменить пользовательскую формулу на **неинтерпретированный** символ I

$$x = 0 \wedge y = 0 \rightarrow I(x, y)$$
$$I(x, y) \wedge x' = x + 1 \wedge y' = y + x \rightarrow I(x', y')$$
$$I(x, y) \rightarrow y \geq 0$$

Дизъюнкты Хорна с ограничениями

Как автоматизировать вывод индуктивных инвариантов?

Заменить пользовательскую формулу на **неинтерпретированный** символ I

Дизъюнкты Хорна с ограничениями

$$x = 0 \wedge y = 0 \rightarrow I(x, y)$$
$$I(x, y) \wedge x' = x + 1 \wedge y' = y + x \rightarrow I(x', y')$$
$$I(x, y) \rightarrow y \geq 0$$

Дизъюнкты Хорна формально

Дизъюнкт Хорна C — это формула первого порядка следующего вида:

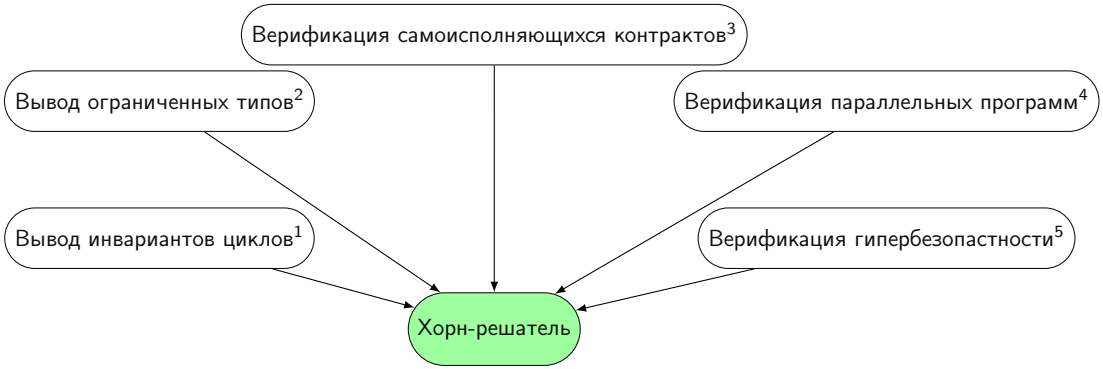
$$\varphi \wedge P_1(\overline{x}_1) \wedge \ldots \wedge P_n(\overline{x}_n) \to H$$

- **ограничение** φ — это формула теории
- **голова** H — это либо ложь $\bot$, либо атом $P(\overline{x})$
- $P_1, \ldots, P_n, P$ — это неинтерпретированные символы
- все переменные (неявно) универсально квантифицированы

Система дизъюнктов Хорна — это конъюнкция дизъюнктов Хорна

Хорн-решатель — программа, проверяющая выполнимость системы дизъюнктов

Применения Хорн-решателей



[1] Gurfinkel и др. The SeaHorn Verification Framework. CAV'15
[2] Tan и др. SolType: refinement types for arithmetic overflow in solidity. POPL'22
[3] Alt и др. SolCMC: Solidity Compiler's Model Checker. CAV'22
[4] Hoenicke и др. Thread Modularity at Many Levels. POPL'17
[5] Shemer и др. Property Directed Self Composition. CAV'19

Дизъюнкты Хорна над алгебраическими типами данных (АТД)

Пример программы на языке HASKELL:

```
data Nat = Z | S Nat
data List = nil | cons Nat List
drop Z xs = xs
drop _ nil = nil
drop (S n) (cons(_, xs)) = drop n xs
assert (¬∃ n xs . xs /= nil && drop n xs == drop (S n) xs)
```

Условия верификации в виде дизъюнктов Хорна над АТД:

$$\top \rightarrow drop(Z, xs, xs)$$

$$\top \rightarrow drop(S(n), nil, nil)$$

$$drop(n, xs, rs) \rightarrow drop(S(n), cons(x, xs), rs)$$

$$\neg(xs = nil) \wedge drop(n, xs, ys) \wedge drop(S(n), xs, ys) \rightarrow \bot$$

Индуктивный инвариант

Пусть $\mathcal{H}$ — модель теории АТД, $\mathcal{S}$ — система дизъюнктов Хорна.

Индуктивный инвариант $\mathcal{I}$ — расширение модели $\mathcal{I} = \langle \mathcal{H}, \mathcal{R} \rangle$, такое что $\mathcal{I} \models \mathcal{S}$.

Индуктивный инвариант

Пусть $\mathcal{H}$ — модель теории АТД, $\mathcal{S}$ — система дизъюнктов Хорна.

Индуктивный инвариант $\mathcal{I}$ — расширение модели $\mathcal{I} = \langle \mathcal{H}, \mathcal{R} \rangle$, такое что $\mathcal{I} \models \mathcal{S}$.

$$x = Z \wedge y = S(Z) \to inc(x, y)$$

$$x' = S(x) \wedge y' = S(y) \wedge inc(x, y) \to inc(x', y')$$

$$x = y \wedge inc(x, y) \to \bot$$

$$\mathcal{I}_1 = \mathcal{H}\Big\{ inc \mapsto \{(x, y) \mid y = S(x)\} \Big\}$$

$$\mathcal{I}_2 = \mathcal{H}\Big\{ inc \mapsto \{(x, y) \mid x \neq y \quad\} \Big\}$$

$$\mathcal{I}_3 = \dots$$

Индуктивные инварианты составляют решётку

Индуктивный инвариант

Пусть $\mathcal{H}$ — модель теории АТД, $\mathcal{S}$ — система дизъюнктов Хорна.

Индуктивный инвариант $\mathcal{I}$ — расширение модели $\mathcal{I} = \langle \mathcal{H}, \mathcal{R} \rangle$, такое что $\mathcal{I} \models \mathcal{S}$.

$$x = Z \wedge y = S(Z) \to inc(x, y)$$

$$x' = S(x) \wedge y' = S(y) \wedge inc(x, y) \to inc(x', y')$$

$$x = y \wedge inc(x, y) \to \bot$$

$$\mathcal{I}_1 = \mathcal{H}\Big\{ inc \mapsto \{(x, y) \mid y = S(x)\Big\}$$

$$\mathcal{I}_2 = \mathcal{H}\Big\{ inc \mapsto \{(x, y) \mid x \neq y \quad \Big\}$$

$$\mathcal{I}_3 = \dots$$

Как представлять эти **бесконечные** множества?

Индуктивный инвариант

Пусть $\mathcal{H}$ — модель теории АТД, $\mathcal{S}$ — система дизъюнктов Хорна.

Индуктивный инвариант $\mathcal{I}$ — расширение модели $\mathcal{I} = \langle \mathcal{H}, \mathcal{R} \rangle$, такое что $\mathcal{I} \models \mathcal{S}$.

$$x = Z \wedge y = S(Z) \rightarrow inc(x, y)$$

$$x' = S(x) \wedge y' = S(y) \wedge inc(x, y) \rightarrow inc(x', y')$$

$$x = y \wedge inc(x, y) \rightarrow \bot$$

$$\mathcal{I}_1 = \mathcal{H}\{inc \mapsto \boxed{y = S(x)}\}$$

$$\mathcal{I}_2 = \mathcal{H}\{inc \mapsto \boxed{\neg(x = y)}\}$$

$$\mathcal{I}_3 = \ldots$$

Как представлять эти **бесконечные** множества?

Инварианты обычно представляются в логике первого порядка (ЛПП)
ЛПП задаёт т.н. *класс элементарных инвариантов*

Проблема выразимости класса элементарных инвариантов

$$x = Z \to even(x)$$
$$even(y) \wedge x = S(S(y)) \to even(x)$$
$$even(x) \wedge even(S(x)) \to \bot$$

Проблема выразимости класса элементарных инвариантов

$$x = Z \rightarrow even(x)$$
$$even(y) \wedge x = S(S(y)) \rightarrow even(x)$$
$$even(x) \wedge even(S(x)) \rightarrow \bot$$

Невыполнима

Проблема выразимости класса элементарных инвариантов

$$x = Z \rightarrow even(x)$$
$$even(y) \wedge x = S(S(y)) \rightarrow even(x)$$
$$even(x) \wedge even(S(x)) \rightarrow \bot$$

Выполнима

Невыполнима

Проблема выразимости класса элементарных инвариантов

$$x = Z \to even(x)$$
$$even(y) \wedge x = S(S(y)) \to even(x)$$
$$even(x) \wedge even(S(x)) \to \bot$$

Выполнима

Невыполнима

Инвариант выразим элементарно

$\mathcal{I}(even) \equiv \varphi$

Проблема выразимости класса элементарных инвариантов

$$x = Z \to even(x)$$
$$even(y) \wedge x = S(S(y)) \to even(x)$$
$$even(x) \wedge even(S(x)) \to \bot$$

Проблема выразимости класса элементарных инвариантов

$$x = Z \rightarrow even(x)$$

$$even(y) \wedge x = S(S(y)) \rightarrow even(x)$$

$$even(x) \wedge even(S(x)) \rightarrow \bot$$

Вывод инвариантов в языке АТД **расходится**!

например, Z3/SPACER расходится

Выполнима

Невыполнима

Инвариант выразим элементарно

$\mathcal{I}(even) \equiv \varphi$

Инвариант невыразим элементарно

???

Проблема выразимости класса элементарных инвариантов

$$x = Z \rightarrow even(x)$$
$$even(y) \wedge x = S(S(y)) \rightarrow even(x)$$
$$even(x) \wedge even(S(x)) \rightarrow \bot$$

Проблема: класс элементарных инвариантов невыразителен

Выполнима

Невыполнима

Инвариант выразим элементарно

$\mathcal{I}(even) \equiv \varphi$

Инвариант невыразим элементарно

???

Постановка задачи

Цель работы — предложение новых классов индуктивных инвариантов для программ с АТД и создание для них методов автоматического вывода. **Задачи:**

1. Предложить методы вывода инвариантов в существующих классах
2. Предложить новый класс индуктивных инвариантов программ с АТД
3. Предложить метод автоматического вывода инвариантов в новом классе
4. Выполнить пилотную программную реализацию предложенных методов
5. Провести экспериментальное сопоставление реализованного инструмента с существующими на представительном тестовом наборе

Результаты

1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность
2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность
3. Предложен новый класс инвариантов, основанный на булевой комбинации элементарных и регулярных инвариантов
4. Предложен метод совместного вывода инвариантов в этом классе посредством вывода инвариантов в подклассах и доказана его корректность
5. Проведено теоретическое сравнение рассмотренных классов инвариантов
6. Выполнена пилотная реализация предложенных методов на языке F# в рамках инструмента RInGen
   Разработанный инструмент решил из бенчмарка «Tons of Inductive Problems» в 3.74 раза больше задач, чем наилучший из существующих инструментов

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 1. Устранить АТД ограничения при помощи унификации и введения новых дизъюнктов

Система дизъюнктов

$$\top \to even(Z)$$
$$even(y) \to even(S(S(y)))$$
$$even(x) \wedge even(S(x)) \to \bot$$

АТД ограничения устранены

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 2. Трансформировать систему в формулу ЛПП введением логических связок

Система дизъюнктов как формула ЛПП

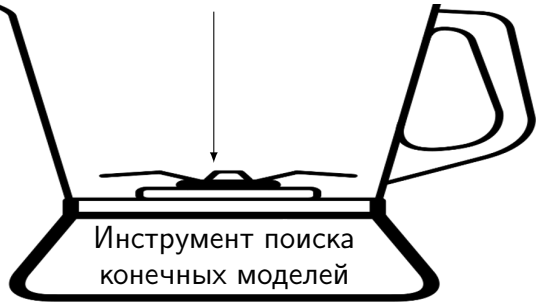
$$\begin{aligned} &\top \to \mathit{even}(Z)) \wedge \\ &\forall y.(\mathit{even}(y) \to \mathit{even}(S(S(y)))) \wedge \\ \forall x.(&\mathit{even}(x) \wedge \mathit{even}(S(x)) \to \bot) \end{aligned}$$

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 3. Передать формулу в сторонний инструмент поиска конечных моделей

Система дизъюнктов как формула ЛПП

$$\top \to even(Z)) \wedge$$
$$\forall y.(even(y) \to even(S(S(y)))) \wedge$$
$$\forall x.(even(x) \wedge even(S(x)) \to \bot)$$



Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 3. Передать формулу в сторонний инструмент поиска конечных моделей

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 4. По конечной модели построить автомат над деревьями

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

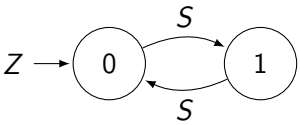
Этап 4. По конечной модели построить автомат над деревьями

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 4. По конечной модели построить автомат над деревьями

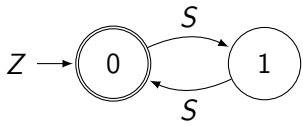
$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$



Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 4. По конечной модели построить автомат над деревьями

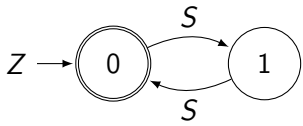
$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$



Результат 1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 4. По конечной модели построить автомат над деревьями

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$



Язык построенного автомата является регулярным инвариантом исходной системы

$$\mathcal{I}(even) = \mathcal{L}(\mathcal{A}) = \{S^{2n}(Z) \mid n \geq 0\}$$

Результат 2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Регулярные языки не позволяют представлять синхронные отношения

$$\top \to lt(Z, S(x))$$
$$lt(x, y) \to lt(S(x), S(y))$$
$$lt(x, y) \wedge lt(y, x) \to \bot$$

Результат 2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 1. Устранить АТД ограничения при помощи унификации и введения новых дизъюнктов

Регулярные языки не позволяют
представлять синхронные отношения

$$\top \to lt(Z, S(x))$$
$$lt(x, y) \to lt(S(x), S(y))$$
$$lt(x, y) \wedge lt(y, x) \to \bot$$

Результат 2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 2. Построить декларативное описание синхронного автомата, выражающего инвариант системы

Регулярные языки не позволяют представлять синхронные отношения

$$\top \to lt(Z, S(x))$$
$$lt(x, y) \to lt(S(x), S(y))$$
$$lt(x, y) \wedge lt(y, x) \to \bot$$

Декларативное описание синхронного автомата

$$R(q) \to R(p(d(f, g, q), d(f, g, q)))$$
$$R(p(q_1, q_2)) \to \Big(F(q_1) \to F(d(S, S, q_2))\Big)$$
$$\dots$$

Результат 2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 3. Передать формулу в сторонний инструмент поиска конечных моделей

Регулярные языки не позволяют представлять синхронные отношения

$$\top \to lt(Z, S(x))$$
$$lt(x, y) \to lt(S(x), S(y))$$
$$lt(x, y) \wedge lt(y, x) \to \bot$$

Декларативное описание синхронного автомата

$$R(q) \to R(p(d(f, g, q), d(f, g, q)))$$
$$R(p(q_1, q_2)) \to \Big(F(q_1) \to F(d(S, S, q_2))\Big)$$
$$\dots$$

Результат 2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность

Этап 4. По конечной модели построить автомат над деревьями

Регулярные языки не позволяют представлять синхронные отношения

$$\top \to lt(Z, S(x))$$
$$lt(x, y) \to lt(S(x), S(y))$$
$$lt(x, y) \wedge lt(y, x) \to \bot$$

Декларативное описание синхронного автомата

$$R(q) \to R(p(d(f, g, q), d(f, g, q)))$$
$$R(p(q_1, q_2)) \to \Big(F(q_1) \to F(d(S, S, q_2))\Big)$$
$$\dots$$

Из модели можно извлечь определение синхронного автомата

$$A = \Big\langle \{0, 1\}, \Sigma_F^{\leqslant 2}, \{1\}, \Delta \Big\rangle$$

$$\langle Z, Z\rangle \mapsto 0 \qquad Z \mapsto 0 \qquad S(q) \mapsto 0$$
$$\langle Z, S\rangle(q) \mapsto 1 \qquad \langle S, Z\rangle(q) \mapsto 0 \qquad \langle S, S\rangle(q) \mapsto q$$

$$\mathcal{L}(A) = \big\{\langle S^n(Z), S^m(Z)\rangle \mid n < m\big\}$$

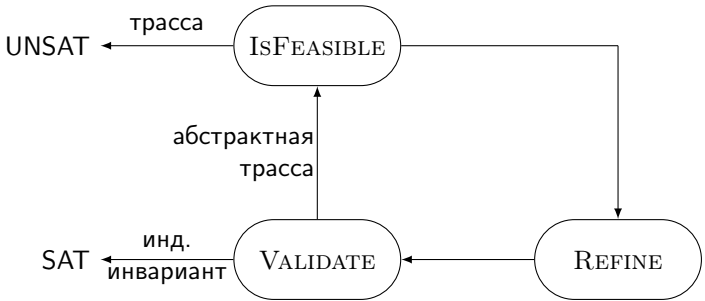
Результат 3. Предложен новый класс инвариантов, основанный на булевой комбинации элементарных и регулярных инвариантов

Новый класс *комбинированных инвариантов* представляется формулами вида:

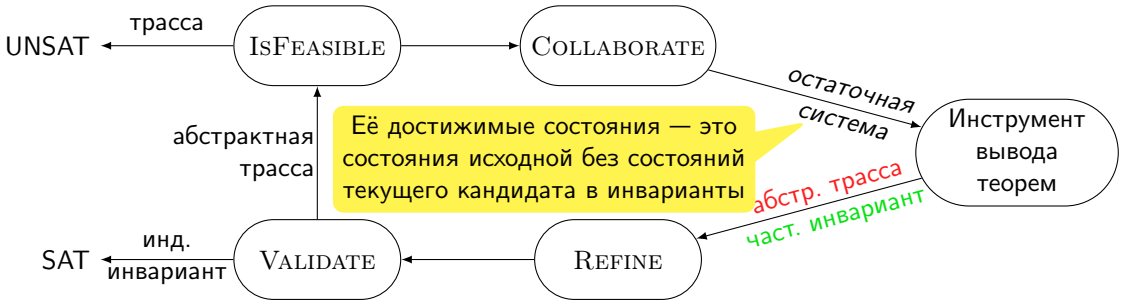
$$\varphi ::= \overline{t} \in \mathcal{L}(A) \mid t = t' \mid \neg\psi \mid \psi \wedge \psi' \mid \psi \vee \psi'$$

- $\overline{t} \in \mathcal{L}(A)$ — принадлежность кортежа термов регулярном языку автомата A

Результат 4. Предложен метод совместного вывода инвариантов в этом классе посредством вывода инвариантов в подклассах и доказана его корректность



Результат 4. Предложен метод совместного вывода инвариантов в этом классе посредством вывода инвариантов в подклассах и доказана его корректность



Результат 5. Проведено теоретическое сравнение рассмотренных классов инвариантов

Результаты, доказанные в диссертации; результаты с тривиальным доказательством

Свойство \ Класс	Elem	SizeElem	Reg	Reg$_+$	Reg$_\times$	ElemReg
Замкнут по $\cap$	Да	Да	Да	Да	Да	Да
Замкнут по $\cup$	Да	Да	Да	Да	Да	Да
Замкнут по $\setminus$	Да	Да	Да	Да	Да	Да
Разрешимо $\overline{t} \in I$	Да	Да	Да	Да	Да	Да
Разрешимо $I = \varnothing$	Да	Да	Да	Да	Да	Да
Выразимы рекурсивные отношения	Нет	Частично	Да	Да	Да	Да
Выразимы синхронные отношения	Да	Да	Нет	Частично	Да	Да

Класс	Elem	SizeElem	Reg	Reg$_+$	Reg$_\times$	ElemReg
Elem	$\varnothing$	$\varnothing$	*lr*	*lr*	*lr*	$\varnothing$
SizeElem	∞	$\varnothing$	*lr*	*lr*	*lr*	*lt*
Reg	*even*	*even*	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
Reg$_+$	*even*	*even*	∞	$\varnothing$	$\varnothing$	*lt*
Reg$_\times$	*even*	*even*	∞	∞	$\varnothing$	*lt*
ElemReg	∞	*even*	∞	*lr*	*lr*	$\varnothing$

Реализация

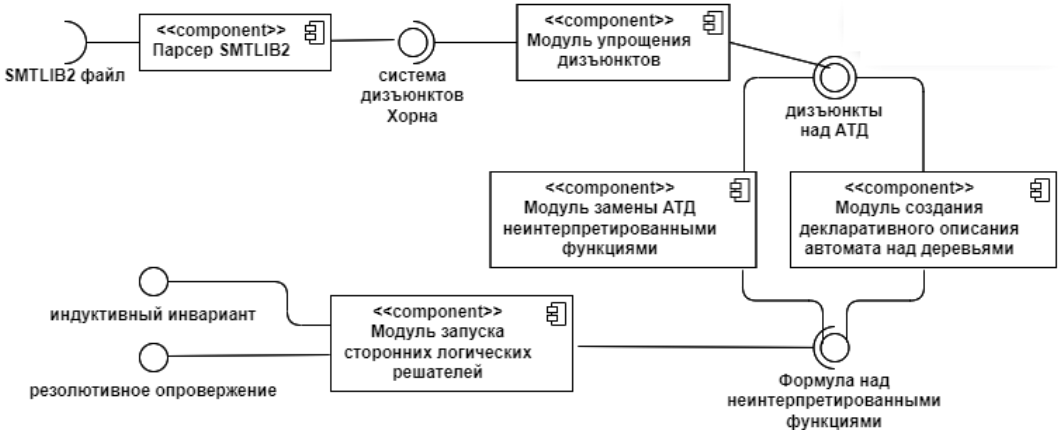
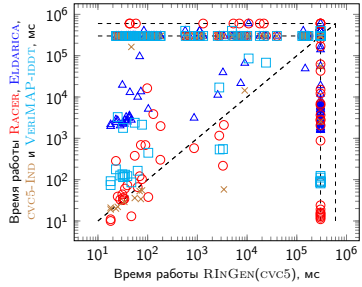
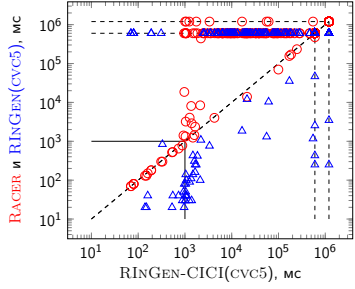


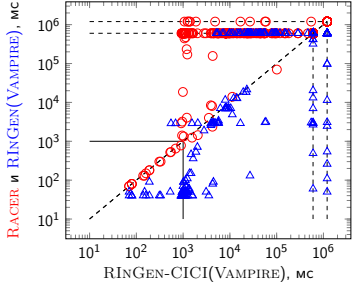
Рис.: Хорн-решатель RInGen: `https://github.com/Columpio/RInGen`

Эксперименты

Инструмент	SAT	UNSAT
Racer	26	22
Eldarica	46	12
VeriCaT	16	10
cvc5-Ind	0	13
RInGen(cvc5)	25	21
RInGen(Vampire)	135	46
RInGen-Sync	43	21
RInGen-CICI(cvc5)	117	19
RInGen-CICI(Vampire)	189	28







Результаты

1. Предложен метод вывода регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность
2. Предложен метод вывода синхронных регулярных инвариантов при помощи поиска конечных моделей и доказана его корректность
3. Предложен новый класс инвариантов, основанный на булевой комбинации элементарных и регулярных инвариантов
4. Предложен метод совместного вывода инвариантов в этом классе посредством вывода инвариантов в подклассах и доказана его корректность
5. Проведено теоретическое сравнение рассмотренных классов инвариантов
6. Выполнена пилотная реализация предложенных методов на языке F# в рамках инструмента RInGen
   Разработанный инструмент решил из бенчмарка «Tons of Inductive Problems» в 3.74 раза больше задач, чем наилучший из существующих инструментов

Соответствие результатов паспорту специальности 2.3.5

Результаты соответствуют направлению исследования № 1

- Модели, **методы и алгоритмы** проектирования, анализа, трансформации, **верификации** и тестирования **программ** и программных систем

из паспорта специальности.

Научная новизна

1. Впервые предложен класс инвариантов, основанный на булевой комбинации элементарных и регулярных инвариантов
2. Впервые предложен алгоритм вывода инвариантов для программ с АТД, основанный на поиске конечных моделей
3. Предложен новый алгоритм совместного вывода инвариантов в комбинации классов инвариантов на базе методов вывода инвариантов в подклассах
4. Впервые введены и доказаны леммы о «накачке» для языков первого порядка в сигнатуре теории АТД

Публикации по теме диссертации

[1] Kostyukov Yurii, Mordvinov Dmitry, Fedyukovich Grigory. Beyond the Elementary Representations of Program Invariants over Algebraic Data Types // Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. — PLDI 2021. — New York, NY, USA : Association for Computing Machinery, 2021. — P. 451–465.

[2] Kostyukov Yurii, Mordvinov Dmitry, Fedyukovich Grigory. Collaborative Inference of Combined Invariants // Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning / Ed. by Ruzica Piskac, Andrei Voronkov. — Vol. 94 of EPiC Series in Computing. — EasyChair, 2023. — P. 288–305.

[3] Автоматическое доказательство корректности программ с динамической памятью / Юрий Олегович Костюков, Константин Аланович Батоев, Дмитрий Александрович Мордвинов и др. // Труды Института системного программирования РАН. — 2019. — Т. 31, № 5. — С. 37–62.

[4] Генерация слабейших предусловий программ с динамической памятью в символьном исполнении / Александр Владимирович Мисонижник, Юрий Олегович Костюков, Михаил Павлович Костицын и др. // Научно-технический вестник информационных технологий, механики и оптики. — 2022. — Т. 22, № 5. — С. 982–991.

Выступления по теме диссертации

- Международный семинар HCVS 2021 (28 марта 2021, Люксембург)
- Семинар компании Huawei (18-19 ноября 2021, Санкт-Петербург)
- Ежегодный внутренней семинар JetBrains Research (18 декабря 2021, Санкт-Петербург)
- Конференция PLDI 2021 (23-25 июня 2021, Канада)
- Внутренний семинар Венского технического университета (3 июня 2022, Австрия)
- Конференция LPAR 2023 (4-9 июня 2023, Колумбия)

Разработанный инструмент в 2021 и 2022 годах занял, соответственно, 2 и 1 место на АТД секции международных соревнований CHC-COMP.

Сравнение Хорн-решателей с поддержкой АТД

Инструмент	Класс инвариантов	Метод	Возвращает инвариант	Полностью автоматический
Spacer	Elem	IC3/PDR	Да	Да
Racer	CatElem	IC3/PDR	Нет	Нет
Eldarica	SizeElem	CEGAR	Да	Да
VeriCaT	–	Трансф.	Нет	Да
HoIce	Elem	ICE	Да	Да
RCHC	Reg_{+}	ICE	Да	Да
RInGen(cvc5)	Reg	Трансф. + FMF	Да	Да
RInGen(Vampire)	–	Трансф. + Насыщение	Нет	Да
RInGen-Sync	$\text{Reg}_{\times}$	Трансф. + FMF	Да	Да
RInGen-CICI(cvc5)	ElemReg	CEGAR($\mathcal{O}$)	Да	Да
RInGen-CICI(Vampire)	–	CEGAR($\mathcal{O}$)	Нет	Да